THE EXQUIRES (EXTENSIBLE QUANTITATIVE IMAGE RESAMPLING) TEST SUITE: IMPACT OF THE DOWNSAMPLER, DIFFERENCE METRIC, TEST IMAGE, RESAMPLING RATIO AND COLOUR SPACE ON UPSAMPLER RANK

BY

ADAM TURCOTTE

THESIS SUBMITTED IN PARTIAL FULFILMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE (MSC) OF COMPUTATIONAL SCIENCES

SCHOOL OF GRADUATE STUDIES
LAURENTIAN UNIVERSITY
SUDBURY, ONTARIO

Your file Votre référence
ISBN: 978-0-494-91873-9

Our file Notre référence
ISBN: 978-0-494-91873-9

Abstract

Image re-enlargement tests are used to evaluate the accuracy of image resampling methods. In such tests, an image is reduced (downsampled), enlarged back to the original width and height, and the difference between the re-enlargement and the original is measured. The smaller the difference, the better the resampling method is deemed to be.

The EXtensible QUantitative Image RESampling test suite, written for this thesis, is used to perform re-enlargement tests and analyze the results. EXQUIRES is FLOSS (Free/Libre and Open Source Software), its Python code is transparent, and tests performed with it are self-documenting. Additional image enlargement methods, image difference metrics and test images can be plugged in. A number of image difference metrics based on the sRGB, XYZ and L*C*h colour spaces are included, and an improved algorithm for the computation of the MSSIM (Mean Structural SIMilarity) metric is implemented.

EXQUIRES can compute aggregate rankings. It is used to rank sixty four filters as natural image reconstructors. In addition, EXQUIRES can compute Spearman cross-correlations, and the impact of various factors is assessed with them.

Commonly used image difference metrics are far from being in perfect agreement. If, for example, XYZ RMSE, that is, Root Mean Square Error based on differences in the XYZ linear light colour space, is the chosen image difference metric, the top ranked upsampling methods are Nohalo face split subdivision with Locally Bounded Bicubic finish (Nohalo-LBB), Hamming-windowed Sinc 4 (Hamming 4), Cosine-windowed Sinc 3 (Cosine 3), Hann 4 and Lanczos 4. With sRGB RMSE, top ranks go to Nohalo-LBB, Bartlett 2, two

variants of Kaiser 3, and Bohman 3. In fact, the Spearman rank correlation of these two standard metrics is only 0.510.

The choice of downsampler is also found to have significant impact. Here is an extreme example: Upsampler rankings are almost reversed if downsampling is performed with nearest neighbour decimation instead of box filtering. When downsampling with nearest neighbour, the top methods are Mitchell-Netravali, bilinear, Nohalo-LBB, Hann 2 and Elliptical Weighted Averaging (EWA) Robidoux, all methods with mild or non-existent halos. On the other hand, when downsampling with effective low pass filters, top ranks go to Welch 4, Cosine 4, Lanczos 4, Hamming 4 and EWA Catmull-Rom, all with significant haloing.

The above ranks and cross-correlations were obtained with de-noised high quality digital photographs and one archival scan stored in 16-bit sRGB and downsized as well as re-enlarged through linear light. Strong evidence is given that the choice of colour space through which the enlargement is performed deserves careful consideration. For example, re-enlarging sRGB images obtained by downsampling through linear light is better done by filtering sRGB values directly, but re-enlarging sRGB images obtained by downsampling sRGB pixel values directly is better done through linear light. With all downsamplers included, Hamming 4, Bartlett 4, Hann 4, Lanczos 3 and Cosine 3 are top ranked in both mixed toolchains. The mixed toolchains actually produce highly correlated rankings.

These results make clear that the experimental set up has a large impact on rankings. For example, it is expected that adding noise or compression artifacts, or using different types of test images (black on white typeset text, for example), would completely change the results.

To my friends and family for inspiring me to go far,

and to my supervisor for giving me a direction to travel.

Acknowledgements

Without the assistance of Dr. John Cupitt of Imperial College, London, and Anthony Thyssen of Griffith University, this thesis would be very different. Their willingness to add needed features to the VIPS and ImageMagick image processing libraries and their answers to my countless questions contributed greatly to the quality of the EXQUIRES test suite. Cristy and Glenn Randers-Pehrson also deserve thanks for tirelessly improving ImageMagick.

I am extremely grateful to everyone on the advisory committee for their guidance. I would especially like to thank my supervisor, Dr. Nicolas Robidoux, for providing me with an overwhelming amount of support and investing considerable time and effort into this research, and my co-supervisor, Dr. Ralf Meyer, for taking the time to answer my questions and for providing me with some excellent suggestions. Without their assistance, this thesis would not have been possible. I would also like to thank Dr. Julien Dompierre and Dr. Torsten Möller for providing useful feedback regarding the organization of the thesis and other helpful revision comments.

I would like to thank those who contributed digital photographs to the 16bit840x840images test bank: Jean-François Avon, Anthony Barnett, Dr. John Cupitt, Jana Duncan, Dr. Minglun Gong, Holly Graham, Henry Ho, Dr. Kirk Martinez, Michael Muré, Mukund Sivaraman and Luiz E. Vasconcellos.

Furthermore, I would like to thank Mark Thompson of Laurentian University, Alexandre Prokoudine, editor of Libre Graphics World, and Dr. Kirk Martinez of the University of

Southampton for providing computer hardware on which to run EXQUIRES prototypes.

Even though the research I performed on Jacobian adaptivity in the course of my Masters is not included in this thesis, I would like to thank Øyvind Kolås, lead developer of the GEGL image compositing library, for his assistance in implementing it.

Finally, I am grateful for an NSERC Alexander Graham Bell Canada Graduate Scholarship, for an Ontario Graduate Scholarship, for Google Summer of Code 2010 funding awarded to GIMP (GNU Image Manipulation Program), as well as for the financial support provided by Laurentian University.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Image resampling

Image resampling essentially consists of combining known pixel values so as to obtain new pixel values at locations where none are provided (between the original pixel locations, as in image enlargement), or to obtain a pixel value suitable for different viewing conditions (as in image reduction). It is performed when zooming in on a digital photograph on a tablet or computer screen, watching a classic sitcom in HD (high definition), registering (aligning) two medical images taken at different times with slightly different positioning and state of the patient in order to compare them, using the software zoom in a digital camera, correcting lens distortion, warping a portrait for artistic purposes, assembling multiple satellite views of the Earth or the Sun, producing image thumbnails, making an image fit a browser window, processing one's desktop so that it is shown within a computer monitor taking into account the type and configuration of the screen's physical pixels, etc.

A very large number of methods—both general purpose and specialized—have been and are still being developed, which brings up a question: How does one measure the quality of an image resampling method?

1.2 Quantitative evaluation of image resampling methods

Although one can test the suitability of an image resampling method for a specific task by measuring its performance directly within the application, the quantitative evaluation of the

quality of a general purpose image resampling method is generally performed with one of the following methods:

Re-enlargement (a.k.a. reconstruction) tests, in which a gold standard image—a reference image labelled as "perfect" when computing errors—is first downsampled (downsized, that is, reduced and shrunk, so that there are fewer pixels in all directions) with some filter (box filtering, for example), and then re-enlarged to its original size with the evaluated resampling method, at which point the original and re-enlarged images are compared [1–18]

Rotation tests, in which a gold standard image is repeatedly rotated with the evaluated resampling method in such a way that the total rotation angle adds up to an exact multiple of 90 degrees, after which the overall rotation is exactly undone and cropping performed on both the result and original image before comparing them to minimize the impact of the abyss (pixels that lie outside of the extent of the gold standard image) [1, 19–26]

Sub-pixel translation tests, in which a gold standard image is repeatedly translated in such a way that the total translation is an exact multiple of the inter-pixel distance both horizontally and vertically, following which the overall translation is exactly undone before cropping and comparing [1, 21, 22, 25–27]

Frequency response tests, in which the frequency response of the filters is measured [1, 28–32]. Such tests do not directly measure the performance of an image resampling method, since they do not involve "real" images. The same can be said of plotting the impulse response and, to a lesser extent, of evaluating the results of enlarging "archetypal" shapes (diagonal interfaces etc) [32]. They are nonetheless informative.

Neil Dodgson lists additional instances of such tests in Appendix A of [33].

Given that the above references are not exhaustive, it appears that such quantitative comparisons of image resampling methods will continue to be performed and published regularly. For this reason, they are a worthy topic of investigation.

With the exception of frequency response tests, these quantitative evaluation methods follow similar procedures:

Step 1. Geometrically transform the gold standard image (step omitted in the rotation and translation tests)

Step 2. Geometrically transform the result with the evaluated image resampling filter

Step 3. "Exactly" undo the overall geometrical transformation (omitted in re-enlargement tests)

Step 4. Crop the result and the gold standard image (usually omitted in re-enlargement tests)

Step 5. Compare the result with the gold standard image using one or more image difference metrics.

The quality of image resampling methods can also be evaluated using scores given by human evaluators, that is, using subjective, as opposed to quantitative, measures of quality. Qualitative testing and the related issue of how quantitative testing gives rankings that correlate with subjective evaluation are topics which are not directly addressed by this thesis.

1.3 What is missing?

After reading many articles that compare image resampling methods, one cannot help but be struck by the lack of standardization of testing methods, which makes it impossible to

reproduce and validate the data or compare another method using the same testing procedure.

Test images are often unavailable and badly documented; Code implementing the tested methods is not made available, which not only makes direct testing of novel methods difficult, but also leaves undocumented design specifics like boundary handling; and so on.

The lack of standardization is especially surprising given the methodological commonality exhibited by published testing approaches. In addition, novel methods are generally only compared to a few classical ones, making the nearly invariable top ranking for the authors' own highly questionable.

A more subtle issue with such quantitative testing also becomes apparent, namely the implicit assumption that similarly structured but nonetheless different tests give method rankings that are directly comparable and consequently allow one to draw "universal" conclusions from their results. Leaving aside the issue that very few methods are generally compared in published articles, would different testing parameters—ratios, angles, test images, image difference metrics, colour spaces, image formats etc.—have changed the ranking? In other words, are rankings stable with respect to the particulars of the experimental setup?

1.4 Statement of purpose

The primary aim of this thesis is to foster the following standards in comparative image resampling method evaluation:

- reproducibility
- standardization
- meta-analysis.

The secondary goals of this thesis are to rank a large number of image resampling methods and, more importantly, to measure the impact of various experimental design parameters on results, in particular, on the rankings themselves. This work yielded an unexpected insight, namely that image resampling results are affected in nontrivial ways by colour space choice.

The EXQUIRES (EXtensible QUantitative Image RESampling) test suite, specifically written for this thesis, is the method through which these goals have been achieved. By virtue of being FLOSS (Free/Libre and Open Source Software), EXQUIRES provides a transparently defined test bed for image resampling methods, documented down to the most minute detail thanks to the use of well documented open source software and copyfree image data distributed via the Internet. A large number of image resampling methods are included, and a system is provided for hooking up external resampling programs in order to test them against the built in ones. In other words, EXQUIRES can be used as a standard quantitative test for the accuracy of image resampling methods.

Re-enlargement tests were chosen because they are the most commonly used. However, EXQUIRES would be a good starting point for a test suite that performs rotation or sub-pixel translation tests. In addition, almost trivial modifications would allow EXQUIRES to perform re-enlargement tests in which blur, noise or JPEG compression artifacts are added to the downsampled images, for the purpose of evaluating the reconstruction power of upsamplers with blemished, as opposed to "clean", images. (Blurring essentially consists of applying local averaging to the image in order to remove high frequency detail. JPEG is the most commonly used image compression method.)

Yet, EXQUIRES does not force researchers into a one size fits all comparative test because it is highly customizable. A key feature of EXQUIRES is that it makes it easy to set up and unambiguously document alternate testing procedures so as to allow third parties to reproduce and extend them. That is, far from enforcing a rigid standard, EXQUIRES allows anyone to define his or her own test procedure, automatically documented by way

of EXQUIRES' Python-based scripting.

EXQUIRES also provides a platform for evaluating the impact of test specifics on rankings. For example, it can be used to evaluate the impact of using a different image difference metric. This meta-analysis capability is arguably even more important than documenting and standardizing the attribution of ranks to resampling methods because it allows one to estimate the stability of the ranks with respect to specific changes in the experimental setup.

In order to illustrate EXQUIRES' capabilities, this thesis contains both image resampling method rankings obtained with several carefully chosen experimental setups, and analyses of the correlation of the rankings obtained with varying experimental factors. Although these results are important in and by themselves, the reader should keep in mind that they are but illustrations of EXQUIRES' capabilities. That is: the tool is more important than the results.

EXQUIRES makes it possible to compare resampling methods against many other methods, aggregating results generated using many ratios, downsamplers, image difference metrics, and test images, all this using a transparently defined but nonetheless fully customizable methodology involving clearly written and documented free/libre and open source code. In addition, EXQUIRES facilitates the analysis of the impact of the testing configuration on results, a seldom discussed topic of investigation [11, 16].

2 Previous work

This chapter summarizes some of the many publications that contain a comparison of image resampling methods.

2.1 Re-enlargement (a.k.a. reconstruction) tests

In order to test how an image resampling method performs when enlarging an image, one must have an image to compare against the result. For this reason, it is common practice to first reduce the original image in some way, then re-enlarge this reduced image using various resampling methods. The images can be reduced by several factors that define the ratio of the original width and height to the width and height of its downsampled version, using several methods, and the re-enlargements can be compared to the original images using any number of image difference metrics.

When testing a novel image resampling method, especially one focused on enlarging images, some form of this procedure is typically employed. This section describes several articles that involve performing a re-enlargement test.

Jia-Guu Leu developed a technique for enlarging images that maintains the sharpness of step edges while reducing aliasing and blurring [2]. Testing was performed as follows: First, six test images were reduced by several factors (2, 3, and 4) using neighbour averaging (nearest neighbour). Then the downsampled images were re-enlarged and the results produced by the proposed method were compared against nearest neighbour and bilinear interpolation. The comparison was performed using mean squared error (MSE) and mean

absolute error (MABSE), which appears to be the same as the average absolute error (AAE) used in this thesis, but using an uncommon acronym. It was concluded, based on the quantitative results and from visual inspection, that the proposed method produces better results.

Guoping Qiu developed a new technique for upsampling images that involves creating an image pyramid from common interpolation methods, designing a vector quantizer called the interresolution look-up table (IRLUT), and using this look-up table to obtain an enlarged image from the lower resolution image data [3]. The idea is to use this approach with pre-existing methods of enlarging images in order to improve the quality of the produced images. Four grayscale test images were reduced by a factor of 2 and 4, and these images were re-enlarged using bilinear and bicubic interpolation, Burt and Adelson's Gaussian filtering [34], and a wavelet filter pair, as proposed by Villasenor et al. [35]. The reductions were performed by the same method that performed the re-enlargements, so there is no standard technique for downsampling, and the reductions and enlargements by a factor of 4 were achieved by reducing and enlarging twice by a factor of 2. The results produced by each upsampling method were compared against the IRLUT versions in terms of MSE and signal-to-noise ratio (SNR), which showed that the look-up table approach improves the results of each method.

Morse and Schwartzwald developed a technique for smoothing upsampled images using level-set reconstruction [4]. They low-pass filtered and reduced three test images by a factor of 3 using an unspecified method. Then they re-enlarged the images using nearest neighbour, bilinear, and bicubic interpolation, and applied their smoothing technique to the bicubic result. They used the MSE and mean contour curvature metrics for each individual colour channel to assess the quality of the images produced by each upsampler, and concluded that their technique improves the bicubic results and produces images that are more visually appealing.

Chen et al. proposed a fast method that analyzes the local image structure and partitions it into homogeneous and edge areas [8]. First, they reduced six grayscale images and three

colour images by a factor of 2 and 4 using an unspecified method. They then re-enlarged the downsampled images and compared the results produced by their method against nearest neighbour (called zero-order in their article), bilinear, and bicubic interpolation, as well as the new edge-directed interpolation (NEDI) [36] method. They used the peak signal-to-noise ratio (PSNR) as a metric and determined that their method produces results with an objective quality similar to NEDI. Since they aimed to create a fast method, and their method is more suited to real-time image interpolation than NEDI, achieving similar PSNR values was considered a success.

Dong et al. proposed a non-local iterative back-projection (NLIBP) algorithm for enlarging images [15]. They applied a Gaussian point spread function (which smooths images and blurs them) to four natural test images and downsampled them by a factor of 2 using an unspecified method. Then they re-enlarged the downsampled images and compared the results produced by their method against bicubic interpolation, iterative back projection (IBP) [37], and bilateral filter based IBP (BFIBP) [38] using the PSNR metric. They concluded that NLIBP outperforms all of the other methods.

Chung et al. proposed a fractal-based technique for enlarging images [5]. First, they downsampled seven test images by a factor of 2 using an unspecified method. Then they re-enlarged the downsampled images and compared, using the PSNR metric, the results produced by their method against bicubic and bilinear interpolation, a parametric cubic convolution method proposed by Han and Baek [39], and a conventional fractal method [40]. They concluded that their method is a better reconstructor than the other methods.

Su and Willis developed an image interpolation method based on data-dependent triangulation of lower resolution images [7]. In order to improve performance, they implemented an OpenGL version of their method that allows for real-time image reconstruction. They chose to downsample the test images using a Gaussian filter in order to reduce artifacts. They reduced 20 images by a factor of 2 and compared the results of two versions of their method against nearest neighbour, bilinear, and bicubic interpolation, as well as

NEDI, using the MSE for each individual colour channel. They performed a statistical t-test on the results and determined that no significance could be read into them. Regardless, by visual inspection, they concluded that their method produces good results.

Muresan and Parks developed an adaptively quadratic (AQua) image interpolation method based on optimal recovery and on analysis of the local image behaviour to determine the quadratic signal class [6]. They applied low-pass filtering and decimation by a factor of 2 to five test images, and compared the results produced by six versions of their proposed method against sub-pixel edge localization, as well as edge-directed, bicubic, and Bayesian interpolation, using the PSNR metric. They concluded that AQua interpolation performs better than the other methods, and suggested that it introduces fewer artifacts around edges.

Luming Liang developed a convolution-based image interpolation method [13]. A single test image was reduced by five different factors (1.5, 2, 2.5, 3, and 3.5) using bilinear downsampling, then re-enlarged using Keys and osculatory adaptive rational interpolation, and six versions of the proposed method. The comparison of the upsampled images was performed using the PSNR metric, and it was concluded that the proposed method improves the visual quality of the enlarged images to some extent.

Robidoux et al., including the author of this thesis and his primary supervisor, have developed several image resampling methods, including an exact area method using natural biquadratic histosplines (EANBQH) [14] and an edge-adaptive method called Nohalo [16].

The testing performed for the EANBQH article used the earliest prototype of EXQUIRES. Ten images were downsampled by seven integer factors (2, 3, 4, 5, 6, 7, and 8) and six fractional factors (8/7, 7/6, 6/5, 5/4, 4/3, and 3/2) using box filtering. The image enlargement method EANBQH was compared to box filtering, two bicubic spline implementations using the Scilab SIVP package [41], and 12 methods implemented by the ImageMagick image processing library (Blackman, Catmull-Rom, Hamming, Hann, Hermite, Kaiser, Lagrangian, Lanczos 3, Mitchell-Netravali, Parzen, Triangle, and Welch)

[42]. The RMSE, AAE (average absolute error), MAE (maximum absolute error), and MSSIM (mean structural similarity) metrics were used, and EANBQH was shown to obtain better objective results than the other methods. Although the EANBQH article identifies the tested geometry convention with a slightly different terminology, this convention is consistent with the one used by EXQUIRES (see §4.2).

The second article describes a version of Nohalo that uses bilinear interpolation as a finishing method. (A different version of Nohalo that uses locally bounded bicubic (LBB) interpolation as a finishing method is tested in the present thesis. See §4.8.7.1.) The resampling factors of the EANBQH article were considered, and seven more integer factors (8, 10, 12, 13, 14, 15, and 16) were added. The same upsampling methods as the EANBQH article were tested, except that the ImageMagick Triangle method was replaced by a bilinear interpolation implementation from the image processing library VIPS (Virtual Image Processing System) [43], an additional SIVP bicubic spline implementation was added, and two more ImageMagick methods (Bessel and Gaussian) were tested. However, the image geometry convention was switched to the other most common one, as discussed in §4.2. Nohalo ranked highest for all tests in terms of RMSE and near the top for the other metrics. It was concluded that Nohalo is accurate and produces pleasant enlargements of natural images.

Miravet and Rodriguez proposed a method for upsampling images based on neural processing of local image representations [9]. First, two sequences of 25 grayscale satellite images were reduced by a factor of 2 using an unspecified method. They re-enlarged the downsampled images and compared the results produced by the proposed method against their previously proposed method MLP-PNN (Multi-Layer Perceptron – Probabilistic Neural Network) and nearest neighbour interpolation in terms of root mean squared error (RMSE), both with and without the addition of Gaussian noise. In both tests, their method achieved the lowest RMSE values. Coupled with their own visual inspection, they concluded that the proposed method improves noise rejection and image definition.

Meijering et al. proposed a convolution-based method of image reconstruction using a class of Sinc-approximating symmetrical piecewise nth-order polynomial kernels [1]. They used 16 test images and computed the MSE for enlargements produced with linear interpolation, and cubic, quintic, and septic convolution. The original images were low-pass filtered and downsampled with an unspecified method by a ratio of 4 before upsampling with the various interpolation methods. In addition to the re-enlargement tests, they performed subpixel translation and rotation tests, and used MSE to assess the quality of the resampling methods. Septic convolution produced the lowest error values in all cases, but the authors cautioned that the choice of method should be application-sensitive due to the computational cost.

Based on an article by Roussos and Maragos [44], Pascal Getreuer proposed an image interpolation method that makes use of tensor-driven diffusion [18]. While several images are discussed in the article, error data is only shown for a single image, which is reduced by applying a Gaussian convolution and downsampling by a factor of 4. The proposed method is compared against bicubic interpolation, a fractal zooming method from the Genuine Fractals software package (now called Perfect Resize) [45], Fourier zero-padding with deconvolution, TV minimization [46], and contour stencils [47]. The PSNR and MSSIM metrics were used to compare the resulting re-enlargements with the original, and the proposed method achieved the best results for both metrics.

The article by Amanatiadis and Andreadis presents an overview of evaluation methods for image interpolation methods and describes several different image transformation procedures for obtaining images to compare with the original image [48]. In addition to the procedure used in this thesis, they describe upsampling an image before downsampling and using successive rotations as alternative approaches. As a demonstration, they performed some simple tests with nearest neighbour, bilinear, and bicubic interpolation, and showed with PSNR and RMSE that bicubic interpolation produces the best results, quantitatively speaking.

Dumic et al. performed a re-enlargement test with the goal of determining which factors have the greatest influence on image quality [11]. The factors they considered are the downsampling methods, error metrics, and contents of the test images. Without performing an in-depth data analysis, they concluded that the choice of downsampler has the most impact on the quality of the re-enlarged images. Further analysis of their results is presented in §3.2.

2.2 Rotation tests

Rotation tests involve repeatedly rotating an image with the same resampling method until the total rotation angle reaches a multiple of 90°, then undoing the rotation and cropping the result in order to reduce the impact of the abyss (the region outside of the image) when comparing against the original image. This section describes several articles that involve rotation tests.

Unser et al. developed a convolution-based method optimized for image rotation [19]. They used three test images and performed 16 successive rotations of 22.5°. Using RMSE, they compared the results of several version of their method against nearest neighbour and bilinear interpolation, Keys cubic, cubic spline, and several 2-pass and 3-pass separable filters. They concluded, both in terms of speed and quality, that the higher-order versions of their algorithm are superior to all the standard high-quality methods they were familiar with.

Larkin et al. developed a fast Fourier method optimized for image rotation [20]. They used a small grayscale image of the Mona Lisa, applied zero padding in the real and Fourier space, then rotated the image twice by 45°. Rather than using an error metric, they simply subtracted the resulting image from a version of the original that was transposed by 90°. In the unpadded region, the difference was found to be exactly zero.

In addition to the re-enlargement test described in the previous section, Meijering et al.

also performed a rotation test on their convolution-based method [1], where the test images were rotated by 15° by each method, then rotated back to their original position using the same method. The ranks for each method are the same as those for the re-enlargement test. They also used rotation tests in their comparison of Sinc-approximating kernels [21], spline interpolation and other convolution-based methods [22], and a more comprehensive comparison of convolution-based methods [25], all in terms of interpolating medical images. In all, 126 different kernels were evaluated, including a variety of piecewise polynomial kernels (nearest neighbour, bilinear, Lagrange, generalized convolution, and B-spline), and several windowed Sinc kernels (Bartlett, Blackman, Blackman-Harris, Bohman, Cosine, Gaussian, Hamming, Hann, Kaiser, Lanczos, Rectangular, and Welch). For these articles, they used two different sets of rotation angles: the sequence of 0.7°, 3.2°, 6.5°, 9.3°, 12.1°, 15.2°, 18.4°, 21.3°, 23.7°, 26.6°, 29.8°, 32.9°, 35.7°, 38.5°, 41.8°, and 44.3° [21, 25], and 16 rotations of 22.5° each [22]. They used a collection of metrics to compare the results to the original images: MSE [1], RMSE [21, 22, 25], largest absolute error (LAE) [22], which is referred to as maximum absolute error (MAE) in this thesis, and root peak square error (RPSE) [25], which is not referred to by any of the other articles mentioned in this chapter. They concluded that spline interpolation is the preferred method, with cubic convolution resulting in a significant reduction in interpolation errors when compared against linear interpolations. While better results are obtained with higher-degree spline methods, they are computationally expensive.

Thévenaz et al. authored two articles on image interpolation and resampling: one making use of a circular pattern and a human subject as test images [23], and the other making use of both of these as well as a CT scan [24]. Both articles involve rotating the test images 15 times by 24° and using SNR to compare the results against the original images. The tested resampling methods are nearest neighbour and bilinear interpolation, Keys cubic [49], methods by Alan P. Schaum [50], Ismail German [51], Neil A. Dodgson [30], Meijering et al. [1], and Blu et al. [52], as well as several windowed Sinc (Bartlett, Dirich-

let, Hamming, and Hann) and spline methods. In the first article, the best SNR results were obtained by the sextic version of the optimal maximal order and minimum support (O-MOMS) method [52]. However, the authors conlcuded that the lack of regularity of O-MOMS methods makes them less suitable than B-splines for edge detection and other problems involving the computation of integrals. In the second article, the authors advocate the use of B-splines once again, and they go on to suggest that the MOMS class of functions—such as B-splines, Schaum [50], and O-MOMS [52]—provide the best compromise between speed and quality.

Nehab and Hoppe are responsible for a comparison of several resampling methods [26]. They performed 31 successive rotations by an angle of $2\pi/31$ about the centre of a set of four test images, taken from the Kodak Lossless True Color Image Suite [53], and an artificial image created by a radial function. They used the MSSIM index to compare the results produced by nearest neighbour and linear interpolation, as well as Catmull-Rom, Hamming, Lanczos, Mitchell-Netravali, and methods by Alan P. Schaum [50], Blu et al. [54, 55], Condat et al. [56], and Dalai et al. [57]. The top three methods were O-MOMS with degree 5 [55], B-spline with degree 5, and O-MOMS with degree 3 [55].

2.3 Sub-pixel translation tests

Sub-pixel translation tests involve repeatedly translating an image, with a fixed resampling method, so that the total translation is an exact multiple of the inter-pixel distance (both horizontally and vertically), then undoing the translation and cropping before comparing the result with the original image. This section describes several articles that involve performing a sub-pixel translation test.

In the previous sections, several articles by Meijering et al. were mentioned. In addition to the aforementioned tests, all of these articles also involve sub-pixel translation tests. They performed a translation by $(0.4, 0.7)$ pixels [1], as well as a set of horizontal transla-

tions by the following distances: 0.01, 0.04, 0.07, 0.11, 0.15, 0.18, 0.21, 0.24, 0.26, 0.29, 0.32, 0.35, 0.39, 0.43, 0.46, and 0.49 (totalling 4.0 pixels) [21, 22, 25]. In all cases, the same images, resamplers, and metrics were used as in the previous tests. Furthermore, regardless of the type of test being performed, the authors concluded that spline interpolation provided the best trade-off between speed and accuracy.

In addition to the rotation test described in the previous section, Nehab and Hoppe also performed a sub-pixel translation test [26]. They performed 32 successive translations—cycling over the set $\{(0.5, 0.5), (0.5, -0.5), (-0.5, -0.5), (-0.5, 0.5)\}$—on the four Kodak test images and the artificial radial image. Once again, MSSIM was used to compare the results obtained using the same resampling methods as the rotation test. Interestingly, the top methods for this test are the same as for the rotation test.

Ellis Freedman studied the impact of resampling on medical imagery by translating several images $15/32$ of a pixel to the left, then back $15/32$ of a pixel to the right and subtracting the result from the original [27]. He compared the constant modulation transfer function (CMTF) interpolation method against a standard cubic convolution. No metrics are used, but by visual inspection alone, it is clear that CMTF introduces less error.

2.4 Frequency response tests

James F. Blinn provides an overview of the Fourier transform and convolution theorem for image processing, as well as a discussion of frequency response issues [28]. Frequency response tests involve measuring the frequency response of the method in question, meaning that they do not directly measure the performance of an image resampling method, since they do not involve "real" images. Regardless, they are still informative. This section describes several articles that involve frequency response tests. Because they are completely different from the other tests described in this chapter and this thesis, they will not be discussed in detail.

Ken Turkowski describes several popular filters—box, tent, Gaussian, and Sinc—and shows the frequency response of several versions of the filters [29]. In particular, he shows the frequency response for several decimation ratios, and provides filter coefficients for various phases and decimation ratios. Furthermore, he analyzes these filters in terms of their stopband and passband response.

Neil A. Dodgson has written a highly comprehensive technical report on image resampling [33] and has proposed a quadratic image interpolation method [30]. By comparing the stopband and passband response of several methods, it was determined that the proposed method has a frequency response better than linear interpolation, but inferior to Catmull-Rom.

In addition to re-enlargement, rotation, and sub-pixel translation tests, Meijering et al. also performed a frequency response test [1]. They compared the kernels of the four lowest-order interpolators to their corresponding ideal interpolation kernels. Using the total square error (distance) of the spectra with respect to the spectrum of the Sinc function, they concluded that cubic, quintic, and septic convolution are, respectively, a 33.9%, 36.4%, and 39.2% improvement over linear interpolation. Furthermore, quintic and septic convolution are only, respectively, a 3.8% and 8.0% improvement over cubic convolution.

Amir Said proposed several kernels for image interpolation and resizing and derived parameters for approximating popular interpolation kernels, namely Lanczos, Blackman-Harris, Cubic B-spline, and Mitchell-Netravali [31]. The differences between these approximations and the original kernels are established by measuring the frequency response and computing the difference between several images resized with the kernels by way of the PSNR metric.

Chantal Racette's Master's thesis includes chapters discussing the frequency response of several linear filters, as well as relative minimax polynomial approximations of the 2-lobe and 3-lobe versions of Lanczos [32]. Also included is Scilab code that computes the frequency response of decimation by a factor of n performed with various filters.

2.5 Subjective tests

Since viewer experience is the true test of an image resampling method's performance, one might opt to supplement the aforementioned objective tests with subjective testing, in which a group of people (possibly image processing experts) evaluate methods based on how visually appealing their output is. This section describes several articles that involve subjective tests.

Mitchell and Netravali developed a family of cubic filters defined by two parameters: B and C (equivalent to the α parameter of the Keys family of filters) [58]. They assembled a group of image processing experts to help classify the types of artifacts that are found in images enlarged by this family of filters and concluded that setting B and C to $1/3$ gives very good results results. This cubic filter is now known as the Mitchell-Netravali filter.

Wang et al. developed the Structural SIMilarity (SSIM) index, which is an image metric based on the idea that the Human Visual System (HVS) is tuned to extract structural information from a scene [59]. They conducted a subjective test using a collection of JPEG and JPEG2000 images. These images were viewed by a small group of people who were asked to assign a quality rating to each image. Then they compared their metric against PSNR, the Sarnoff model, and the Universal Quality Index (UQI) [60]—an earlier metric based on the concept of structural similarity—using several non-linear and variance-weighted regressions to map between the subjective data and the objective metric data. Most notably, they used the Spearman rank-order correlation coefficient as a measure of prediction monotonicity.

More recently, Xu et al. published a paper that deals with the use of online crowdsourcing for subjective image quality assessment [61]. This technique makes use of HodgeRank, which is a framework for decomposing paired comparison data. They used several images from the IVC [62] and LIVE [63] image databases, which contain reference images and various distorted versions of those images. For each reference image they used,

they also selected 15 distorted versions of the image. Subjects were presented with a series of image pairs, and were asked to decide which looked better. It was concluded that online HodgeRank is an effective way of studying large-scale subjective image quality assessment on the Internet.

2.6 Discussion

In each of the articles described in this chapter, various tests are performed. Generally, these tests are not described in sufficient detail to be reproducible. In addition, there is a lack of standardization. Image metrics are used rather haphazardly (for example, not all articles use the same definition of PSNR), which makes comparing results difficult, and none of the articles explicitly state which colour space is used when performing the resampling steps. Furthermore, there is never any mention of the image geometry convention that is used.

Another shortcoming of the tests performed in these articles is the tendency to compare proposed, novel, image resampling methods against low quality interpolation methods, such as nearest neighbour, bilinear, and bicubic. Furthermore, the authors rarely identify which type of bicubic interpolation they are comparing against.

3 Methodology

The quantitative methodology used by EXQUIRES to rank image resampling methods is both standard and formally simple: Reduce the dimensions of carefully chosen gold standard images (images considered to be flawless), re-enlarge with the filters that are to be ranked, measure the deviation between the re-enlargement and the original with some image difference metric, and rank in decreasing order of deviation. The specifics of the ranking experiment are discussed in the next chapter and in the EXQUIRES user manual and source code, reproduced in Appendices.

Besides its emphasis on detailed documentation and open distribution, EXQUIRES improves on earlier quantitative image resampling evaluation testing with its use of the Spearman correlation to measure the agreement and disagreement between rankings obtained with different experimental setups.

The Spearman rank correlation [64] is defined in the first section of this chapter. In the second section, we use the small data set found in a published study of the impact of various factors on image resampling data to illustrate its use to analyze and summarize the results of an image reconstruction study.

3.1 Correlation between rankings of the same items under different conditions

The Spearman rank correlation specifically measures whether two alternate rankings of the same set of items agree, disagree, or are effectively "disconnected" [65]. In the present

thesis, the Spearman rank correlation is used to study the agreement between rankings derived under different conditions.

Generally, a correlation coefficient is a quantitative measure of the agreement (or disagreement) between two related datasets, usually expressed by a real number in the interval $[-1, 1]$. A positive value of the correlation coefficient indicates that ranks within the first ranking tend to increase as the ranks increase within the second ranking (and vice versa), whereas a negative value of the correlation coefficient indicate that ranks within the first ranking tend to decrease as the ranks, within the second ranking, increase. For example, a rank correlation of 1 indicates a perfect positive correlation, meaning that the ranks obtained by items are exactly the same within the two rankings, and a rank correlation of -1 indicates perfect negative correlation, meaning that the first ranking lists items from last to first within the second ranking. A correlation of 0 suggests that there is no correlation, meaning that the rank of an item within the first ranking generally gives no information on the rank within the second ranking. Thus, a near zero value, positive or negative, of the rank correlation, indicates that the ranks within the two rankings are weakly correlated. The strength of the correlation is greater for values further away from zero.

The Spearman rank correlation is used by Z. Wang et al. to measure the correlation between the mean SSIM (Structural SIMilarity) index—an error metric they developed—and similarity scores derived from human subject perceptual studies [66]. The EXQUIRES test suite makes use of several error metrics which are completely different from one another. Regardless, it is often useful to aggregate the errors from these widely different metrics to give an overall assessment of each upsampler's ability to reconstruct the gold standard images. Even when the errors are not aggregated, ranks allow one to compare results more easily. As a result, the raw error data is first converted to ranks.

In this thesis, resampling filters are ranked under different conditions: using different test images, different downsamplers, different resampling ratios, different image difference metrics etc. Rank correlations provide a quantitative measure of whether rankings obtained

under different conditions agree. Consequently, rank correlations can be used to assess the sensitivity of the rankings with respect to changes in the experimental setup.

It is expedient to summarize this information with correlation matrices. In such a matrix, every row represents one possibility among a set of conditions used to derive rankings. For example, every row may correspond to a test image. The columns represent the same conditions, in the same order. Then, every entry of the correlation matrix shows the rank correlation between the ranking obtained under the condition that corresponds to the row, and the ranking obtained under the condition that corresponds to the column. Since a ranking obtained with a certain condition (with a certain test image, for example) is perfectly correlated with the ranking obtained with the same condition, correlation matrices have a diagonal of ones. Since the correlation between a first ranking and a second ranking is the same as the correlation between the second ranking and the first, the matrix is symmetric about its main diagonal.

Matrix entries near 1 indicate that the corresponding row and column conditions give rise to rankings that are in general agreement. Otherwise—when the corresponding matrix entry is close to zero or negative—the rankings obtained under different conditions disagree. A large number of matrix entries that are far from 1 consequently indicates a high sensitivity of the rankings with respect to changes in the experimental condition under consideration.

3.1.1 Computation of the Spearman rank correlation

Given the results of two different measures of quality e_i and f_i applied to the same set of upsamplers (where i is a key that identifies each upsampler), the Spearman rank correlation is computed as follows.

First, the upsamplers are independently ranked from best to worst with respect to each of the two quality measures. Fractional ranks are used: ties are resolved by assigning to

all tied upsamplers the average rank they would obtain if the ties were resolved arbitrarily without affecting the ranks of the other upsamplers. This is sometimes called "1 2.5 2.5 4" ranking [67]. Fractional ranking guarantees that the average of all ranks is $(N+1)/2$ (where N is the number of upsamplers considered) regardless of whether or not there are ties.

Let r_i (resp. s_i), for $i = 1, \ldots, N$, be the fractional rank of the ith upsampler with respect to the e_i quality measurements (resp. f_i). Then, the Spearman rank correlation is

$$\rho = \frac{\sum_i \left(r_i - \frac{N+1}{2}\right)\left(s_i - \frac{N+1}{2}\right)}{\sqrt{\left(\sum_i \left(r_i - \frac{N+1}{2}\right)^2\right)\left(\sum_i \left(s_i - \frac{N+1}{2}\right)^2\right)}}.$$

In other words, the Spearman rank correlation is the Pearson's correlation (product-moment coefficient) of the associated fractional ranks [68].

In this thesis, e_i and f_i are often aggregated error measures or mean SSIM indexes, or even averaged fractional ranks when several different quality measures are aggregated.

3.1.2 Kendall's rank correlation

Z. Wang also uses Kendall rank correlation [69] when evaluating the performance of SSIM [66]. The Kendall rank correlation is a better choice for measuring rank correlation strength: As noted at the StatsDirect website, while Spearman's rank correlation is a satisfactory tool for testing the null hypothesis of independence between two variables, Kendall's rank correlation provides a better measure of the strength of the dependence [70, 71]. For this reason, a welcome addition to EXQUIRES would be a tool that computes Kendall's rank correlation. Its computation is, unfortunately, more complex.

3.2 Analysis of the Dumic, Grgic & Grgic data

Hidden Influences on Image Quality when Comparing Interpolation Methods, by Emil Dumic, Sonja Grgic and Mislav Grgic [11], contains just enough data to compute somewhat meaningful Spearman rank correlations. In this chapter, this data is used to illustrate the analysis process, namely the computation of overall ranks, and the computation of Spearman rank correlations.

The three upsamplers compared by Dumic et al. are

- bilinear
- global cubic spline interpolation
- TVC10_18 wavelet filtering.

They are compared using four error metrics,

- SNR (Signal to Noise Ratio), in which the ℓ_2-error is normalized by the ℓ_2-norm (RMS norm) of the original image
- PSNR (Peak Signal to Noise Ratio, the peak being the maximum possible pixel value), which we convert to normalized ℓ_2 (RMSE) error for our aggregate rank computations computations
- PQS (Picture Quality Scale)
- SSIM (Structural SIMilarity index)

three resampling ratios,

- 2
- 4
- 8

three downsampling methods,

- bilinear
- global cubic spline interpolation
- TVC10_18 wavelet filtering

and four 2048×2048 test images,

- Andromeda
- Train
- Tree
- Roof.

Dumic et al. do not specify what colour space(s) the images belong to. In particular, they do not state whether the pixel values can be understood as linear light, and whether MSSIM is computed using luminance or luma (which roughly means luminance, a measure of instensity, based on a nonphysical colour space [72, 73]).

3.2.1 Overall upsampler ranks

To compute overall upsampler ranks, we first aggregate, for each upsampler, the errors obtained with each quality metric across all ratios, downsamplers, and images. We then aggregate the ranks obtained with each quality metric to obtain overall ranks representative of all metrics, ratios, downsamplers and images.

Aggregating SNR errors by simple averaging (for simplicity), we obtain

$$\begin{aligned}
&\mathrm{SNR}_{\text{bilinear}} \\
&= \frac{1}{36}\left(\begin{array}{l} 17.13+17.25+17.43+12.53+12.38+12.85+7.87+7.8+8.61+ \\ 28.16+28.94+29.84+22.61+22.54+23.57+17.14+17.00+18.21+ \\ 19.80+20.20+20.42+15.77+15.56+16.24+13.19+13.02+13.96+ \\ 28.95+29.65+30.52+23.55+23.50+24.54+17.65+17.51+18.73 \end{array}\right) \\
&= 19.01,
\end{aligned}$$

$\mathrm{SNR}_{\text{spline}} = 19.66,$

$\mathrm{SNR}_{\text{wavelet}} = 19.68.$

17.13, 17.25, ..., 18.73 are the thirty six SNRs obtained by the re-enlargements performed with the bilinear upsampler. The SNRs for the spline and wavelet upsamplers are obtained likewise from the corresponding groups of thirty six SNRs.

Although, as discussed in the article, the average SNRs for spline and wavelet upsampling are close, we ignore this near tie and assign a rank of 3 to bilinear, 2 to spline interpolation, and 1 for wavelet filtering. (Higher SNRs are better.)

The aggregate normalized ℓ_2 errors, derived from the article's reported PSNRs by taking 100 times the square root of the average of $10^{-\frac{\mathrm{PSNR}}{10}}$, are

$$\mathrm{RMSE}_{\text{bilinear}} = 7.678,\ \mathrm{RMSE}_{\text{spline}} = 7.648,\ \text{and}\ \mathrm{RMSE}_{\text{wavelet}} = 7.567,$$

leading to the same ranks since lower ℓ_2 errors (which correspond to higher PSNRs) are better.

The aggregate PQS errors, obtained by simple averaging of the individual PQS errors, are

$$\mathrm{PQS}_{\text{bilinear}} = 1.255,\ \mathrm{PQS}_{\text{spline}} = 1.849,\ \text{and}\ \mathrm{PQS}_{\text{wavelet}} = 1.873,$$

leading to the same ranks since higher PQSs are better.

The aggregate SSIM errors, obtained by simple averaging, are

$$\mathrm{SSIM}_{\mathrm{bilinear}} = .8269,\ \mathrm{SSIM}_{\mathrm{spline}} = .8273, \text{ and } \mathrm{SSIM}_{\mathrm{wavelet}} = .8296,$$

leading, again, to the same ranks since higher SSIMs are better.

Since the aggregate ranks are exactly the same across all metrics, the overall ranking of the tested upsampling methods is

overall rank 1: TVC10_18 wavelet filtering

overall rank 2: global cubic spline interpolation

overall rank 3: bilinear

in agreement with the discussion given by Dumic et al.

3.2.2 Impact of the error metric

The above makes clear that when aggregating data over all ratios, downsamplers and images, upsampler ranks are the same irregardless of the considered metric. For this reason, the data presented by Dumic et al. suggests that, when ranking upsampling methods as reconstructors in the context of a re-enlargement test, it makes no difference which metric is used, since aggregate ranks obtained with SNR, PSNR (or, equivalently, MSE or ℓ_2), PQS and SSIM are perfectly correlated, and the metric rank correlation matrix P between the various metrics is a perfect

$$\mathrm{P}_{\mathrm{metric}} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}.$$

Dumic et al. state that "Usage of only PSNR or SNR as image quality measures can lead to wrong conclusions when comparing interpolation methods for different test images." This

statement is not borne out their data, since our analysis shows that the *aggregate* rankings derived from the data of Dumic et al. are the same for all four considered image difference metrics. (We will see that our own data suggests that the choice of image difference metric has a significant impact on rankings. In other words, our own data validates the above statement of Dumic et al.)

3.2.3 Impact of the resampling ratio

In order to evaluate whether the resampling ratio influences upsampler rank, we aggregate an upsampler's error data across all metrics, downsamplers and images, keeping results corresponding to different ratios separate.

Because they are not directly comparable, we aggregate results obtained with different quality metrics by averaging the corresponding ranks. We first aggregate the data corresponding to each ratio and metric across downsamplers and test images:

$$\mathrm{SNR}_{\mathrm{bilinear},2} = \frac{1}{12}\left(\begin{array}{l} 17.13 + 17.25 + 17.43 + 28.16 + 28.94 + 29.84 + \\ 19.80 + 20.20 + 20.42 + 28.95 + 29.65 + 30.52 \end{array}\right) = 24.02,$$

$$\mathrm{SNR}_{\mathrm{spline},2} = 25.45,$$

$$\mathrm{SNR}_{\mathrm{wavelet},2} = 25.39.$$

Averaged ranks may, or may not, be ranks themselves, so we convert these average errors to fractional ranks:

$$\mathrm{rank}_{\mathrm{bilinear},2,\mathrm{SNR}} = 3,\ \mathrm{rank}_{\mathrm{spline},2,\mathrm{SNR}} = 1,\ \mathrm{rank}_{\mathrm{wavelet},2,\mathrm{SNR}} = 2.$$

(Fractional ranks are most appropriate because we eventually compute Spearman rank correlations.) Note that these partial ranks are not the same as the overall ranks shown at the end of §3.2.1.

Likewise,

$$\mathrm{SNR}_{\mathrm{bilinear},4} = 18.80,\ \mathrm{SNR}_{\mathrm{spline},4} = 19.37,\ \mathrm{SNR}_{\mathrm{wavelet},4} = 19.40,$$

so that the ratio 4 partial SNR ranks are

$$\text{rank}_{\text{bilinear,4,SNR}} = 3,\ \text{rank}_{\text{spline,4,SNR}} = 2,\ \text{rank}_{\text{wavelet,4,SNR}} = 1,$$

and

$$\text{SNR}_{\text{bilinear,8}} = 14.22,\ \text{SNR}_{\text{spline,8}} = 14.15,\ \text{SNR}_{\text{wavelet,8}} = 14.26,$$

so that the ratio 8 partial SNR ranks

$$\text{rank}_{\text{bilinear,8,SNR}} = 2,\ \text{rank}_{\text{spline,8,SNR}} = 3,\ \text{rank}_{\text{wavelet,8,SNR}} = 1.$$

We now proceed to aggregating the data generated with the other metrics, and converting them to ranks.

First, ℓ_2 (by way of PSNR data from Dumic et al.). Given that

$$\text{RMSE}_{\text{bilinear,2}} = 4.222,\ \text{RMSE}_{\text{spline,2}} = 3.753,\ \text{RMSE}_{\text{wavelet,2}} = 3.734,$$

the ℓ_2 ranks with ratio 2 are

$$\text{rank}_{\text{bilinear,2,RMSE}} = 3,\ \text{rank}_{\text{spline,2,RMSE}} = 2,\ \text{rank}_{\text{wavelet,2,RMSE}} = 1.$$

Similarly,

$$\text{RMSE}_{\text{bilinear,4}} = 7.074,\ \text{RMSE}_{\text{spline,4}} = 6.883,\ \text{RMSE}_{\text{wavelet,4}} = 6.846,$$

so that

$$\text{rank}_{\text{bilinear,4,RMSE}} = 3,\ \text{rank}_{\text{spline,4,RMSE}} = 2,\ \text{rank}_{\text{wavelet,4,RMSE}} = 1,$$

and

$$\text{RMSE}_{\text{bilinear,8}} = 10.44,\ \text{RMSE}_{\text{spline,8}} = 10.68,\ \text{RMSE}_{\text{wavelet,8}} = 10.53,$$

so that

$$\text{rank}_{\text{bilinear,8,RMSE}} = 1,\ \text{rank}_{\text{spline,8,RMSE}} = 3,\ \text{rank}_{\text{wavelet,8,RMSE}} = 2.$$

We now aggregate PQS errors. Given that

$$\mathrm{PQS}_{\mathrm{bilinear},2} = 4.333,\ \mathrm{PQS}_{\mathrm{spline},2} = 4.690,\ \mathrm{PQS}_{\mathrm{wavelet},2} = 4.747,$$

the PQS ranks with ratio 2 are

$$\mathrm{rank}_{\mathrm{bilinear},2,\mathrm{PQS}} = 3,\ \mathrm{rank}_{\mathrm{spline},2,\mathrm{PQS}} = 2,\ \mathrm{rank}_{\mathrm{wavelet},2,\mathrm{PQS}} = 1.$$

Similarly,

$$\mathrm{PQS}_{\mathrm{bilinear},4} = 2.600,\ \mathrm{PQS}_{\mathrm{spline},4} = 3.445,\ \mathrm{PQS}_{\mathrm{wavelet},4} = 3.436,$$

so that

$$\mathrm{rank}_{\mathrm{bilinear},4,\mathrm{PQS}} = 3,\ \mathrm{rank}_{\mathrm{spline},4,\mathrm{PQS}} = 1,\ \mathrm{rank}_{\mathrm{wavelet},4,\mathrm{PQS}} = 2,$$

and

$$\mathrm{PQS}_{\mathrm{bilinear},8} = -3.168,\ \mathrm{PQS}_{\mathrm{spline},8} = -2.587,\ \mathrm{PQS}_{\mathrm{wavelet},8} = -2.563,$$

so that

$$\mathrm{rank}_{\mathrm{bilinear},8,\mathrm{PQS}} = 3,\ \mathrm{rank}_{\mathrm{spline},8,\mathrm{PQS}} = 2,\ \mathrm{rank}_{\mathrm{wavelet},8,\mathrm{PQS}} = 1.$$

Finally, we aggregate SSIM errors. Given that

$$\mathrm{SSIM}_{\mathrm{bilinear},2} = .9363,\ \mathrm{SSIM}_{\mathrm{spline},2} = .9493,\ \mathrm{SSIM}_{\mathrm{wavelet},2} = .9492,$$

the SSIM ranks with ratio 2 are

$$\mathrm{rank}_{\mathrm{bilinear},2,\mathrm{SSIM}} = 3,\ \mathrm{rank}_{\mathrm{spline},2,\mathrm{SSIM}} = 1,\ \mathrm{rank}_{\mathrm{wavelet},2,\mathrm{SSIM}} = 2.$$

Also,

$$\mathrm{SSIM}_{\mathrm{bilinear},4} = .8329,\ \mathrm{SSIM}_{\mathrm{spline},4} = .8353,\ \mathrm{SSIM}_{\mathrm{wavelet},4} = .8375,$$

so that

$$\text{rank}_{\text{bilinear},4,\text{SSIM}} = 3,\ \text{rank}_{\text{spline},4,\text{SSIM}} = 2,\ \text{rank}_{\text{wavelet},4,\text{SSIM}} = 1,$$

and

$$\text{SSIM}_{\text{bilinear},8} = .7117,\ \text{SSIM}_{\text{spline},8} = .6974,\ \text{SSIM}_{\text{wavelet},8} = .7023,$$

so that

$$\text{rank}_{\text{bilinear},8,\text{SSIM}} = 1,\ \text{rank}_{\text{spline},8,\text{SSIM}} = 3,\ \text{rank}_{\text{wavelet},8,\text{SSIM}} = 2.$$

The second major step is the aggregation of results across metrics by averaging the corresponding ranks:

$$\begin{aligned}
\overline{\text{rank}}_{\text{bilinear},2} &= \frac{1}{4}(3+3+3+3) = 3, \\
\overline{\text{rank}}_{\text{spline},2} &= \frac{1}{4}(1+2+2+1) = 1.5, \\
\overline{\text{rank}}_{\text{wavelet},2} &= \frac{1}{4}(2+1+1+2) = 1.5.
\end{aligned}$$

Since spline and wavelet are tied, this gives the following fractional ranks:

$$\text{rank}_{\text{bilinear},2} = 3,\ \text{rank}_{\text{spline},2} = 1.5,\ \text{rank}_{\text{wavelet},2} = 1.5.$$

Proceeding likewise with the ranks obtained by ratio 4 resampling:

$$\begin{aligned}
\overline{\text{rank}}_{\text{bilinear},4} &= \frac{1}{4}(3+3+3+3) = 3, \\
\overline{\text{rank}}_{\text{spline},4} &= \frac{1}{4}(2+2+1+2) = 1.75, \\
\overline{\text{rank}}_{\text{wavelet},4} &= \frac{1}{4}(1+1+2+1) = 1.25,
\end{aligned}$$

so that

$$\text{rank}_{\text{bilinear},4} = 3,\ \text{rank}_{\text{spline},4} = 2,\ \text{rank}_{\text{wavelet},4} = 1.$$

With ratio 8,

$$\overline{\text{rank}}_{\text{bilinear},8} = \frac{1}{4}(2+1+3+1) = 1.75,$$
$$\overline{\text{rank}}_{\text{spline},8} = \frac{1}{4}(3+3+2+3) = 2.75,$$
$$\overline{\text{rank}}_{\text{wavelet},8} = \frac{1}{4}(1+2+1+2) = 1.5,$$

so that

$$\text{rank}_{\text{bilinear},8} = 2,\ \text{rank}_{\text{spline},8} = 3,\ \text{rank}_{\text{wavelet},8} = 1.$$

We are finally ready to compute Spearman rank correlations between the various resampling ratios.

$$\rho_{2,4} = \frac{(3-2)(3-2)+(1.5-2)(2-2)+(1.5-2)(1-2)}{\sqrt{((3-2)^2+(1.5-2)^2+(1.5-2)^2)\,((3-2)^2+(2-2)^2+(1-2)^2)}}$$
$$= .866,$$
$$\rho_{2,8} = 0,$$
$$\rho_{4,8} = .5,$$

yielding a ratio rank correlation matrix equal to

$$\text{P}_{\text{ratio}} = \begin{pmatrix} 1 & .866 & 0 \\ .866 & 1 & .5 \\ 0 & .5 & 1 \end{pmatrix}.$$

(Comment: When there is no tie, there is a shortcut formula for Spearman rank correlations. We do not use the shortcut to illustrate the general case.) The choice of resampling ratio consequently appears to have a very strong impact on upsampler method ranking, so much so that the rankings obtained with the smallest and largest ratios are perfectly uncorrelated. That is: Based on their data, resampling ratio is a strong "hidden influence" which appears to have escaped the notice of Dumic et al. (Our own data does not suggest that resampling ratio has that large an impact.)

3.2.4 Impact of the downsampler

This time, we aggregate, keeping results obtained with different downsamplers separate.

$$\mathrm{SNR}_{\mathrm{bilinear,bilinear}} = \frac{1}{12}\left(\begin{array}{l} 17.13 + 12.53 + 7.87 + 28.16 + 22.61 + 17.14 + \\ 19.80 + 15.77 + 13.19 + 28.95 + 23.55 + 17.65 \end{array}\right) = 18.70,$$

$$\mathrm{SNR}_{\mathrm{spline,bilinear}} = 19.43,$$

$$\mathrm{SNR}_{\mathrm{wavelet,bilinear}} = 19.40,$$

so that

$$\mathrm{rank}_{\mathrm{bilinear,bilinear,SNR}} = 3,\ \mathrm{rank}_{\mathrm{spline,bilinear,SNR}} = 1,\ \mathrm{rank}_{\mathrm{wavelet,bilinear,SNR}} = 2.$$

Proceeding as before, we obtain the Spearman rank correlations between the various downsamplers,

$$\rho_{\mathrm{bilinear,spline}} = \frac{(3-2)(2-2) + (2-2)(3-2) + (1-2)(1-2)}{\sqrt{\left((3-2)^2 + (2-2)^2 + (1-2)^2\right)\left((2-2)^2 + (3-2)^2 + (1-2)^2\right)}}$$

$$= .5 = \rho_{\mathrm{spline,wavelet}},$$

$$\rho_{\mathrm{bilinear,wavelet}} = 1.$$

yielding a downsampler rank correlation matrix equal to

$$\mathrm{P}_{\mathrm{downsampler}} = \begin{pmatrix} 1 & .5 & 1 \\ .5 & 1 & .5 \\ 1 & .5 & 1 \end{pmatrix}.$$

This suggests significant dependence of the ranking on downsampler choice (in accordance with the conclusions of Dumic et al.), even though the overall rankings obtained with bilinear downsampling and wavelet downsampling are perfectly correlated. The outlier, cubic B-spline smoothing used as a downsampler, is quite blurry compared to the other two downsampling methods, leading one to posit that the blurriness of the downsampler may be an

important factor. (Given that all the downsamplers considered by Dumic et al. are decent low pass filters, and that their data is sparse, this pretty much agrees with the conclusion we will draw from our own data.)

3.2.5 Impact of the test image

This time, we aggregate keeping results obtained with different test images separate.

$$\begin{aligned}&\mathrm{SNR}_{\mathrm{bilinear,Andromeda}}\\&= \frac{1}{9}\left(17.13 + 17.25 + 17.43 + 12.53 + 12.38 + 12.85 + 7.87 + 7.8 + 8.61\right) = 12.65,\\&\mathrm{SNR}_{\mathrm{spline,Andromeda}} = 13.12,\\&\mathrm{SNR}_{\mathrm{wavelet,Andromeda}} = 13.16,\end{aligned}$$

so that

$$\mathrm{rank}_{\mathrm{bilinear,Andromeda,SNR}} = 3,\ \mathrm{rank}_{\mathrm{spline,Andromeda,SNR}} = 2,\ \mathrm{rank}_{\mathrm{wavelet,Andromeda,SNR}} = 1,$$

$$\mathrm{SNR}_{\mathrm{bilinear,Train}} = 23.11,\ \mathrm{SNR}_{\mathrm{spline,Train}} = 24.022,\ \mathrm{SNR}_{\mathrm{wavelet,Train}} = 24.017$$

(note that spline and wavelet are extremely close to being tied), so that

$$\mathrm{rank}_{\mathrm{bilinear,Train,SNR}} = 3,\ \mathrm{rank}_{\mathrm{spline,Train,SNR}} = 1,\ \mathrm{rank}_{\mathrm{wavelet,Train,SNR}} = 2.$$

Also,

$$\mathrm{SNR}_{\mathrm{bilinear,Tree}} = 16.46,\ \mathrm{SNR}_{\mathrm{spline,Tree}} = 16.73,\ \mathrm{SNR}_{\mathrm{wavelet,Tree}} = 16.82,$$

so that

$$\mathrm{rank}_{\mathrm{bilinear,Tree,SNR}} = 3,\ \mathrm{rank}_{\mathrm{spline,Tree,SNR}} = 2,\ \mathrm{rank}_{\mathrm{wavelet,Tree,SNR}} = 1,$$

and

$$\mathrm{SNR}_{\mathrm{bilinear,Roof}} = 23.84,\ \mathrm{SNR}_{\mathrm{spline,Roof}} = 24.75,\ \mathrm{SNR}_{\mathrm{wavelet,Roof}} = 24.74$$

(again, spline and wavelet are extremely close to being tied), so that

$$\text{rank}_{\text{bilinear,Roof,SNR}} = 3,\ \text{rank}_{\text{spline,Roof,SNR}} = 1,\ \text{rank}_{\text{wavelet,Roof,SNR}} = 2.$$

Proceeding as before, we eventually obtain that the only non-unit Spearman rank correlations are those involving the Roof image, for example,

$$\rho_{\text{Andromeda,Roof}} = .866,$$

yielding the image rank correlation matrix

$$\text{P}_{\text{image}} = \begin{pmatrix} 1 & 1 & 1 & .866 \\ 1 & 1 & 1 & .866 \\ 1 & 1 & 1 & .866 \\ .866 & .866 & .866 & 1 \end{pmatrix}.$$

Given that many of the deviations from the overall ranks were near ties, this suggests weak dependence of the upsampler rankings on test image choice, in agreement with Dumic et al. (This agrees with the conclusions we will draw from our own data.)

3.2.6 Conclusions

Keeping in mind that the following conclusions are based on sparse data, and that in some cases different ranks were obtained by almost identical results, Spearman rank correlations based on aggregated data from Dumic et al. suggest that the choice of resampling ratio has a strong influence on upsampler ranking, the choice of downsampler a significant influence, the choice of test image a minor influence, and the choice of quality metric no influence whatsoever.

The meta-conclusion is that Spearman rank correlation matrices are a powerful and concise way of summarizing the agreement and disagreement between rankings obtained with different parameter choices for a reconstruction study of the accuracy of image upsampling methods.

4 Experimental setup

> Sir Isaac Newton said, "Indeed rays, properly expressed, are not coloured." SPDs [(spectral power distributions)] exist in the physical world, but colour exists only in the eye and the brain. (Charles Poynton [73])

4.1 Key issues: alignment, colour spaces and accuracy

Before proceeding to a detailed specification of the experimental set up, we discuss two important issues which users should consider carefully before extending or modifying EXQUIRES and which matter to anyone using reconstruction tests to evaluate image resampling methods.

In §4.2, we discuss image geometry in the context of resizing images. This issue is important because misalignment leads to reconstruction errors which should not be attributed to the tested resampling filters themselves. Consequently, a careful specification of the geometrical transformations used for the reconstruction test is extremely important. Without meeting this specification, or changing it all around in a uniform way, plugging "external" image resampling programs into EXQUIRES gives meaningless results unless, perchance, the alignment happens to match.

Another important issue is the choice of colour space used for the gold standard images and the re-enlargement results, as well as, internally, by the resampling filters themselves, both in the downsampling and the upsampling stage. Within EXQUIRES, three colour spaces are used: sRGB, linear RGB with sRGB primary colours, and XYZ. Experimental setup issues related to colour spaces are discussed in §4.3.

In the context of an image resampling test suite, another issue is worth mentioning: computation accuracy. This is discussed in §4.4.

The remainder of the chapter is devoted to less critical test configuration features.

4.2 Image geometry conventions: "align (image) corners" vs "align (corner pixel) centres"

Since a key component of the EXQUIRES test suite involves comparing re-enlarged images with a gold standard image, it is important that the pixels at each location in these images are aligned with one another. If the images are misaligned, the errors are meaningless, because alignment errors overwhelm sampling errors.

While there are many methods of specifying the geometrical correspondence between an image and a resized version of the image, there are two that are most commonly used: "align (corner pixel) centres" and "align (image) corners". This section defines the method being used with the EXQUIRES test suite ("align corners") and discusses the most common alternative ("align centres") for contrast.

4.2.1 Index-based pixel location

Throughout this thesis, the pixels of an $m \times n$ image (with m columns and n rows, that is, with a width of m pixels and height of n pixels) are indexed as follows: The top left pixel has index $(0, 0)$, the bottom right pixel has index $(m - 1, n - 1)$. Enlarging one of these images produces a new image with a width of M pixels and height of N pixels.

It is convenient to associate the pixel with index (i, j) with the position (i, j) in a Cartesian plane with second axis pointing down. A key component of image resizing is then the specification of the geometrical correspondence between locations within the plane original that corresponds to the $m \times n$ image and locations within its $M \times N$ version.

4.2.2 Resizing with "align centres"

The forward affine transformation $\mathrm{T} : \mathbb{R}^2 \to \mathbb{R}^2$ from the input image plane to the output image plane is

$$\mathrm{T}(x, y) = \left(\frac{M-1}{m-1} x, \frac{N-1}{n-1} y \right).$$

This forward transformation is the only affine transformation that matches the coordinates of corner pixels. In other words, T is specified by

$$\mathrm{T}(0, 0) = (0, 0),$$

$$\mathrm{T}(m-1, 0) = (M-1, 0),$$

$$\mathrm{T}(0, n-1) = (0, N-1),$$

$$\mathrm{T}(m-1, n-1) = (M-1, N-1).$$

Nearest neighbour image interpolation works well with "align centres" because, invariably, some input and output sampling positions are exact matches. In fact, all the output sampling positions match input sampling positions when reducing an $(r \times M + 1) \times (s \times N + 1)$ image to $M \times N$, with r and s positive integers. Nearest neighbour then boils down to skipping consecutive groups of r rows and s columns ("decimating").

When $(m-1)/(M-1)$ is an integer, downsizing by filtering with the "align centres" convention is such that all sampling locations within the output image correspond exactly to pixel locations within the input image. In other words, downsampling is then equivalent to convolution followed by decimation. Likewise, if $(M-1)/(m-1)$ is an integer, enlarging by filtering is such that the positions of the sampling locations within the output image correspond to a small set of positions relative to pixel locations within the input image. In other words, upsampling is then equivalent to applying a small number of convolutions and assembling the output image from decimated versions of the various results.

Of the two image geometry conventions, "align centres" is the best for enlarging, because no extrapolation occurs with it in this situation, and consequently the impact of the

abyss is minimized. However, "align centres" is generally not as good as "align corners" when reducing size, because it gives more weight to boundary pixels than interior pixels, and because the abyss has more impact on boundary pixels with the former convention.

A prototype of the EXQUIRES test suite using the "align centres" geometry is discussed by Robidoux et al. [16].

4.2.3 Resizing with "align corners"

The forward "align corners" affine transformation T is constructed so that

$$\begin{aligned}
\mathrm{T}\left(-\frac{1}{2},-\frac{1}{2}\right) &= \left(-\frac{1}{2},-\frac{1}{2}\right),\\
\mathrm{T}\left(m-\frac{1}{2},-\frac{1}{2}\right) &= \left(m-\frac{1}{2},-\frac{1}{2}\right),\\
\mathrm{T}\left(-\frac{1}{2},n-\frac{1}{2}\right) &= \left(-\frac{1}{2},n-\frac{1}{2}\right),\\
\mathrm{T}\left(m-\frac{1}{2},n-\frac{1}{2}\right) &= \left(m-\frac{1}{2},n-\frac{1}{2}\right).
\end{aligned}$$

The geometrical basis of this choice is as follows: Interpreting pixels as unit squares centred at the corresponding index, the top left corner of the top left pixel is located at $(-1/2,-1/2)$ (since its centre is located at $(0,0)$), the top right corner of the top right pixel of the input image is located at $(m-1/2,-1/2)$ (since its centre is located at $(m-1,0)$), etc.

$$\mathrm{T}(x,y)=\left(\frac{M}{m}x+\frac{1}{2}\left(\frac{M}{m}-1\right),\frac{N}{n}y+\frac{1}{2}\left(\frac{N}{n}-1\right)\right)$$

is the only affine transformation that matches the outer corners of the corner pixels. When resizing images, the inverse of this affine transformation is used for reverse look up by ImageMagick [74].

Box filtering works well with "align corners" because, when reducing an $(r\times M)\times(s\times N)$ image to $M\times N$ with r and s positive integers, it boils down to averaging groups of $r\times s$ pixels.

When m/M is an integer, downsizing by filtering with the "align corners" convention is such that the positions of the sampling locations within the output image correspond to a small set of positions relative to pixel locations within the input image. Consequently, downsampling is then equivalent to convolution followed by decimation. Likewise when M/m is an integer, in which case upsampling is equivalent to applying a small number of convolutions and assembling the output image from decimated versions of the various results.

Of the two main image geometry conventions, "align corners" is the best one for reducing images, because output image sampling positions correspond to interior input sampling positions, and consequently the impact of the abyss is minimized: The sampling positions, as measured in the input image, "creep inward" when reducing size with the "align corners" image geometry convention. In addition, with "align corners", the weights used with boundary pixels are smaller than those used with interior pixels.

When enlarging, "align corners" is less desirable than "align centres" because sampled positions "creep outward", and extrapolation is necessary near the boundary. As a result, boundary pixels are weighted more than interior ones, and the abyss has more of an impact on the enlarged image.

4.2.4 Align corners is the convention used in this version of EXQUIRES

The current version of EXQUIRES uses the "align corners" convention, like its very first prototype [14]. It would not be a huge endeavour to modify EXQUIRES so that it uses the other convention, like its second prototype [16].

4.3 Linear light ("physical") vs sRGB ("perceptual") resampling

Because they concern engineering solutions to psychovisual and technical problems, colour space issues are much more complex than the uninitiated can imagine. We will, therefore, only discuss issues relevant to an educated use of the EXQUIRES test suite, without attempting to provide definitions applicable outside of the confines of an image reconstruction suite. More comprehensive discussions are found in [72, 73, 75].

A raster image consists of rows of pixels, which are discrete locations at which colour is specified.

> Color is the perceptual result of light in the visible region of the spectrum, having wavelengths in the region of 400 nm to 700 nm, incident upon the retina. (Charles Poynton [73])

Thus, there are two main components to colour: A strictly physical component, which directly relates to the spectral power distribution of light (which describes "how much" of each wavelength the light contains), and a perceptual component, which relates to the sensory reaction triggered by the light incident to the retina of a human viewer. Consequently, there are two main types of colour representations: "linear light" colour spaces are closely associated with the "physical" component of colour, and "perceptual" colour spaces are more closely associated with the psychovisual system.

Although special purpose colour systems, like CMYK (Cyan-Magenta-Yellow-Black, primarily used in printing systems that use black in addition to coloured inks) or RGBA (in which "A" stands for the transparency, or "alpha," channel) have more than three colour channels, and "true grayscale" images (without transparency) only need one "colour" channel, colour is satisfactorily specified with three numbers. For this reason, most colour spaces use three coordinates [73].

In the XYZ colour space, each of the coordinates is associated with the response of one of the three type of photoreceptor cone cells to incoming light [73]. The XYZ colour space

is the standard linear light (physical) colour used to describe colours perceptible to human beings [73].

The ubiquitous sRGB is a perceptual colour space. In the sRGB colour space, the three coordinates relate to the primary colours red ("R"), green ("G") and blue ("B"). However, the relationship between the three sRGB component coordinates and incident light power is not linear. Instead, sRGB colour coordinates are such that changing the value of one of the coordinates (very approximately) has the same visual impact regardless of the initial value [76].

The linear RGB with sRGB primaries colour space is a linear (physical) version of the sRGB colour space.

What makes the XYZ and linear RGB with sRGB primary colours "linear" or "physical" is that the colour channel values each approximate a weighted average of light intensities over a range of wavelengths. Indeed, X, Y and Z are defined as weighted integrals of the spectral power distribution $I(\lambda)$ [77]. Linear RGB with sRGB primaries can be understood two different ways. Linear RGB with sRGB primaries can be obtained from sRGB by undoing its defining nonlinear transformation (which is approximately a power law often called "gamma encoding" or "gamma correction"). Alternately, linear RGB can be obtained by applying a linear transformation to XYZ colour values [76]. This latter definition makes clear that linear RGB with sRGB primaries inherits its linearity from XYZ, and actually, that it can itself be described as an integral of $I(\lambda)$ with different weighting functions. Warning: There are subtleties having to do with mismatches in the ranges of colours contained in the XYZ and sRGB colour spaces and the issue of white point choice ("What is white?" [73]).

In a typical image processing toolchain, pixel values are combined, linearly and nonlinearly, as well as clamped, at several stages of the toolchain that extends from the captured real world scene to the final image. In particular, one of many consequences of the clamping is that nominally linear filters are invariably used within a nonlinear context: clamping

to a range is a nonlinear operation. (The only filters with respect to which this observation may be deemed somewhat irrelevant are those which do not produce overshoots and undershoots like nearest neighbour, bilinear, B-spline smoothing, and Gaussian blur.) In this respect, it should be noted that the sRGB colour space covers a somewhat small subset of the colours perceptible by human beings, that is, it strongly clamps the captured scene (see Wikipedia user Spigget's chart, which shows the sRGB colour space as it fits inside XYZ [78]).

Another source of nonlinearity within the toolchain is the use of perceptual colour spaces. The sRGB colour space, as well as, for example, the Lab colour spaces—Hunter Lab and CIE 1976 L*a*b* among them—are the result of applying a nonlinear transformation to linear light coordinates [73]. Since the results of averaging are affected by nonlinear transformations, so are, generally, the results of filtering, in particular, the results of resampling. For example, averaging sRGB colour values directly generally gives a lower (darker) result than averaging through linear RGB—that is, converting the values to linear RGB, averaging in this colour space, and converting back the average to sRGB—because the transformation from sRGB to linear RGB is convex. As a result, even simple resampling methods like bilinear interpolation give different results in these two alternate toolchains.

Two versions of every upsampling method and downsampling method are tested in the plain vanilla version of the EXQUIRES suite:

Plain (sRGB) version: The "plain" version of each upsampling method takes 16-bit sRGB pixel values literally. Since the test images are sRGB, no colour space transformation is performed at any point within the resampling, and "linear" filtering and averaging is performed even though sRGB is a nonlinear colour space.

Linear light (linear RGB or XYZ) version: In the "linear light" toolchain, the resampling method is applied to a linear light version of the image using the standard three step procedure [74]:

1. the 16-bit sRGB input images are converted to a linear light colour space (16-bit linear RGB with sRGB primaries for all methods except Nohalo-LBB, which uses XYZ)
2. resampling is performed using the linear light pixel values
3. the linear light enlargement is converted back to 16-bit sRGB.

In the current version of EXQUIRES, "_linear" is used to distinguish the linear light versions of operations from those that take sRGB values as is, versions identified with the "_srgb" suffix.

"Which colour space?" is a question that needs to be asked at every step of the testing procedure, from the conversion of raw image data performed inside or outside of the digital camera, to the storage format of the gold standard image and the images produced at various stages of the toolchain, to the colour space used internally by the resampling filters, to the colour space used by each image difference metric. In addition, the mechanism used to convert from one colour space to another needs to be specified, since some of the transformations are not reversible, and the accuracy of the conversion software affects the accuracy of the results.

4.3.1 The ICC sRGB v2 profile with the Perceptual rendering intent is used in this version of EXQUIRES

In the present context, a colour profile is an encoded specification of the mapping that converts between two colour spaces [79]. (They are also used to map colours stored within a digital image to colours displayed or produced by a specific device or group of devices, like a computer monitor or a printer [79].) Rendering intents are colour profile options used to mitigate subjective side effects [75, 79]. Given that different colour spaces have different gamuts, meaning that the range of physical combinations of wavelenghts they contain are different, truncation ("clamping to the gamut") is involved, in particular when so-called

raw digital capture images, or the result of processing a raw image in a format that is not overly lossy, are converted to sRGB, a fairly "small" colour space. For this reason, the choice of profile is especially significant in the context of the production of the test image bank [80].

The ICC is the International Colour Consortium of vendors of hardware and software which perform colour management [79]. The EXQUIRES test suite is standardized so that all test and derived images are produced with the ubiquitous ICC sRGB v2 profiles with Perceptual rendering intent, or with formulas taken from the specification of the sRGB colour space and built into ImageMagick. By virtue of being uncommon, sRGB v4 profiles are not used even though they have several advantages over v2, notably more accurate inverse transformations [81]. (sRGB v4 profiles are getting wider support, however, so they may be a good choice for a future version of EXQUIRES [82].)

Raw digital photograph processing tools like Photivo [83] and RawTherapee [84] allow one to choose both the profile used to perform the conversion to sRGB, and the so-called rendering intent option used within the profile, which in our context affects the way colour values that fall outside of sRGB are forced into it. (The reader should note that legacy profiles produced by hardware vendors often implemented rendering intents in somewhat arbitrary ways.)

Whenever the VIPS image processing library had to perform a conversion into or out of sRGB, either the `sRGB.icm` profile distributed with its GUI (Graphical User Interface) NIP2 (New Image Processor 2) [85] or the `sRGB_IEC61966-2-1_black_scaled.icc` profile, downloaded directly from the ICC [81], was used. According to the ICC, black scaled profiles are representative of the majority of v2 profiles in current use [81]. It turns out that the `sRGB_IEC61966-2-1_black_scaled.icc` profile gives identical results to the `sRGB.icm` distributed with NIP2 when used with the Perceptual rendering intent; consequently, the distinction is moot.

With ImageMagick, the internal conversion machinery between linear RGB with sRGB primaries and sRGB is used instead of one based on ICC profiles. This machinery is independent of the rendering intent. (Future versions of EXQUIRES may also use the "algebraic" or "formulaic" approach for VIPS operations, since VIPS recently acquired this capability.)

In any case, once an image has been converted to sRGB, the difference between the algebraic or various profile approaches for converting back to linear RGB or XYZ is not as significant, since this conversion mostly involve colours already in the sRGB colour space. The distinction is much more significant in the other direction, in particular, when converting from raw digital camera output to sRGB, that is, from a "large" colour space to a relatively "small" one.

Perceptual rendering intent was chosen for three reasons [86]. First, it minimizes clipping of out of gamut colours. Such clipping greatly changes the local geometrical character of rendered images, which is quite undesirable given that box filtering is the most commonly used downsampling method in the context of re-enlargement. Second, Perceptual is the most reversible of the rendering intents, and the linear light toolchains involve conversion into and out of sRGB. The third reason is that the Perceptual rendering intent is the most commonly used.

4.4 Computation accuracy

When one states that, say, the Lanczos filter is tested, what is really being tested is a specific implementation of the Lanczos filter (in this thesis, the one obtained using a recent release of version 6 of the ImageMagick image processing library [42] compiled with floating point intermediate result storage instead of the more common 16-bit unsigned integer storage, with colour space conversions performed with internal implementations of the algebraic formulas defining the relationship between sRGB and linear RGB colours instead

of colour profiles). Given this, there is an implicit assumption that the implementation is precise enough to accurately represent the "infinite precision" result of filtering with the "ideal" Lanczos filter applied to "infinite precision" data resulting from error free colour conversions, and that the boundary handling specific to the implementation does not overly affect results. (This last issue could be mitigated by cropping before using image difference metric, with the unfortunate side effect that poor boundary handling becomes invisible.) With respect to accuracy issues, note that ImageMagick performs filtering and colour space conversions with (fairly dense) LUTs (Look Up Tables), while the tested "external" method from the image processing library VIPS (Virtual Image Processing System) [43], namely Nohalo-LBB, uses a more precise LUT-free double precision computation system, but an arguably less accurate colour space conversion system that relies on colour profiles. (Bleeding edge VIPS can perform the transformations using "algebraic" approaches, but this enhancement was not done in time to be used to derive the results of this thesis.)

We believe that the various implementations of the filters, colour space transformations, and image difference metrics used to produce the data are sufficiently accurate not to influence the rankings, but we have not directly measured their accuracy. In particular, colour space conversions are at the top of the list of EXQUIRES components that should be revisited [87]. On the other hand, our decision to use sRGB gold standard test images, and our reliance on widely used colour space conversion tools, make the tests more representative of common "real world" toolchains.

4.5 Tested resampling ratios

The remaining sections of this chapter list specific ingredients of the EXQUIRES test suite as configured for this thesis.

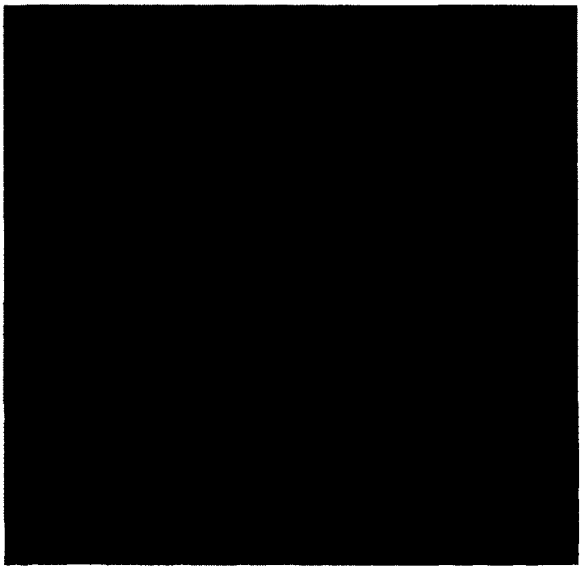
Seven downsampling ratios are used to downsample the original test images: 2, 3, 4, 5, 6, 7 and 8. The same ratios are used to re-enlarge the downsampled version back to full size, namely 840×840. 840 was chosen because it is the LCM (Least Common Multiple) of the tested ratios. Since the "align corners" image geometry convention is used throughout, the smaller images have sizes 420×420 (ratio 2), 280×280 (ratio 3), 210×210 (ratio 4), 168×168 (ratio 5), 140×140 (ratio 6), 120×120 (ratio 7) and 105×105 (ratio 8).

4.6 Test images

The results of this thesis are obtained by re-enlarging reduced versions of seventeen copyfree test images. These test images are 840×840 colour images stored in 16-bit TIFF with the `sRGB_IEC61966-2-1_black_scaled.icc` ICC profile (or equivalent) used with Perceptual rendering intent [81].

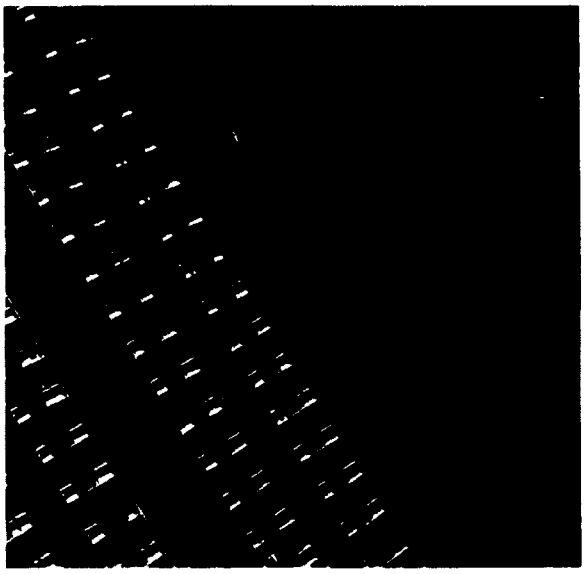
EXQUIRES includes one such copyfree test image, `wave.tif`, shown in Figure 4.1. The wave image is a cropped and downsampled version of a high quality Library of Congress scan of an ancient woodblock print (the copyist is anonymous) of Hokusai Katsushika's Kanagawa oki nami ura (The great wave off shore of Kanagawa) [88].

The test image bank 16bit840x840images [80], specifically assembled for this thesis by Robidoux et al., contains sixteen additional copyfree images. These additional images are shown in Figures 4.2–4.5. Every one is a carefully downsampled version of a digital photograph taken with a high quality Digital Single Lens Reflex camera. Details of their capture, processing and copyfree licensing are found in the accompanying documentation [80]. A copy of the wave image is also included with the 16bit840x840images test bank. Thus, the results of this thesis are based on the seventeen images distributed with 16bit840x840images.



(a) wave (author: Hokusai Katsushika/anonymous (Library of Congress scan))

Figure 4.1: Test image built into EXQUIRES.



(a) apartments (author: Henry Ho)

(b) baby (author: Minglun Gong)

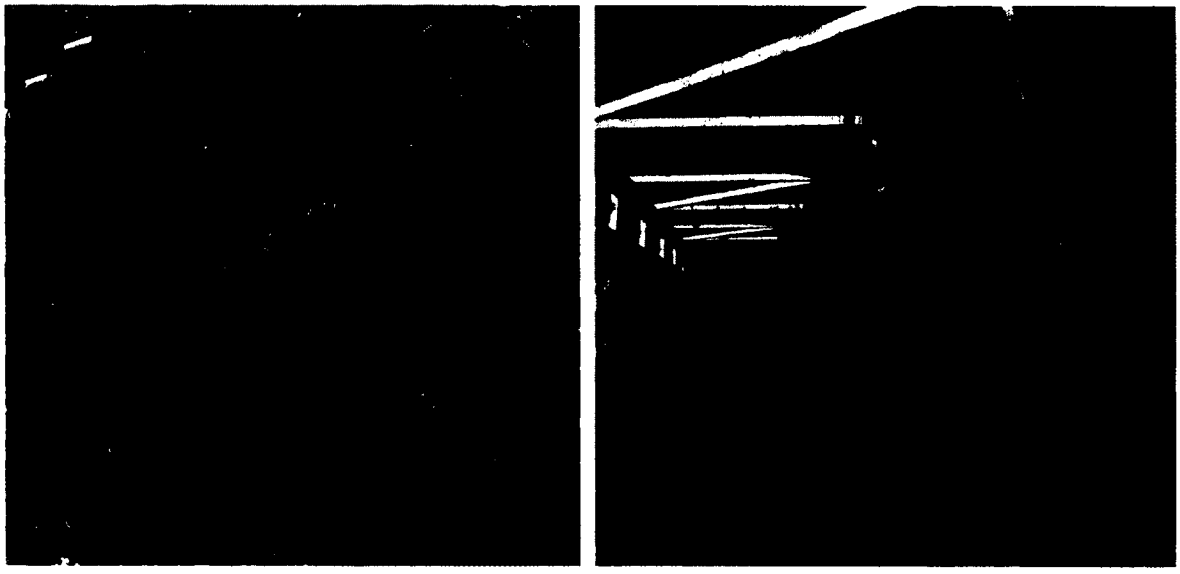
(c) boy (author: John Cupitt)

(d) cabins (author: Anthony Barnett)

Figure 4.2: Test images from the 16bit840x840images bank.

(a) cat (author: Luiz E. Vasconcellos)

(b) curios (author: Jean-François Avon)



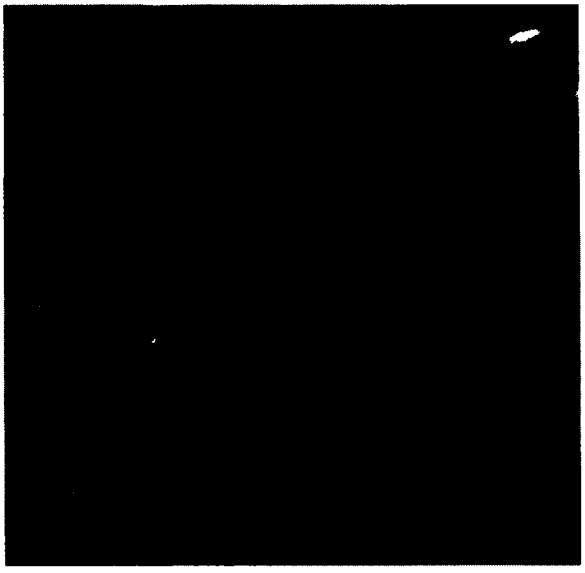
(c) dragon (author: Michael Muré)

(d) footbridge (author: Jana Duncan)

Figure 4.3: Test images from the 16bit840x840images bank.

(a) frog (author: Luiz E. Vasconcellos)

(b) garland (author: Mukund Sivaraman)



(c) horse (author: Kirk Martinez)

(d) man (author: Holly Graham)

Figure 4.4: Test images from the 16bit840x840images bank.

(a) paint (author: Anthony Barnett)

(b) shed (author: Jana Duncan)

(c) tower (author: Jana Duncan)

(d) wreck (author: Anthony Barnett)

Figure 4.5: Test images from the 16bit840x840images bank.

4.7 Tested downsampling methods

Five downsampling methods, each with two variants, are used by the current version of EXQUIRES. In the following, code for both linear light downsampling and "direct sRGB" downsampling is shown. EXQUIRES can do both. In the current version of EXQUIRES, all downsampling is performed with ImageMagick 6 [42, 74, 89]. Unix shell syntax is given. (To use on Windows systems, see [90].) Within Python, the syntax is slightly different; see EXQUIRES' source code (Appendix A.6).

4.7.1 Box filtering

Linear light box filtering mimics scene recapture with a mosaic of contiguous square digital sensors larger than those used to capture the original image [14]. When downsampling while preserving the aspect ratio, box filtering basically consists of averaging square groups of pixel values.

The sRGB "`box_srgb`" downsampler and the linear light "`box_linear`" downsampler are obtained with

box_srgb:

```
convert input.tif \
-filter Box \
-resize nxn -strip output.tif
```

box_linear:

```
convert input.tif -colorspace RGB \
-filter Box \
-resize nxn -colorspace sRGB -strip output.tif
```

where n is the width = height of the reduced version of the image.

4.7.2 Gaussian filtering

Gaussian filtering is another low pass filter. We use $\sigma = 1/2$, the default ImageMagick value, recommended by Craig DeForest [91]. The standard tensor approximation of the filter and default truncation extent, are used.

The **gaussian_srgb** and **gaussian_linear** downsamplers are obtained by replacing "`Box`" by "`Gaussian`" in the calls of §4.7.1.

4.7.3 EWA (Elliptical Weighted Averaging) Lanczos (Jinc-windowed Jinc) 3-lobe filtering

In addition to the fact that the Fourier transform of the Jinc radial function is the characteristic function on a disc, there is accumulating empirical evidence that Jinc-windowed Jinc 3-lobe is a very good low pass filter.

The **ewa_lanczos3_srgb** and **ewa_lanczos3_linear** downsamplers are obtained by replacing "`Box -resize`" by "`Lanczos -distort Resize`" in the calls of §4.7.1.

4.7.4 Lanczos (Sinc-windowed Sinc) 3-lobe filtering

Lanczos is a low pass filter oft recommended for producing sharp looking natural image reductions and thumbnails.

The **lanczos3_srgb** and **lanczos3_linear** downsamplers are obtained by replacing "`Box`" by "`Lanczos`" in the calls of §4.7.1.

Note that with this filter—actually, with any tensor (`-resize`) filter with negative weights—it is imperative to use a version of ImageMagick compiled in HDRI mode, otherwise the results are clipped between the two orthogonal passes. HDRI ImageMagick uses float buffers, so that no clipping occurs in intermediate stages of the computation.

4.7.5 Nearest neighbour interpolation with averaged ties

We also test nearest neighbour interpolation. The spirit of nearest neighbour is that the colour of the input pixel nearest to the sampling location is chosen. This, of course, leaves open the question of how to resolve ties (when two or more input pixel locations are equally close).

When downsampling with the "align corners" convention by an exact even ratio in one or both directions, all sampled pixel locations are equidistant to two or four input pixel locations, that is, there are ties with respect to the nearest neighbour criterion. Since ties are not resolved randomly by ImageMagick, we do not use the ImageMagick version of nearest neighbour sampling, namely `Point`, in order to prevent alignment issues (resolving ties top-left, for example, has the effect of translating the entire image by a sub-pixel distance in this direction). Instead, we use an ImageMagick command with the effect of averaging the pixel values of the nearest pixels when there are more than one. Note that this command will have the desired behaviour—nearest neighbour when there is only one nearest neighbour, and average the nearest neighbours when there are several—only when downsampling by exact factors of the input width and height. The correct behaviour is based on the values returned by bilinear interpolation when one is sampling at original pixel locations (when both the horizontal and vertical downsampling ratios are odd) or at locations equidistant from the two or four nearest neighbours.

The **nearest_srgb** and **nearest_linear** downsamplers are obtained by replacing "`Box -resize`" by "`Triangle -interpolative-resize`" in the calls of §4.7.1.

4.8 Tested upsampling methods

This chapter gives brief descriptions and code for the tested upsampling methods. Given that this thesis is primarily about the process of comparing upsampling methods, not about the methods themselves, the descriptions are terse. More details are given when the method is novel. In any case, detailed calls to the relevant libraries are provided, so that both the method and its implementation are unambiguously documented.

The image enlargement methods included in the base version of the test suite are all ImageMagick methods [42]. ImageMagick has a rich catalogue of resampling methods, all implemented with the "align corners" convention. (Some of the methods can be used with arbitrary image geometry conventions with fairly complex calls.) Methods can be further customized by changing the values of parameters that allow, among other things, "stretching" the support of the filters. Besides "baseline" methods (nearest neighbour, bilinear), the test suite includes a sampling of the best standard methods implemented in ImageMagick. Windowed Sinc methods are tested with all implemented windowing methods. In addition, a number of novel methods, formulated by Thomas Theußl, Helwig Hauser and Eduard Gröller (§4.8.2.1), one by Anthony Thyssen (§4.8.6.2), one by Henry Ho (§4.8.6.4), and several by Nicolas Robidoux and collaborators (§4.8.2.2, 4.8.6.1, 4.8.6.3, 4.8.6.5, 4.8.6.6 and 4.8.6.7), are included.

Tested ImageMagick methods fall into two broad categories:

- tensor (a.k.a. 2-pass, a.k.a. orthogonal) methods, in which image enlargement is performed by applying a "1D" resampling method twice (once horizontally, and once vertically)

- Elliptical Weighted Averaging (EWA) methods, in which enlargement is done using weights that depend on the Euclidean distance from the sampling location.

When the aspect ratio is preserved, the support of a tensor filter is a square, and the support

of an EWA filter is a disc. EWA methods are particularly suited for texture mapping and image warping; They actually were invented to perform perspective transformations [92]. This thesis is likely the first academic publication in which EWA methods are carefully tested in the context of image enlargement.

To illustrate the extensibility of the EXQUIRES test suite with respect to the addition of "arbitrary" upsamplers, we also test a non-ImageMagick method, namely the VIPS [43] library's implementation of Nohalo-LBB ("no halo" face split subdivision with Locally Bounded Bicubic finishing scheme). The novel Nohalo-LBB method, developed by N. Robidoux and collaborators, is discussed last (in §4.8.7.1).

4.8.1 Standard interpolatory linear tensor methods

4.8.1.1 Nearest neighbour

Nearest neighbour is the only tested upsampling method for which the sRGB and linear light versions are identical: With nearest neighbour, at least in principle, no arithmetic is performed on the pixel values. Consequently, it should not matter whether the nearest neighbour look up is performed with sRGB or linear light values, since the latter are converted back to sRGB at the end. The error introduced by converting between sRGB and linear light should be very small with 16-bit images. As a sanity check (to ensure that the same basic result is obtained), we test both nearest neighbour versions.

In ImageMagick, the nearest neighbour upsampling filter is called "Point" [93]. Consequently, the sRGB and linear light nearest neighbour enlargement operations are

nearest_srgb:

```
convert input.tif \
-filter Point \
-resize 840x840 -strip output.tif
```

nearest_linear:

```
convert input.tif -colorspace RGB \
-filter Point \
-resize 840x840 -colorspace sRGB -strip output.tif
```

4.8.1.2 Bilinear

In ImageMagick, the bilinear upsampling filter is called "Triangle" [93]. Consequently, the sRGB (**bilinear_srgb**) linear light bilinear (**bilinear_linear**) enlargement operations are obtained by replacing "`Point`" by "`Triangle`" in the calls shown in §4.8.1.1.

4.8.1.3 Cubic Hermite

In terms of visual quality, tensor filtering (= interpolation) with cubic Hermite (with null nodal slopes) is not very good. It is included for the sake of completeness, and because it is one of the few standard halo-free methods.

In ImageMagick, the cubic Hermite (with null end slopes) filter is called "Hermite" [93], so that **cubic_hermite_srgb** and **cubic_hermite_linear** are obtained by replacing "`Point`" by "`Hermite`" in the calls of §4.8.1.1.

4.8.1.4 Catmull-Rom

In ImageMagick, the Catmull-Rom filter is called "Catrom" [93]. The **catmull_rom_srgb** and **catmull_rom_linear** calls are obtained by replacing "`Point`" by "`Catrom`" in §4.8.1.1.

4.8.1.5 Lanczos 2-, 3- and 4-lobe

In ImageMagick, the Lanczos (Sinc-windowed Sinc) 2-lobe filter is called "Lanczos2" [93]. The **lanczos2_srgb** and **lanczos2_linear** calls are obtained by replacing "`Point`" by "`Lanczos2`" in §4.8.1.1.

The Lanczos 3-lobe filter is called "Lanczos" [93]. The **lanczos3_srgb** and **lanczos3_linear** calls are obtained by replacing "`Point`" by "`Lanczos`" in §4.8.1.1.

Lanczos 4-lobe is obtained by setting the number of lobes. The **lanczos4_srgb** and **lanczos4_linear** calls are obtained by replacing "`Point`" by "`Lanczos -define filter:lobes=4`" in §4.8.1.1.

4.8.1.6 Other windowed-Sinc filters

We test nine windowed Sincs besides Lanczos: Bartlett-, Bohman-, Blackman-, Cosine-, Hamming-, Hann-, Kaiser-, Parzen- and Welch-windowed Sinc. In addition, three sets of parameters for the Kaiser windowing function are used. Blackman, Hamming and Hann are generally considered to be the best choices for digital signal processing [93, 94], although it is unclear how relevant such recommendations are with respect to image resampling. Cosine windowing is recommended by Meijering et al. [95].

"Bartlett" is the name given to the "Mexican hat" (linear interpolation basis) function when used as a windowing function, and "Parzen" is the name given to the uniform cubic B-spline in the same context.

The Hamming and Kaiser window functions are not continuous by virtue of not vanishing at the edge of the window, although Kaiser's discontinuity is minuscule when the parameter β is large. Such discontinuities lead to slope discontinuities when the images are greatly enlarged, and lead to unsightly artifacts. So does the slope discontinuity at the centre of the support of the Bartlett window function.

Closely related to the Hamming windowing function, the Hann windowing function

is often incorrectly called "Hanning." The Welch window is sometimes incorrectly called "Welsh" [93, 96]. This is the case within older versions of ImageMagick.

A slightly different type of call is generally used to use windowed Sinc filters other than Cosine, Lanczos and Welch [93], owing to the fact that the default number of lobes for the other windows is four instead of three.

4.8.1.7 Cosine- and Welch-windowed Sinc 2-, 3- and 4-lobe

The sRGB and linear light Cosine and Welch-windowed Sinc enlargement operations are obtained by substituting the window name for "`Lanczos`" in §4.8.1.5.

4.8.1.8 Bartlett-windowed Sinc 2-, 3- and 4-lobe

The sRGB and linear light Bartlett-windowed Sinc 2-lobe enlargement operations **bartlett2_srgb** and **bartlett2_linear** are obtained by replacing "`Point`" by "`Bartlett -define filter:lobes=2`" in §4.8.1.1. The **bartlett3_srgb** and **bartlett3_linear** calls are obtained by replacing "`Point`" by "`Bartlett -define filter:lobes=3`" in §4.8.1.1. Because four is the default number of lobes in ImageMagick for windowed Sinc methods other than Cosine, Lanczos and Welch, **bartlett4_srgb** and **bartlett4_linear** are obtained by replacing "`Point`" by "`Bartlett`" in §4.8.1.1.

4.8.1.9 Blackman-, Bohman-, Hamming-, Hann- and Parzen-windowed Sinc 2-, 3- and 4-lobe

These windowed Sinc methods are obtained by following the recipe given in §4.8.1.8 for Bartlett-windowed Sinc 2-, 3- and 4-lobe, replacing "`Bartlett`" by the corresponding window function name.

4.8.2 Novel interpolatory linear filtering methods

4.8.2.1 Kaiser-windowed Sinc 2-, 2- and 4-lobe

Kaiser is a parameterized family of window functions. We use the values of the parameter β computed by Thomas Theußl, Helwig Hauser and Eduard Gröller by matching Taylor series coefficients [97].

In ImageMagick, the sRGB and linear light Kaiser-windowed Sinc 2-lobe enlargement operations **kaiser2_srgb** and **kaiser2_linear** are obtained by replacing "`Point`" by "`Kaiser -define filter:lobes=2 -define filter:kaiser-beta=5.36`" in §4.8.1.1. The **kaiser3_srgb** and **kaiser3_linear** calls are obtained by replacing "`Point`" by "`Kaiser -define filter:lobes=3 -define filter:kaiser-beta=8.93`" in §4.8.1.1. **kaiser4_srgb** and **kaiser4_linear** are obtained by replacing "`Point`" by "`Kaiser -define filter:kaiser-beta=12.15`" in §4.8.1.1.

Warning: There are two different Kaiser window parameters called "α" in the literature and used in published software. The Kaiser β parameter used by ImageMagick, and consequently in EXQUIRES, is equal to π times the smaller of the two versions of the parameter α. Equivalently, ImageMagick's β is exactly equal to the larger of the two versions of the parameter α.

4.8.2.2 KaiserSoft- and KaiserSharp-windowed Sinc 2-, 3- and 4-lobe

Using a back of the envelope computation involving the Nyquist frequency, N. Robidoux formulated two (rather obvious) rules of thumb setting the value of the Kaiser β parameter in terms of the number of Sinc lobes (which is equal to half the window width) [98].

The "KaiserSharp" rule of thumb sets β to the *minimum* possible number of sampling

points contained in the 1D window when enlarging times the relevant Nyquist frequency $(1/2)$, times π, and the "KaiserSoft" rule of thumb sets β to the *maximum* possible number of sampling points contained in the 1D window when enlarging times Nyquist times π. In other words,

$$\beta = \frac{\pi}{2}(2(\#\text{ of lobes}) - 1) = \pi\left(\#\text{ of lobes} - \frac{1}{2}\right)$$

for KaiserSharp, and

$$\beta = \pi(\#\text{ of lobes})$$

for KaiserSoft. These values are based on yet another rule of thumb stating that β/π is the bin number of the edge of the main lobe of the frequency response [99].

The KaiserSharp and KaiserSoft β values bracket the values recommended by Thomas Theußl et al. and used to set the key parameter of the "plain" Kaiser-windowed Sincs (§4.8.2.1).

kaisersharp2_srgb and **kaisersharp2_linear** upsampling are obtained by replacing "`5.36`" by "`4.7123889803846899`", **kaisersharp3_srgb** and **kaisersharp3_linear** by replacing "`8.93`" by "`7.853981633974483`", and **kaisersharp4_srgb** and **kaisersharp4_linear** by replacing "`12.15`" by "`10.995574287564276`" in §4.8.2.1. Likewise, **kaisersoft2_srgb** and **kaisersoft2_linear** are obtained by replacing "`5.36`" by "`6.2831853071795865`", **kaisersoft3_srgb** and **kaisersoft3_linear** by replacing "`8.93`" by "`9.4247779607693797`", and **kaisersoft4_srgb** and **kaisersoft4_linear** by replacing "`12.15`" by "`12.566370614359173`".

4.8.3 Standard non-interpolatory linear tensor methods

4.8.3.1 Quadratic B-spline smoothing

In ImageMagick, the quadratic B-spline smoothing filter is called "Quadratic" [93]. The **quadratic_b_spline_srgb** and **quadratic_b_spline_linear** calls are obtained by replacing

"`Point`" by "`Quadratic`" in §4.8.1.1.

4.8.3.2 Cubic B-spline smoothing

In ImageMagick, the cubic B-spline smoothing filter is called "Spline" [93]. The **cubic_b_spline_srgb** and **cubic_b_spline_linear** calls are obtained by replacing "`Point`" by "`Spline`" in §4.8.1.1.

4.8.3.3 Mitchell-Netravali cubic spline

The Mitchell-Netravali filter, under the name "Mitchell", is the default ImageMagick filter for tensor resizing [93]. Consequently, **mitchell_netravali_srgb** and **mitchell_netravali_linear** are obtained by removing "`-filter Point`" from the calls of §4.8.1.1.

4.8.4 Standard interpolatory EWA (Elliptical Weighted Averaging) linear filtering methods

Elliptical Weighted Averaging [100] uses filter kernels very differently than tensor filtering. When enlarging, an EWA (cylindrical) kernel is obtained from the 1D kernel by rotating it around its centre, that is, by turning a function of x into a function of r, the distance from the centre. In other words, the EWA ellipses are discs when enlarging (if the aspect ratio is preserved).

In ImageMagick, one resizes with Elliptical Weighted Averaging instead of tensor filtering by replacing "`-resize`" by "`-distort Resize`" [74]. The `-distort Resize` operation was added to ImageMagick v6.6.9-2. Note that, for a number of versions thereafter, using `-distort Resize` to produce TIFF images added an empty alpha (transparency) channel, overwriting the existing one. In the case of input images without an alpha channel, this can be fixed by inserting "`-alpha Off`" before "`-strip`" in all the

commands that use `-distort Resize` [100]. This is not an issue with recent enough ImageMagick 6 and in ImageMagick 7.

We do not know of any high quality interpolatory EWA upsampling filter, which is why only two are tested: teepee and Hermite. (There is some evidence that blending the two filters may give an EWA filter which is better than both.)

4.8.4.1 EWA Teepee

EWA Teepee filtering, that is, with a cone of radius 1, is called "Triangle" in ImageMagick, due to the fact that slicing the "teepee" gives the "Mexican hat", the basis function for linear filtering and interpolation, the graph of which is an isosceles triangle with a height of 1 and width of 2. The sRGB and linear light EWA enlargement with the teepee kernel are given by

ewa_teepee_srgb:

```
convert input.tif \
-filter Triangle \
-distort Resize 840x840 -strip output.tif
```

ewa_teepee_linear:

```
convert input.tif -colorspace RGB \
-filter Triangle \
-distort Resize 840x840 \
-colorspace sRGB -strip output.tif
```

In terms of visual quality, EWA Teepee does not give satisfactory results when upsampling. It is currently the downsampling component of the GEGL resampler NoHalo [101], used by GIMP (GNU Image Manipulation Program).

4.8.4.2 EWA Hermite

In terms of visual quality, EWA Hermite is probably not a great method.

In ImageMagick, the EWA Hermite filter is called "Hermite" like its tensor counterpart. Upsampling with **ewa_hermite_srgb** and **ewa_hermite_linear** is obtained by replacing "`Triangle`" by "`Hermite`" in the calls of §4.8.4.1.

4.8.5 Standard non-interpolatory EWA linear filtering methods

4.8.5.1 EWA quadratic B-spline smoothing

In ImageMagick, the EWA quadratic B-spline smoothing filter is called "Quadratic" like its tensor counterpart. Upsampling with **ewa_quadratic_b_spline_srgb** and **ewa_quadratic_b_spline_linear** is obtained by replacing "`Triangle`" by "`Quadratic`" in the calls of §4.8.4.1.

4.8.5.2 EWA cubic B-spline smoothing

In ImageMagick, the EWA cubic B-spline smoothing filter is called "Cubic" like its tensor counterpart. Upsampling with **ewa_cubic_b_spline_srgb** and **ewa_cubic_b_spline_linear** is obtained by replacing "`Triangle`" by "`Spline`" in the calls of §4.8.4.1.

4.8.5.3 EWA Jinc-windowed Jinc (Jinc Lanczos) 2-, 3- and 4-lobe

In ImageMagick, the Jinc-windowed Jinc 2-lobe EWA filter is called "Lanczos2" like its Sinc-windowed Sinc tensor counterpart [93]. Upsampling with **ewa_lanczos2_srgb** and **ewa_lanczos2_linear** is obtained by replacing "`Triangle`" by "`Lanczos2`" in the calls of §4.8.4.1.

The Jinc-windowed Jinc 3-lobe EWA filter is called "Lanczos" like its Sinc-windowed Sinc tensor counterpart [93]. Upsampling with **ewa_lanczos3_srgb** and

ewa_lanczos3_linear is obtained by replacing "`Triangle`" by "`Lanczos`" in the calls of §4.8.4.1.

Upsampling with **ewa_lanczos4_srgb** and **ewa_lanczos4_linear** is obtained by replacing "`Triangle`" by "`Lanczos -define filter:lobes=4`" in §4.8.4.1.

4.8.6 Novel non-interpolatory EWA linear filtering methods

With the exception of EWA Mitchell-Netravali, a method most likely first used by Anthony Thyssen of Griffith University, who initially programmed EWA resampling into ImageMagick with the assistance of Fred Weinhaus, and EWA Catmull-Rom, a method most likely first used by Henry Ho although it was described by Anthony Thyssen in the ImageMagick documentation, the novel EWA methods presented in this section are due to N. Robidoux.

4.8.6.1 EWA Robidoux (Keys cubic with $\alpha = 113/(58 + 216\sqrt{2})$)

The Robidoux Keys cubic is the default ImageMagick filter for EWA distortion and resizing [93]. Consequently, **ewa_robidoux_srgb** and **ewa_robidoux_linear** are obtained by removing "`-filter Triangle`" from the calls of §4.8.4.1.

The Robidoux EWA filter is the unique Keys cubic with the following property: When upsampling (actually, when not downsampling) an image with pixel values constant on rows (or columns), the output image's pixel values at positions that correspond to original pixel locations are unchanged. This property can be summarized as follows: The Robidoux EWA filter is interpolatory on images that are constant row-wise (or column-wise).

Here is a more concrete definition of the Robidoux cubic spline: When upsampling (not downsampling) and evaluating the reconstructed surface at an output pixel location that corresponds exactly to an input pixel location, the (positive) coefficients of the input values immediately left, right, top and bottom are twice as large as the (negative) coefficients of

the four input pixel values that are closest in the four diagonal directions (NE, NW, SW and SE) [102].

The way things work out, Robidoux is close to Mitchell-Netravali: Robidoux is the Keys cubic with $\alpha \approx .3109$, Mitchell-Netravali with $\alpha \approx .3333$ [93]. EWA Robidoux is the downsampling component of the GEGL resampling filter LoHalo [103], used by GIMP (GNU Image Manipulation Program); sigmoidized tensor Mitchell-Netravali is LoHalo's upsampling component.

4.8.6.2 EWA Mitchell-Netravali (Keys cubic with $\alpha = 1/3$)

Until it was displaced by Robidoux, the default ImageMagick filter for EWA distortion and resizing was the Mitchell-Netravali cubic. EWA with Mitchell-Netravali as a radial weighting function was determined to be a good method by Anthony Thyssen of Griffith University based on visual comparison with the results produced with a number of alternative kernels.

Upsampling with **ewa_mitchell_netravali_srgb** and **ewa_mitchell_netravali_linear** is obtained by replacing "`Triangle`" by "`Mitchell`" in §4.8.4.1.

4.8.6.3 EWA RobidouxSharp (Keys cubic with $\alpha = 7/(2 + 12\sqrt{2})$)

Upsampling with **ewa_robidoux_sharp_srgb** and **ewa_robidoux_sharp_linear** is obtained by replacing "`Triangle`" by "`RobidouxSharp`" in the calls of §4.8.4.1.

With $\alpha \approx .3690$, RobidouxSharp is, like Robidoux, close to Mitchell-Netravali. On the map of BC-splines [93], Robidoux is closer to B-spline smoothing than Mitchell-Netravali; RobidouxSharp is closer to Catmull-Rom.

Among Keys cubics, the RobidouxSharp EWA filter is characterized by the following minimax property: It minimizes the maximum possible deviation between the input image's values and those at corresponding positions in the output image. (The key mathe-

matical assumption is that pixel values are bounded.) In other words, the RobidouxSharp EWA filter is as close to being interpolatory as a Keys EWA filter can be [104].

There are several alternate, but equivalent, characterizations of the RobidouxSharp cubic. Some can be formulated concisely in terms of "no-op" resampling, that is, resizing an image to the exact same size (with the same alignment and image geometry convention). The first characterization of the RobidouxSharp filter in terms of no-op is that it is the Keys cubic for which the corresponding no-op EWA resampling operation is closest to the identity in ℓ_∞ (linear) operator norm. Equivalently, the matrix representation of EWA RobidouxSharp no-op resampling is the closest to the identify matrix in ℓ_1 matrix norm. (Abyss issues are ignored in these characterizations.) A less abstract characterization: When sampling at an output location that corresponds exactly to an input location (and not downsampling), the sum of the weights corresponding to locations other than the input location under consideration is exactly 0. By symmetry, this means that the (positive) weights of the nearest pixels to the left, right, top and bottom are exactly as large as the (negative) weights of the four closest pixels in the diagonal direction. Yet another characterization: When sampling at an input pixel location without downsampling, the weight of the original input pixel in the weighted sum is exactly 1 [105].

4.8.6.4 EWA Catmull-Rom

Upsampling with **ewa_catmull_rom_srgb** and **ewa_catmull_rom_linear** is obtained by replacing "`Triangle`" by "`Catrom`" in the calls of §4.8.4.1.

EWA Catmull-Rom was apparently first seriously considered by Henry Ho, who suggested, in the Retouching forum of Digital Photography Review, that it being strongly sharpening would allow one to skip the commonly used USM (Un-Sharp Mask) sharpening step when reducing images [106]. Catmull-Rom as an EWA filter had earlier been mentioned by Anthony Thyssen, who showed its cardinal basis function without discussing

whether it was useful [107, 108].

4.8.6.5 EWA LanczosRadiusN (Jinc-windowed Jinc N-lobe with support shrunk to radius N)

These methods were proposed by N. Robidoux in an ImageMagick Forum post [109].

EWA LanczosRadius2 is obtained from plain vanilla EWA Lanczos 2 by shrinking its support so that it is a disk of radius exactly 2. Upsampling with **ewa_lanczos_radius2_srgb** and **ewa_lanczos_radius2_linear** is obtained by replacing "`Triangle`" by "`LanczosRadius -define filter:lobes=2`" in §4.8.4.1.

EWA LanczosRadius3 is obtained from EWA Lanczos by shrinking its support so that it is a disk of radius exactly 3. Upsampling with **ewa_lanczos_radius3_srgb** and **ewa_lanczos_radius3_linear** is obtained by replacing "`Triangle`" by "`LanczosRadius`" in §4.8.4.1. Three lobes is the default.

EWA LanczosRadius4 is obtained from EWA Lanczos 4 by shrinking its support so that it is a disk of radius exactly 4. Upsampling with **ewa_lanczos_radius4_srgb** and **ewa_lanczos_radius4_linear** is obtained by replacing "`Triangle`" by "`LanczosRadius -define filter:lobes=4` in §4.8.4.1.

4.8.6.6 EWA LanczosNSharp (Jinc-windowed Jinc N-lobe with support shrunk to preserve vertical and horizontal lines)

There is a filter called "Lanczos2Sharp" in ImageMagick, derived by applying methods similar to those used to derive RobidouxSharp, but with respect to preservation of vertical and horizontal lines, as was the case for the Robidoux filter. The built-in ImageMagick method (currently) has a slightly different blur factor than the one recommended now by N. Robidoux: The built-in method uses blur = 0.9549963639785485, and the current recommendation is 0.9580278036312191. Upsampling with

ewa_lanczos2sharp_srgb and **ewa_lanczos2sharp_linear** is obtained by replacing "`Triangle`" by "`Lanczos2 -define filter:blur=.9580278036312191`" in §4.8.4.1.

Analogous to Lanczos2Sharp, there is a "LanczosSharp" built into ImageMagick. The built-in ImageMagick method (currently) has a slightly different blur factor than the one currently recommended by N. Robidoux, who revised the objective function used to optimize the blur parameter: The built-in method uses blur = 0.9812505644269356, and the value used in the test suite is 0.9891028367558475 [110]. Upsampling with **ewa_lanczos3sharp_srgb** and **ewa_lanczos3sharp_linear** is obtained by replacing "`Triangle`" by "`Lanczos -define filter:blur=.9891028367558475`" in §4.8.4.1.

Upsampling with **ewa_lanczos4sharp_srgb** and **ewa_lanczos4sharp_linear** is obtained by replacing "`Triangle`" by "`Lanczos -define filter:lobes=4 -define filter:blur=.9870395083298263`" in §4.8.4.1.

4.8.6.7 EWA LanczosNSharpest (Jinc-windowed Jinc N-lobe with support shrunk to make it almost interpolatory)

EWA Lanczos2Sharpest is derived by applying methods similar to those used to derive RobidouxSharp. Specifically, Lanczos2Sharpest uses a deblur which minimizes the deviation from being interpolatory in a minimax sense, in the context of pixel values bound to a finite range [111]. Upsampling with **ewa_lanczos2sharpest_srgb** and **ewa_lanczos2sharpest_linear** is obtained by replacing "`Triangle`" by "`Lanczos2 -define filter:blur=0.8826421508540347`" in §4.8.4.1.

Lanczos3Sharpest is the 3-lobe version of Lanczos2Sharpest. Upsampling with **ewa_lanczos3sharpest_srgb** and **ewa_lanczos3sharpest_linear** is obtained by replacing "`Triangle`" by

"`Lanczos -define filter:blur=0.88549061701764`" in §4.8.4.1.

Lanczos4Sharpest is the 4-lobe version of Lanczos2Sharpest. Upsampling with **ewa_lanczos4sharpest_srgb** and **ewa_lanczos4sharpest_linear** is obtained by replacing "`Triangle`" by "`Lanczos -define filter:lobes=4 -define filter:blur=0.8845100233858514`1" in §4.8.4.1.

4.8.7 "Extra" ("external") upsampling method tested

Only one non-ImageMagick resampling method, namely Nohalo-LBB, is tested in the standard version EXQUIRES. It is included primarily as an example of hooking up external resampling programs to the test suite.

4.8.7.1 Nohalo with Locally Bounded Bicubic (LBB) finishing scheme

A wrapper, which replicates the top row and left column before resampling and crops afterwards, is required in order to ensure that the same image geometry convention is used by Nohalo-LBB and the ImageMagick methods. This C++ wrapper is included in the EXQUIRES distribution, but it is not installed by default. See Appendix B for the code listing.

Nohalo-LBB is a nonlinear method with two components: The first stage of Nohalo-LBB, namely Nohalo, is a nonlinear face split subdivision method described in [16, 32] and originally formulated by N. Robidoux with the assistance of several collaborators, including the author of this thesis. The second stage, LBB, which stands for Locally Bounded Bicubic, is a nonlinear interpolation scheme formulated by N. Robidoux based on a method of Brodlie et al. [112]. Although it is briefly discussed in [32], the best documentation about LBB is found in the VIPS [113] and GEGL source code listings [101] for the Nohalo-LBB method. (The GEGL method, under the name NoHalo, is called through GIMP (GNU Image, Manipulation Program) menus related to resampling.)

4.9 Image difference metrics

An image difference metric takes two images as inputs and returns a number that gives an indication of how similar or different the images are. In the context of this thesis, image difference metrics are used to measure how accurately an upsampler managed to reconstruct the original image from a downsampled version.

Throughout this chapter, it is assumed that two $M \times N$ sRGB images, named x and y, are compared. Since one of the two is considered as being the "reference image", the norm of the difference is often called the "error". The indices i and j represent the row and column of the pixel being compared, and the index k represents the colour channel being compared (of which there are three).

In addition, some of the metrics discussed in this chapter are variants of one another. In order to ensure that there is no misunderstanding, the variants of the various metrics are defined separately.

4.9.1 Metrics based on sRGB differences

For these error norms, the pixel values of the sRGB images x and y are taken literally.

sRGB ℓ_1 error

The ℓ_1 error is also known as the Average Absolute Error (AAE). It is given by

$$\mathrm{srgb}_1(x, y) = \frac{1}{3MN} \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{k=1}^{3} |x_{i,j,k} - y_{i,j,k}| \,.$$

sRGB ℓ_2 error

The ℓ_2 error is also known as Root Mean Squared Error (RMSE). It is given by

$$\mathrm{srgb}_2(x, y) = \sqrt{\frac{1}{3MN} \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{k=1}^{3} (x_{i,j,k} - y_{i,j,k})^2}.$$

sRGB ℓ_4 error

The ℓ_4 error norm has rarely, if ever, been used in image processing. It gives more weight to extreme differences than the earlier error norms, without singling out the largest one. It is given by

$$\mathrm{srgb}_4(x,y) = \sqrt[4]{\frac{1}{3MN}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{3}(x_{i,j,k}-y_{i,j,k})^4}.$$

sRGB ℓ_∞ error

The ℓ_∞ error is also known as the Maximum Absolute Error (MAE). It is given by

$$\mathrm{srgb}_\infty(x,y) = \max_{1\le i\le N, 1\le j\le M, 1\le k\le 3} |x_{i,j,k}-y_{i,j,k}|\,.$$

Mean SSIM (Structural SIMilarity) index

We have performed extensive refactoring of the computation of the standard SSIM index. For this reason, the discussion of the image difference metric is postponed to the next chapter.

Blurred sRGB ℓ_1, ℓ_2, ℓ_4 and ℓ_∞ errors

Blurring an image involves averaging pixels locally, for example, to remove noise or to smooth out the image. The "blurred" metrics blur_1, blur_2, blur_4 and blur_∞ are obtained by applying the corresponding ℓ_p metrics to the results of converting to luma (specifically, sRGB grayscale, a pixel by pixel weighted average of the three sRGB colour channels unconverted to linear light), blurring the input images, and cropping exactly like when computing the mean SSIM index. The reason for the inclusion of such nonstandard metrics into the suite is to estimate how much of the mean SSIM index ranking comes from the steps other than the "product of cross correlations" term.

Note that future versions of EXQUIRES are likely to base the blur_p metrics on the XYZ colour space, and omit cropping, only keeping SSIM's Gaussian blur.

4.9.2 Metrics based on XYZ differences

Created by the International Commission on Illumination (known as the CIE, which stands for "Commission Internationale de l'Éclairage") in 1931, the XYZ colour space is a linear light (meaning physical) colour space that is a good model of the colours that are perceptible by the human visual system.

For these error norms, the pixel values of the sRGB images x and y are converted to linear light, specifically the XYZ colour space, the resulting linear light images being written $\hat{x}$ and $\hat{y}$. The above formulas are then used on these linear light images.

XYZ ℓ_1 error

The ℓ_1-norm of the difference between the two images in the XYZ colour space is

$$\text{xyz}_1(x, y) = \frac{1}{3MN} \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{k=1}^{3} |\hat{x}_{i,j,k} - \hat{y}_{i,j,k}| \,.$$

XYZ ℓ_2 error

The ℓ_2-norm of the difference in the XYZ colour space is

$$\text{xyz}_2(x, y) = \sqrt{\frac{1}{3MN} \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{k=1}^{3} (\hat{x}_{i,j,k} - \hat{y}_{i,j,k})^2}.$$

XYZ ℓ_4 error

The ℓ_4-norm of the difference in the XYZ colour space is

$$\text{xyz}_4(x, y) = \sqrt[4]{\frac{1}{3MN} \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{k=1}^{3} (\hat{x}_{i,j,k} - \hat{y}_{i,j,k})^4}.$$

XYZ ℓ_∞ error

The ℓ_∞-norm of the difference in the XYZ colour space is

$$\mathrm{xyz}_\infty(x, y) = \max_{1 \le i \le N, 1 \le j \le M, 1 \le k \le 3} |\hat{x}_{i,j,k} - \hat{y}_{i,j,k}| .$$

4.9.3 Metrics based on CMC 1:1 ΔE distances

ΔE is a common notation for the distance between two colours within a so-called perceptually uniform colour space. Often used for colour tolerancing, perceptually uniform colour spaces are generally constructed so that there is a distance τ (the "threshold of perceptibility") such that if

$$\Delta\mathrm{E}(\mathrm{colour}_1, \mathrm{colour}_2) < \tau,$$

the two colours are generally indistinguishable, and if

$$\Delta\mathrm{E}(\mathrm{colour}_1, \mathrm{colour}_2) > \tau,$$

they can be perceived as being different.

The CMC l:c quasimetrics were formulated for the purpose of colour tolerancing by the Colour Measurement Committee of the Society of Dyers and Colourists [114, 115]. These quasimetrics approximate the magnitude of the perceived deviation from a reference colour. Based on the CIE L*C*h (Lightness, Chroma and Hue) colour space, which itself is based on the CIE L*a*b* (very approximately) perceptually uniform colour space [73], the CMC l:c quasimetrics are defined by fairly complex formulas that result from modeling the Human Visual System [114, 116]. The l and c parameters are used to change the weighting of the lightness ("l") and chroma ("c") components relative to hue. CMC 2:1 ("two units of lightness error are equivalent to one unit of hue error"), for example, gives less weight to lightness deviations than CMC 1:1 ("one unit of lightness error for each unit of hue error").

The CMC l:c quasimetrics can be symmetrized by exchanging the roles of the modified colour and the reference colour and averaging the two results [117]. We use $\Delta\mathrm{E}_{\mathrm{cmc}}$, the

symmetrized version of CMC 1:1 with $l = h = 1$, values recommended for threshold of perceptibility measurements [115].

CMC 2:1 is recommended for graphics work by Steve Upton [118]. It is however not clear whether CMC 2:1 leads to a better image difference metric. One may also question whether symmetrizing the metric is the best option when there is a golden standard to compare to, as is the case in EXQUIRES. In any case, given that CMC 2:1 was not available through the VIPS library when EXQUIRES was being developed (it is now [119]), we use the built-in symmetrized CMC 1:1 VIPS implementation.

Within the image processing library VIPS, CMC 1:1 colour differences are computed by taking the Euclidean distance after conversion to a custom colour space constructed from L*C*h for this purpose [116, 119]. In terms of coordinates in this custom colour space constructed from scratch by Dr. John Cupitt based on the original quasimetric formulas, the formulas for the various image difference metrics used in EXQUIRES would look just like the previous ones (except cmc_∞) [120, 121]. We prefer, however, to make the connection to the CMC colour distance explicit.

In the following, the L*C*h images derived from the x and y images being compared are written $\tilde{x}$ and $\tilde{y}$.

CMC 1:1 ℓ_1 error

The ℓ_1-norm based on the ΔE CMC 1:1 colour distance is

$$\mathrm{cmc}_1(x, y) = \frac{1}{MN} \sum_{i=1}^{N} \sum_{j=1}^{M} \Delta \mathrm{E}_{\mathrm{cmc}}(\tilde{x}_{i,j}, \tilde{y}_{i,j}).$$

CMC 1:1 ℓ_2 error

The ℓ_2-norm based on the ΔE CMC 1:1 colour distance is

$$\mathrm{cmc}_2(x,y) = \sqrt{\frac{1}{MN}\sum_{i=1}^{N}\sum_{j=1}^{M}(\Delta\mathrm{E}_{\mathrm{cmc}}(\tilde{x}_{i,j},\tilde{y}_{i,j}))^2}.$$

CMC 1:1 ℓ_4 error

The ℓ_4-norm based on the ΔE CMC 1:1 colour distance is

$$\mathrm{cmc}_4(x,y) = \sqrt[4]{\frac{1}{MN}\sum_{i=1}^{N}\sum_{j=1}^{M}(\Delta\mathrm{E}_{\mathrm{cmc}}(\tilde{x}_{i,j},\tilde{y}_{i,j}))^4}.$$

CMC 1:1 ℓ_∞ error

The ℓ_∞-norm based on the ΔE CMC 1:1 colour distance is

$$\mathrm{cmc}_\infty(x,y) = \max_{1\le i\le N, 1\le j\le M}\Delta\mathrm{E}_{\mathrm{cmc}}(\tilde{x}_{i,j},\tilde{y}_{i,j}).$$

5 Implementation

5.1 Streamlined computation of the SSIM (Structural SIMilarity) index

The SSIM (Structural SIMilarity) index, introduced in 2003 by Wang et al. [59], is a widely used image comparison method producing a one-channel pseudo-image indicative of local differences, the SSIM index map, from which an overall similarity score, the mean SSIM index, is derived.

Early versions of the test suite used to compute the results discussed by Robidoux et al. in [14, 16] were unfriendly to use because of the large runtimes of the computation of the SSIM index. For this reason, considerable effort has been devoted to streamlining its computation. The standard computation of the SSIM index involves five Gaussian blurs and 17 in-place floating-point operations (flops) per pixel. (Note that we count multiply-add as one flop.) In this chapter, the linearity of convolution and algebraic refactoring are used to reduce the flop count by 20%. We also reduce the memory footprint of the computation. Informal Matlab and VIPS (Virtual Image Processing System) benchmarking confirmed the benefits.

5.1.1 Mathematical specification of SSIM

This section summarizes the most commonly used formulation of the SSIM Structural SIMilarity) index map (a one channel pseudo-image with pixel values in $(-1, 1]$) and the

MSSIM (Mean SSIM) index (a real number in the same interval) [59].

When dealing with colour images, each image is first converted to grayscale luma by taking a linear combination of sRGB values. (Although the original specification of SSIM [59] suggests that luminance—that is, a linear combination of linear light colours—should be used to compute the grayscale versions of the images to be compared, it appears that luma is now standard [122].) The following description assumes that the transformation of the original colour images to luma (or luminance) has already been performed.

Let $\mathbf{x}$ and $\mathbf{y}$ be the two $W \times H$ grayscale images to be compared, with pixel values in the interval $[0, L]$. The SSIM index map $\mathbf{s}$, a centred grayscale image with slightly reduced dimensions $(W-10) \times (H-10)$, has pixel values given by

$$\mathbf{s} = \frac{\left(2\boldsymbol{\mu}_\mathbf{x}\boldsymbol{\mu}_\mathbf{y} + c_1\right)\left(2\boldsymbol{\sigma}_{\mathbf{xy}} + c_2\right)}{\left(\boldsymbol{\mu}_\mathbf{x}^2 + \boldsymbol{\mu}_\mathbf{y}^2 + c_1\right)\left(\boldsymbol{\sigma}_\mathbf{x}^2 + \boldsymbol{\sigma}_\mathbf{y}^2 + c_2\right)}, \tag{5.1}$$

where arithmetic is performed pixel by pixel. The centred $(W-10) \times (H-10)$ images appearing on the right-hand side of (5.1) are

$$\begin{aligned}
\boldsymbol{\mu}_\mathbf{x} &= \mathbf{E}\left(\mathbf{x}\right), \\
\boldsymbol{\mu}_\mathbf{y} &= \mathbf{E}\left(\mathbf{y}\right), \\
\boldsymbol{\sigma}_{\mathbf{xy}} &= \mathbf{E}\left((\mathbf{x} - \boldsymbol{\mu}_\mathbf{x})(\mathbf{y} - \boldsymbol{\mu}_\mathbf{y})\right), \\
\boldsymbol{\sigma}_\mathbf{x}^2 &= \mathbf{E}\left((\mathbf{x} - \boldsymbol{\mu}_\mathbf{x})^2\right), \\
\boldsymbol{\sigma}_\mathbf{y}^2 &= \mathbf{E}\left((\mathbf{y} - \boldsymbol{\mu}_\mathbf{y})^2\right)
\end{aligned}$$

in terms of the local averaging operator $\mathbf{E}$, which blurs using an 11×11 Gaussian convolution mask with $\sigma = 1.5$ so that, at pixel location (i, j),

$$\left(\mathbf{E}(\mathbf{w})\right)(i,j) = \frac{\sum_{k,l=-5}^{5} \exp\left(-\frac{k^2+l^2}{2\sigma^2}\right) \mathbf{w}(i+k, j+l)}{\sum_{k,l=-5}^{5} \exp\left(-\frac{k^2+l^2}{2\sigma^2}\right)}.$$

(Note that the Gaussian blur σ is a parameter of the blur operator, otherwise separate from the $\boldsymbol{\sigma}$s.) Division by zero is prevented by setting

$$c_1 = (K_1 L)^2 = (0.01L)^2 \text{ and } c_2 = (K_2 L)^2 = (0.03L)^2.$$

The mean SSIM index is the average of the SSIM index map s. It is a single number in the interval $(-1, 1]$.

Algorithm 1 Refactored SSIM index map computation for Matlab (read-only input images)

{The read-only 2D arrays x and y contain grayscale input images.}	
$\mathbf{s} \leftarrow \mathbf{E}(\mathbf{x})$	{blur}
$\mathbf{t} \leftarrow \mathbf{E}(\mathbf{y})$	{blur}
$\mathbf{u} \leftarrow 2\mathbf{st} + c_1$	{2 × and +}
$\mathbf{t} \leftarrow \mathbf{t} - \mathbf{s}$	{−}
$\mathbf{s} \leftarrow 2\mathbf{E}(\mathbf{xy})$	{(scaled) blur and ×}
$\mathbf{s} \leftarrow (\mathbf{s} + (c_1 + c_2) - \mathbf{u})\,\mathbf{u}$	{×, − and +}
$\mathbf{t} \leftarrow \mathbf{t}^2 + \mathbf{u}$	{× and +}
$\mathbf{u} \leftarrow \mathbf{E}(\mathbf{x}^2 + \mathbf{y}^2)$	{blur, 2 × and +}
$\mathbf{s} \leftarrow \mathbf{s} \div ((\mathbf{u} + (c_1 + c_2) - \mathbf{t})\,\mathbf{t})$	{÷, ×, − and +}

5.1.2 Refactored computation for Matlab

Most SSIM implementations are direct translations of the proof-of-concept Matlab code written by Zhou Wang [66]. There are two versions of the Matlab code: one that uses automatic downsampling and one that does not [123, 124].

Nicolas Robidoux refactored the computation of the SSIM map three different ways, and the author of this thesis checked that the implementations clone the original within round off [125].

The first refactoring is shown in Algorithm 1. The linearity of the averaging operator $\mathbf{E}$ allows one to compute $\sigma_{\mathbf{x}}^2 + \sigma_{\mathbf{y}}^2$ with one single convolution (or two one-dimensional convolutions, if the separability of Gaussian blur is taken advantage of). This reduces the total number of blurred intermediate images down to four.

One can fold multiplicative constants into the blur masks, which makes them free.

Without overwriting the input images, one can rearrange the computation so that only three $(W-10)\times(H-10)$ and three $W\times H$ images are sufficient to store the inputs as well as both intermediate and final results. In Algorithm 1, the three smaller images are called $\mathbf{s}$, $\mathbf{t}$ and $\mathbf{u}$, and the extra larger image is defined implicitly by the arguments of the blur operator $\mathbf{E}$.

The per-pixel flop count of the refactored computation of the SSIM index map is four times the cost of performing (scaled) Gaussian blur, plus 17 flops. Given that, if the separability of the Gaussian kernel is exploited, the cost of each blur is 20 multiply-adds and 2 multiplications per pixel with SSIM's standard 11×11 Gaussian blur mask, the grand total is 106 flops per pixel. (Without separability, the formal count is $4(121)+17 = 501$ flops per pixel.)

In the remainder of this section, we compare the refactored version to Z. Wang's original.

The memory footprint of the original Matlab code corresponds to at least nine $(W-10)\times(H-10)$ and three $W\times H$ images ("at least" nine because additional intermediate result storage is triggered). Refactoring thus reduces the memory footprint of the computation by at least 50%.

The original code uses five blur operations (one more than the refactored code) plus 20 flops per pixel (three more than the refactored code) so that the original code has 20% more flops than the refactored one.

5.1.3 SSIM index map computation with input overwriting

If the input images are overwritten, three $(W-10)\times(H-10)$ and two $W\times H$ images are sufficient, the key being the computation of $\boldsymbol{\sigma}_{\mathbf{xy}}$ and $\boldsymbol{\sigma}_{\mathbf{x}}^2+\boldsymbol{\sigma}_{\mathbf{y}}^2$ from $\mathbf{E}((\mathbf{x}\pm\mathbf{y})^2)$. See Algorithm 2, in which every line corresponds to one Matlab assignment. This version is implemented in the code shown in Appendix C.

Algorithm 2 Refactored SSIM index map computation with input overwriting for Matlab

{The 2D arrays $\mathbf{x}$ and $\mathbf{y}$ initially contain grayscale input images.}	
$\mathbf{x} \leftarrow \mathbf{x} - \mathbf{y}$	$\{-\}$
$\mathbf{y} \leftarrow 2\mathbf{y} + \mathbf{x}$	$\{\times \text{ and } +\}$
$\mathbf{t} \leftarrow \frac{1}{\sqrt{2}}\mathbf{E}(\mathbf{x})$	{(scaled) blur}
$\mathbf{x} \leftarrow \mathbf{x}^2$	$\{\times\}$
$\mathbf{x} \leftarrow \frac{1}{2}\mathbf{E}(\mathbf{x})$	{(scaled) blur}
$\mathbf{s} \leftarrow \frac{1}{\sqrt{2}}\mathbf{E}(\mathbf{y})$	{(scaled) blur}
$\mathbf{y} \leftarrow \mathbf{y}^2$	$\{\times\}$
$\mathbf{y} \leftarrow \frac{1}{2}\mathbf{E}(\mathbf{y})$	{(scaled) blur}
$\mathbf{y} \leftarrow \mathbf{y} - \mathbf{x}$	$\{-\}$
$\mathbf{t} \leftarrow \mathbf{t}^2$	$\{\times\}$
$\mathbf{s} \leftarrow \mathbf{s}^2 + (-c_1) - \mathbf{t}$	$\{\times, - \text{ and } +\}$
$\mathbf{t} \leftarrow \mathbf{t} - \mathbf{s}$	$\{-\}$
$\mathbf{s} \leftarrow ((\mathbf{y} + (c_1 + c_2) + \mathbf{s})\,\mathbf{s}) \div ((\mathbf{y} + (c_1 + c_2) + \mathbf{x} - \mathbf{t})\,\mathbf{t})$	$\{\div, 2\times, 2 - \text{ and } 3 +\}$

5.1.4 Threaded local demand-driven pipelined computation for VIPS

In order to access the pipelining power of the VIPS library [43], the entire MSSIM computation is rewritten in Static Single Assignment (SSA) form as shown in Algorithm 3. This pseudocode was written to take advantage of VIPS features accessible through C implementations. Without being an exact translation, EXQUIRES' Python implementation, shown in §A.3, is based on it.

Algorithm 3 Pipelined SSIM index map computation for VIPS

{The 2D arrays $\mathbf{x}$ and $\mathbf{y}$ contain grayscale input images.}

$\mathbf{a} \leftarrow \mathbf{E}(\mathbf{x})$	{blur}
$\mathbf{b} \leftarrow \mathbf{E}(\mathbf{y})$	{blur}
$\mathbf{c} \leftarrow \mathbf{xy}$	$\{\times\}$
$\mathbf{d} \leftarrow 2\mathbf{E}(\mathbf{c}) + (c_1 + c_2)$	{(scaled) blur-add}
$\mathbf{e} \leftarrow \mathbf{x}^2$	$\{\times\}$
$\mathbf{f} \leftarrow \mathbf{y}^2$	$\{\times\}$
$\mathbf{g} \leftarrow \mathbf{e} + \mathbf{f}$	$\{+\}$
$\mathbf{h} \leftarrow \mathbf{E}(\mathbf{g}) + (c_1 + c_2)$	{blur-add}
$\mathbf{i} \leftarrow \mathbf{ab}$	$\{\times\}$
$\mathbf{j} \leftarrow \mathbf{b} - \mathbf{a}$	$\{-\}$
$\mathbf{k} \leftarrow 2\mathbf{i} + c_1$	{multiply-add}
$\mathbf{l} \leftarrow \mathbf{d} - \mathbf{k}$	$\{-\}$
$\mathbf{m} \leftarrow \mathbf{j}^2$	$\{\times\}$
$\mathbf{n} \leftarrow \mathbf{lk}$	$\{\times\}$
$\mathbf{o} \leftarrow \mathbf{m} + \mathbf{k}$	$\{+\}$
$\mathbf{p} \leftarrow \mathbf{h} - \mathbf{o}$	$\{-\}$
$\mathbf{q} \leftarrow \mathbf{po}$	$\{\times\}$
$\mathbf{s} \leftarrow \mathbf{n} \div \mathbf{q}$	$\{\div\}$

5.2 Technical overview of the EXQUIRES test suite

This chapter briefly describes the implementation of the EXQUIRES test suite. It complements the User Manual found in Appendix D and the complete source code found in Appendices A and B, which are based on release version 0.9.9.3.

5.2.1 Reading and writing configuration files

The EXQUIRES test suite can be customized by the user through the use of a configuration file. The reading and writing of configuration files is handled by the `configobj` [126] Python module (specifically the `ConfigObj` class), which enforces a format similar to Windows `.ini` files. Running `exquires-new` produces a configuration file, defaulting to `project1.ini` if a project name is not provided. This file defines a collection of images, downsamplers, resampling ratios, upsamplers, and comparison metrics to use with `exquires-run` and `exquires-update`.

Each section of the configuration file contains a list of key-value pairs of the form `key = value`. For the `[Ratios]` section, `key` is the factor by which to reduce and re-enlarge the images, and `value` is the size of the reduced image (based on an initial width and height of 840 pixels).

For all other sections, `key` is a unique name to reference the `value`. For the `[Images]` section, `value` is the absolute path of the image. For the `[Downsamplers]` and `[Upsamplers]` sections, `value` is the command to resize an image using a particular resampler. These commands must make use of Python replacement fields, representing the paths of the input and output images, the resampling ratio, and the reduced or enlarged image size, respectively.

The `[Metrics]` section is slightly different, since `value` is a comma-separated list of three elements: the command to call the metric (making use of replacement fields that represent the paths of the two images to compare), the command to aggregate a list of

results produced by the metric, and either 0 or 1 to represent the best-to-worst order (ascending and descending, respectively).

When adding programs to the configuration file, users can make direct use of any program located in a directory specified in the system path, or provide the full path to the program. In either case, the commands must make use of the aforementioned replacement fields. As `exquires-run` and `exquires-update` loop through the entries in the configuration file, the replacement fields will be substituted with the values for the current iteration.

Currently, the configuration files produced by `exquires-new` include both linear and sRGB resamplers. A future version will include option flags to specify the desired colour space for upsamplers and downsamplers, since one may wish to investigate the effects of downsampling using a particular class and upsampling using another particular class.

5.2.2 Generating and reporting image comparison data

The EXQUIRES test suite comes with two programs to generate image comparison data: `exquires-run`, which creates a new database file for the specified configuration, and `exquires-update`, which will modify an existing database file when the configuration is changed by the user. Also included are two programs to produce statistics from this data: `exquires-report`, which produces tables of aggregate rankings for each metric and merged ranks across a set of metrics, and `exquires-correlate`, which produces a Spearman cross-correlation matrix for the specified group of images, downsamplers, ratios, or metrics.

Resampling images

As discussed in §5.2.1, calling `exquires-new` generates a configuration file. The default entries in the `[Downsamplers]` and `[Upsamplers]` sections correspond to the down-

samplers and upsamplers described in §4.7 and §4.8, respectively. These are ImageMagick [42] commands that are executed by `exquires-run` and `exquires-update`.

In order to reduce the storage footprint when calling `exquires-run` or `exquires-update`, the downsampled and upsampled images are deleted once they are no longer required for computing comparison data. The main disadvantage of this approach is that certain resampling tasks will be re-executed if `exquires-update` is called after the `[Upsamplers]` or `[Metrics]` sections of the configuration file are modified. However, modifying any other section of the configuration file will not result in any redundant computation.

Comparing images

Once the downsampled images have been re-enlarged, the upsampled images can be compared to the originals with user-provided image comparison metrics, or those defined in the `Metrics` class, which computes metrics by driving VIPS [43] code. In order to provide a framework for calling user-provided metrics, the default metrics are executed by calling `exquires-compare` rather than directly accessing the methods in the `Metrics` class. Since the output of a metric must be inserted into a database, `subprocess.check_output` is used to access the value returned by the metric.

Optionally, the user may directly compare a pair of images by running `exquires-compare` and specifying the metric and paths to the images.

Reading and writing database tables

The reading and writing of database tables relies on `sqlite3` [127]. Each table represents a particular combination of test image, downsampler, and resampling ratio. Each row represents a particular upsampler, with the first column containing the name of the upsampler and each subsequent column containing the comparison data for a particular metric. Since

using the table names to determine the image, downsampler, and ratio could lead to errors, an additional table called `TABLEDATA` is used to associate each table with these pieces of information.

Unfortunately, `sqlite3` does not support the full `ALTER TABLE` syntax [128]. In particular, while it can be used to add columns to an existing table, it is incapable of removing columns. Thus, whenever an existing table must be modified by `exquires-update`, a new table must be created with the same name, but with a modified set of columns.

First, the table is renamed using the `backup_table` method of the `Database` class and a new table is created with the old name and the columns specified in the current configuration file. Then, for each upsampler, the relevant columns from the backup table are returned as a dictionary, to which the results of calling any new metrics are added before being inserted into the new table. Finally, the backup table is removed using the `drop_backup` method.

Once a suitable configuration file has been established, running `exquires-run` will perform all the downsampling, upsampling, and comparison steps.

The `operations` module has been written to ensure that the optimal number of downsampling, upsampling, and comparison operations are performed when `exquires-update` is used to modify an existing database.

Aggregating image comparison data

When the user calls `exquires-report` or `exquires-correlate`, the image comparison data contained in the database must be aggregated to produce the requested ranks or Spearman cross-correlations. These aggregation methods can be called manually using `exquires-aggregate` on any list of numbers. Several of the methods perform exponentiation and averaging on the list. In order to ensure that these operations are performed efficiently, the NumPy numerical Python library [129] is used, which provides basic array

operations.

5.2.3 Parsing command line arguments

All EXQUIRES programs make use of the `argparse` module [130] to handle command line arguments. More specifically, the `argparse.ArgumentParser` class is extended by `ExquiresParser`, which is further extended by `OperationsParser` and `StatsParser`.

Expanding wildcard characters

Wildcard characters are permitted when specifying images, downsamplers, upsamplers, and metrics for `exquires-report` and `exquires-correlate`. These arguments are handled by `ListAction`, a customized `argparse.Action` class included with EXQUIRES. This class uses the `fnmatch` module [131] to match the patterns against the most recently used configuration. If any argument containing a wildcard cannot be expanded to match an entry in the configuration file, an `argparse.ArgumentError` is raised to inform the user of the invalid argument.

Note that a backslash must be included when attempting to specify all valid arguments (`\*`), otherwise the shell will attempt to match filenames in the current directory instead of entries in the configuration file.

Expanding numeric ranges

Hyphenated ranges are permitted when specifying resampling ratios for `exquires-report` and `exquires-correlate`. This option is handled by `RatioAction`, a customized `argparse.Action` class included with EXQUIRES that is used by the `StatsParser` class (a customized `ExquiresParser`). If a hyphen is located in one of the ratio arguments, the argument is expanded into the

specified range of integers. If any of the integers are not found in the configuration file, an `argparse.ArgumentError` is raised to inform the user of the invalid argument.

Handling table sorting and correlation matrix anchors

When using `exquires-report` to produce a table of aggregate image comparison data, it is possible to specify a metric that will determine the row ordering (based on its best-to-worst convention as defined in the configuration file). This is handled by `SortAction`, a customized `argparse.Action` class included with EXQUIRES that is used by the `StatsParser` class.

Similarly, when using `exquires-correlate` to produce a Spearman cross-correlation matrix, it is possible to specify a row/column name that will be designated as the topmost row and leftmost column. All remaining rows and columns will be sorted according to the specified row/column. This is handled by `AnchorAction`, a customized `argparse.Action` class included with EXQUIRES that is used by the `StatsParser` class.

Introspection

For the programs `exquires-aggregate` and `exquires-compare`, the `inspect` module is used to return a list of the public methods defined in the `Aggregate` and `Metrics` classes, respectively. This ensures that each program will only accept `METHOD` arguments that match the name of one of these public methods. If the user provides an invalid argument, `ExquiresParser` will produce an error that lists the valid options.

5.2.4 Code quality

The `pep8` code style checker [132], which enforces the style conventions described in the PEP 8 style guide for Python code [133], is used as a first pass to identify any obvious style

errors. This is accomplished by running `pep8 *.py` in the EXQUIRES package folder.

Once the package clears the first pass, `pylint` [134] is used to identify any source code errors as well as good targets for refactoring. In addition, `pylint` enforces a style convention that is more comprehensive than `pep8`. As a result, it played a major role in developing the current structure of EXQUIRES. Since version 0.9.8, running `pylint exquires` yields a perfect score of 10.0.

5.2.5 Documentation

The EXQUIRES documentation is written in reStructuredText markup language [135], which is part of the Docutils text processing system [136]. In addition to the `.rst` files that define the documentation, the docstrings within each Python module are written in reStructuredText. The docstrings are also used to display command line help messages, so the reStructuredText markup must first be stripped. This is accomplished by `ExquiresParser`, a customized `argparse.ArgumentParser` class that formats the docstrings using the regular expression module `re`. More specifically, `re.sub` is used to match certain patterns in the docstrings and replace them with alternatives that are more suited for outputting to the terminal.

The documentation can be built using Sphinx [137], a popular Python documentation generator. Using the included `Makefile`, it is possible to generate the documentation in several formats, most notably PDF and HTML. The PDF user manual, which provides an extensive overview of EXQUIRES version 0.9.9.3, is found in Appendix D. In addition, the HTML-formatted documentation of the latest version of EXQUIRES is viewable online at `exquires.ca`.

6 Results

6.1 Major caveat regarding upsampler rankings

Despite their ubiquity (see the Introduction and Previous Work chapters), the upsampler rankings produced by image re-enlargement tests should be taken with a very large grain of salt, if only because they generally favour strongly sharpening methods at the expense of methods that introduce fewer unpleasant artifacts, at least when filters with decent low pass filtering are used to downsample. Many authors, in fact, stress the importance of visual inspection, despite its subjectivity, in part because of the limitations of common image metrics to measure perceptual differences [12, 16, 30, 33, 58, 59, 138, 139]. For example, Mitchell-Netravali cubic spline smoothing ranks very poorly overall in re-enlargement tests which include low pass downsamplers, despite it striking a good balance between sharpness and lack of artifacts, according to an early panel of experts [58]. In addition, Mitchell-Netravali cubic spline smoothing has been the default ImageMagick filter for image resizing [93] for a long time, and its many users are generally satisfied with this choice. In any case, it is probably safer to compare ranks for methods known to have similar visual artifacts than to compare ranks for methods with completely different visual character.

The grain of salt may be better described as salt shaker when one takes into account the considerable impact of several factors on the rankings [11, 16]. For example, we will see that Mitchell-Netravali earns the top rank when downsampling is performed with nearest neighbour decimation, a very poor low pass filter, even though all test images are de-noised.

Also keep in mind that the results of re-enlargement tests are only informative with

respect to applications in which no significant downsampling is performed: For example, the rankings shown in this thesis provide no guidance with respect to the choice of a method for thumbnail production.

6.2 Overall upsampler ranks (linear light toolchain and RMSE)

In this section, the reader will find three sets of rankings of sixty four image enlargement methods. The first set could reasonably be construed to be representative of the overall quality of a filter when used to enlarge high quality natural images. The additional two sets of rankings are obtained by considering only part of the data: The second set only uses the results of re-enlarging images obtained by downsampling with an effective low pass filter, and the third set only uses the results of re-enlarging images obtained by downsampling with nearest neighbour. The "top five" methods within various categories are also discussed.

Even though EXQUIRES has the capability to automatically average the fractional ranks obtained by different metrics and use them to compute ranks representative of different image difference measures, the overall ranks reported in the remainder of this thesis only use xyz_2 errors, that is, aggregated RMSE errors in the XYZ linear light ("physical") colour space, *unless otherwise stated.* In particular, all the ranks reported in this section are based on XYZ RMSE. The main reason for only using one metric to derive overall ranks is that some of the tested metrics give nearly reversed rankings, and one can question the usefulness of aggregating rankings obtained with discordant criteria.

XYZ RMSE is arguably the most "physically neutral" metric considered in this thesis: XYZ is a linear light space, so that RMSE is correlated with the total light energy contained in the deviation from the gold standard image [73]. Even though XYZ is a linear (physical) colour space,

> The CIE X,Y,Z values contain information about the level of light absorption by each of the 3 types of cone photoreceptors,

as stated by Zhang and Wandell [140]. For this reason, it is a linear light colour space well suited to studies that concern the Human Visual System.

In addition, only results pertaining to linear light toolchains (downsampling and upsampling performed through linear RGB with sRGB primaries or XYZ) are considered for

the computation of overall ranks.

6.2.1 Overall ranks (all downsamplers included)

Overall ranks, based on all tested ratios, downsamplers and test images, are shown in Table 6.1 (tables are found at the end of this section). In this table, methods which introduce slope discontinuities when an "infinite" enlargement ratio is used (methods which are not "C^1", that is, not continuously differentiable) are indicated by italicizing their name. Note however that the gradient discontinuities introduced by the Kaiser and, to a lesser extent, Hamming filters are fairly small, to the extent of that one could deem them irrelevant even with large enlargement ratios. The same can be said of the gradient discontinuities introduced by the Bartlett, EWA teepee, and bilinear filters when the ratio is small. In any case, such discontinuities are only an issue when the enlargement ratio is large.

These ranks are derived from the aggregated XYZ RMSE errors shown in Table 6.4. (Ranks are computed before rounding.) Each xyz_2 error shown in Table 6.4 is obtained by aggregating the results of re-enlarging, with 7 ratios, the results of downsampling, with 5 downsamplers, 17 840×840 sRGB images. In other words, each error can be understood as the Euclidean distance between two 420-megapixel ($7 \times 5 \times 17 \times 840 \times 840$) colour images in the XYZ colour space.

The top five methods are

1. Nohalo with Locally Bounded Bicubic finishing scheme (Nohalo-LBB)
2. 4-lobe Sinc filtering with Hamming windowing (Hamming 4-lobe)
3. 3-lobe Sinc filtering with Cosine windowing (Cosine 3-lobe)
4. 4-lobe Sinc filtering with Hann windowing (Hann 4-lobe)
5. Lanczos 4-lobe (that is, 4-lobe Sinc filtering with Sinc windowing).

Because of the visual artifacts they introduce, Nicolas Robidoux generally does not recommend methods that fail to be C^1 for enlargement by a large factor [141]. Requiring

the filter to have a continuous gradient (to be "C^1") pushes Hamming out of the ranking:

1. Nohalo-LBB

3. Cosine 3-lobe

4. Hann 4-lobe

5. Lanczos 4-lobe

6. Lanczos 3-lobe.

Given that 4-lobe methods are often felt to introduce too much haloing—they "ring" three times—this suggests that Cosine-windowed Sinc 3 and Lanczos 3 are very good. Indeed, Lanczos 3 is extremely popular.

If one further narrows the field to C^1 methods with 3 lobes or less, that is, methods with at most one halo, the top methods are

1. Nohalo-LBB

3. Cosine 3

6. Lanczos 3

8. Welch 3

16. Hann 3.

If one only considers C^1 methods with 2 lobes or less, one gets

1. Nohalo-LBB

21. Welch 2

25. Cosine 2

31. EWA RobidouxSharp, an EWA method which uses a tuned Keys cubic spline

32. Lanczos 2, a fairly popular method.

It appears that the best (tensor) window function depends on the number of Sinc lobes. Cosine, Sinc (Lanczos), Welch and Hamming generally do well.

Tensor ("orthogonal") methods generally perform better than Elliptical Weighted Averaging methods. The top EWA methods, all C^1, are

12. EWA LanczosSharpest 4

17. EWA LanczosSharpest 3

23. EWA Lanczos Radius 3

26. EWA Lanczos Radius 4

31. EWA RobidouxSharp.

(Note that the only windowing function tested with the Jinc filter was Jinc.)

It is possible that the top rank of Nohalo-LBB owes something to the fact that it is the only tested method that resamples through the XYZ linear colour space instead of linear RGB with sRGB primaries. (Note however that it produces an sRGB enlargement from an sRGB input image, like the other methods.) In addition, it is the only tested nonlinear method, as well as the only method which is not tested with an ImageMagick implementation: The Nohalo-LBB program used for testing comes from the VIPS library. It is also possible, although very unlikely, that the top rank obtained by Nohalo-LBB comes from a different system used for colour space conversion, because the tested version uses a colour profile. At least in principle, colour profiles are less accurate than the formulaic (algebraic) approach used internally by ImageMagick. (VIPS now has the ability to use formulaic (algebraic) conversions from sRGB to XYZ, but this feature was not available in time when EXQUIRES was programmed.) It is also possible that Nohalo-LBB's top rank comes from the fact that it does not use look up tables, like the ImageMagick implementations used to test the other methods. In any case, it is the opinion of N. Robidoux that if the top rank is undeserved, it is most likely a side effect of using a different linear light colour space: XYZ instead of linear RGB with sRGB primaries, or a consequence of the choice of test images and downsampling methods [142].

Rankings do not seem to be affected by the smoothness of the window function: In Tables 6.1 and 6.2, italicized entries, which label methods that are not C^1, appear throughout. For example, Hamming-windowed Sinc 4-lobe, a C^0 but not C^1 method, is the top ranked linear method; Cosine-windowed Sinc 3-lobe, a C^1 method, is just behind.

Generally, 4-lobe methods perform better than 3-lobe methods, which themselves perform better than 2-lobe methods, which rank above methods which do not introduce haloing ("1-lobe methods"). A notable exception is Cosine-windowed Sinc 3-lobe, which actually ranks above every other C^1 method with the exception of Nohalo-LBB. Other exceptions are Welch-windowed Sinc 3 and all the EWA Lanczos methods except LanczosSharpest, for which the top ranked variant is the 4-lobe version. 5- and 6-lobe data would be interesting to add to the comparison. Nohalo-LBB, which is essentially halo-free, gets the top rank, even though it could be described as a 1-lobe method.

It does not make much difference whether one uses bilinear, quadratic B-spline or cubic B-spline for tensor (orthogonal) smoothing or EWA smoothing: With each of them, the EWA version ranks just above the tensor one. This is not the case for the Keys splines with negative lobes tested in both tensor and EWA versions, namely Mitchell-Netravali and Catmull-Rom: The rankings are quite different. (Note that the tested tensor and EWA Lanczos methods are the same only by name: tensor Lanczos refers to Sinc-windowed Sinc, EWA Lanczos to Jinc-windowed Jinc.)

The "sharp" version of the Kaiser window beats the other two versions of the Kaiser window for all numbers of lobes, putting into question the optimality of the choice of β values advocated by Theußl et al. [97]. It would be interesting to explore more choices of the Kaiser β parameter: It looks like smaller values than those tested may be better. Note however that it is possible that the Kaiser window suggested by Theußl et al. leads to less visually offensive artifacts. No visual comparison was performed for this thesis. Also keep in mind that rankings depend on the experimental set up. For example, the three variants of the Kaiser window function appear in different orders in rankings computed with some

of the other image difference metrics.

6.2.2 Overall ranks excluding the results of re-enlarging images obtained with nearest neighbour downsampling

In this section, we revisit the overall ranks and errors, only including, this time around, the results of re-enlarging images obtained by downsampling with a low-pass filter (box filtering, Gaussian blur, Lanczos 3-lobe and Elliptical Weighted Averaging Lanczos 3-lobe), leaving out the results of re-enlarging images obtained by downsampling with nearest neighbour (not a low-pass filter by any means).

One could also argue that the inclusion of re-enlargements of small images produced with nearest neighbour decimation adds balance to a set dominated by small images produced with rather strong low-pass filters. For this reason, we present overall ranks obtained both with (Tables 6.1 and 6.4) and without (Tables 6.2 and 6.5) nearest neighbour downsampling included. Note that in the following sections of the thesis, the results produced with all downsamplers are always included. The present subsection is the only one in which nearest neighbour downsampling results are excluded; the next subsection is the only one in which the other downsamplers are excluded.

Let us revisit the "overall group rankings". Methods that score within the top 5 irregardless of the inclusion or exclusion of nearest neighbour results are in boldface; those that rank in the bottom half when nearest neighbour results are included are in italics.

The top five methods are

1. Welch 4
2. Cosine 4
3. **Lanczos 4**
4. *EWA Catmull-Rom*
5. **Hamming 4**.

Although Welch 4 and Cosine 4 are not in the top five when nearest neighbour downsampling results are included, they are close: Welch 4 has rank 10, and Cosine 4 has rank 9. EWA Catmull-Rom, on the other hand, has rank 49 when nearest neighbour downsampling is included.

Requiring the filter to have a continuous gradient (to be "C^1"), one gets

1. Welch 4
2. Cosine 4
3. **Lanczos 4**
4. *EWA Catmull-Rom*
6. **Welch 3**.

If one further narrows the field to C^1 methods with 3 lobes or less, the top methods are

4. *EWA Catmull-Rom*
6. **Welch 3**
7. **Cosine 3**
10. **Lanczos 3**
12. **EWA LanczosSharpest 3**.

If one only considers C^1 methods with 2 lobes or less, that is, methods with at most one halo, one gets

4. *EWA Catmull-Rom*
21. **Nohalo-LBB**
23. **Welch 2**
26. **Cosine 2**
29. **EWA RobidouxSharp**
35. **Lanczos 2**.

Only three tested C^1 methods with no haloing rank above bilinear interpolation, namely Nohalo-LBB which goes from rank 1 to rank 21, and cubic Hermite and EWA cubic Hermite filtering, with unchanged low ranks.

The top EWA methods, all C^1, are

4. *EWA Catmull-Rom*

8. **EWA LanczosSharpest 4**

12. **EWA LanczosSharpest 3**

18. **EWA Lanczos Radius 4**

19. **EWA Lanczos Radius 3**.

All in all, the rankings obtained with and without the re-enlargements of images obtained by nearest neighbour decimation are not very different. One stunning difference is the high rank of EWA Catmull-Rom when the results with nearest neighbour downsampling are excluded. This suggests that EWA Catmull-Rom has a very low rank as a reconstructor of images downsampled with nearest neighbour decimation. This is verified in the next subsection. Less stunning, but still noteworthy, is the significant change in the rank of Nohalo-LBB. Less significant is that the top ranks are populated with 4-lobe windowed Sinc methods.

6.2.3 Overall ranks based on the results of re-enlarging images obtained with nearest neighbour downsampling

The rankings based solely on nearest neighbour downsampling are extremely different from the rankings based on the other downsamplers: Compare Tables 6.3 and 6.6 to Tables 6.2 and 6.5. Some methods rank near the bottom in both cases: nearest neighbour upsampling and tensor and EWA cubic B-spline smoothing. On the other hand, except for the high ranking Nohalo-LBB, the top of the list is completely different: the highest ranked

4-lobe method is KaiserSoft (the *lowest* ranked tensor 4-lobe method when nearest neighbour downsampling results are excluded) at rank 30, while the top C^1 4-lobe method is Parzen-windowed Sinc, which obtains rank 36 when only considering results with nearest neighbour downsampling and ranks an unremarkable 20 otherwise. The top ranked 3-lobe method is Parzen (cubic B-spline) windowed Sinc, with rank 17 when nearest neighbour results are included, rank 41 when excluded.

The overall top-ranked methods when re-enlarging small images obtained by nearest neighbour decimation are Mitchell-Netravali (rank 55 when nearest neighbour downsampling results are excluded), bilinear (rank 59), Nohalo-LBB (top rank), Hann-windowed Sinc 2-lobe (rank 49), and EWA Robidoux (rank 50). Only considering C^1 methods moves Blackman-windowed Sinc 2-lobe (rank 52) and Bohman-windowed Sinc 2-lobe (rank 53) into the top 5 when re-enlarging images downsampled with nearest neighbour. Except for Nohalo-LBB, the top methods with nearest neighbour downsampling are unremarkable when effective downsamplers are used.

The top 10 methods when re-enlarging images obtained with the better low-pass filters pretty much show up in reverse order near the bottom of the nearest neighbour ranking. In particular, EWA Catmull-Rom, the top ranked method with less than 4 lobes when nearest neighbour downsampling results are excluded (rank 4), finds itself at the bottom when only nearest neighbour downsampling results are considered (rank 63, just above nearest neighbour upsampling). Another example: The rankings of the KaiserSoft, Kaiser and KaiserSharp variants of Kaiser-windowed Sinc filtering are reversed, without being at the very bottom.

This near reversal of the rankings is confirmed by considering Spearman rank correlations: The correlation between the rankings shown in Table 6.2 (nearest neighbour downsampling excluded) and Table 6.3 (only nearest neighbour downsampling) is $-.580$, indicating quite a strong disagreement. (The correlation between Table 6.1 and Table 6.2 is .928, and between Table 6.1 and Table 6.3, $-.426$, indicating that the overall ranking

primarily reflects the results obtained with the effective low pass filtering downsamplers.)

The rather obvious observation that one can almost reverse the rankings obtained through re-enlargement tests by substituting nearest neighbour downsampling for box filtering seems to have been first made explicitly by Robidoux et al. [16]. That the rankings are so drastically affected is especially noteworthy given that EXQUIRES uses a version of nearest neighbour interpolation that resolves ties by averaging. (This implementation was chosen to preserve alignment between the full size images and their reduced versions.) As a result, EXQUIRES' nearest neighbour decimation is closer to box filtering than implementations typically used by image processing applications. For example, ratio 2 downsampling with EXQUIRES' nearest neighbour is exactly identical to box filter downsampling. The rankings would be even more different with a truly decimating nearest neighbour downsampler.

It is noteworthy that the top ranked method, when re-enlarging images downsampled with nearest neighbour decimation, is the popular Mitchell-Netravali. Because it is possible that they are indicative of subjective quality, there may be value to re-enlargement tests involving nearest neighbour downsampling, even though they probably should be reported separately since their results are so different to those obtained with decent low pass downsamplers. In any case, the choice of upsampler should take the characteristics of input images into account (a rather obvious statement): Images obtained by nearest neighbour decimation are obviously very different from those obtained with, say, downsampling with Lanczos filtering.

upsampler	rank
nohalo	1
hamming4	*2*
cosine3	3
hann4	4
lanczos4	5
lanczos3	6
bartlett4	*7*
welch3	8
cosine4	9
welch4	10
blackman4	11
ewa_lanczos4sharpest	12
bartlett3	*13*
bohman4	14
hamming3	*15*
hann3	16
ewa_lanczos3sharpest	17
parzen4	18
kaisersharp4	*19*
kaiser4	*20*
welch2	21
kaisersoft4	*22*
ewa_lanczos_radius3	23
kaisersharp3	*24*
cosine2	25
ewa_lanczos_radius4	26
blackman3	27
kaiser3	*28*
bohman3	29
kaisersoft3	*30*
ewa_robidouxsharp	31
lanczos2	32

upsampler	rank
bartlett2	*33*
catmull_rom	34
kaisersharp2	*35*
parzen3	36
ewa_lanczos2sharpest	37
ewa_lanczos_radius2	38
ewa_lanczos3sharp	39
hamming2	*40*
ewa_lanczos4sharp	41
kaiser2	*42*
ewa_lanczos3	43
ewa_mitchell_netravali	44
ewa_lanczos4	45
kaisersoft2	*46*
ewa_lanczos2sharp	47
hann2	48
ewa_catmull_rom	49
ewa_robidoux	50
ewa_lanczos2	51
blackman2	52
bohman2	53
parzen2	54
mitchell_netravali	55
cubic_hermite	56
ewa_hermite	57
ewa_teepee	*58*
bilinear	*59*
ewa_quadratic_b_spline	60
quadratic_b_spline	61
ewa_cubic_b_spline	62
cubic_b_spline	63
nearest	*64*

Table 6.1: XYZ RMSE ranking of linear light upsampling methods as reconstructors of images obtained by linear light downsampling (all downsamplers included). Filters with gradient discontinuities are in italics.

upsampler	rank
welch4	1
cosine4	2
lanczos4	3
ewa_catmull_rom	4
hamming4	*5*
welch3	6
cosine3	7
ewa_lanczos4sharpest	8
hann4	9
lanczos3	10
bartlett4	*11*
ewa_lanczos3sharpest	12
blackman4	13
bartlett3	*14*
bohman4	15
hann3	16
hamming3	*17*
ewa_lanczos_radius4	18
ewa_lanczos_radius3	19
parzen4	20
nohalo	21
kaisersharp4	*22*
welch2	23
kaiser4	*24*
kaisersoft4	*25*
cosine2	26
kaisersharp3	*27*
blackman3	28
ewa_robidouxsharp	29
bohman3	30
kaiser3	*31*
ewa_lanczos4sharp	32

upsampler	rank
kaisersoft3	*33*
bartlett2	*34*
lanczos2	35
ewa_lanczos3sharp	36
catmull_rom	37
ewa_lanczos4	38
kaisersharp2	*39*
ewa_lanczos3	40
parzen3	41
ewa_lanczos2sharpest	42
ewa_lanczos_radius2	43
hamming2	*44*
kaiser2	*45*
ewa_mitchell_netravali	46
ewa_lanczos2sharp	47
kaisersoft2	*48*
hann2	49
ewa_robidoux	50
ewa_lanczos2	51
blackman2	52
bohman2	53
parzen2	54
mitchell_netravali	55
cubic_hermite	56
ewa_hermite	57
ewa_teepee	*58*
bilinear	*59*
ewa_quadratic_b_spline	60
quadratic_b_spline	61
nearest	*62*
ewa_cubic_b_spline	63
cubic_b_spline	64

Table 6.2: XYZ RMSE ranking of linear light upsampling methods as reconstructors of images obtained by linear light downsampling with effective low pass filters (nearest neighbour downsampling results not included). Filters with gradient discontinuities are in italics.

upsampler	rank
mitchell_netravali	1
bilinear	*2*
nohalo	3
hann2	4
ewa_robidoux	5
kaisersoft2	*6*
ewa_teepee	*7*
blackman2	8
bohman2	9
kaiser2	*10*
hamming2	*11*
ewa_lanczos2sharp	12
cubic_hermite	13
ewa_lanczos2	14
ewa_mitchell_netravali	15
parzen2	16
parzen3	17
kaisersharp2	*18*
catmull_rom	19
kaisersoft3	*20*
lanczos2	21
ewa_lanczos_radius2	22
bartlett2	*23*
kaiser3	*24*
bohman3	25
ewa_lanczos2sharpest	26
blackman3	27
kaisersharp3	*28*
ewa_hermite	29
kaisersoft4	*30*
ewa_robidouxsharp	31
cosine2	32

upsampler	rank
kaiser4	*33*
ewa_quadratic_b_spline	34
kaisersharp4	*35*
parzen4	36
welch2	37
quadratic_b_spline	38
ewa_lanczos3	39
ewa_lanczos3sharp	40
bohman4	41
hamming3	*42*
hann3	43
bartlett3	*44*
blackman4	45
ewa_lanczos_radius3	46
ewa_lanczos4	47
ewa_lanczos4sharp	48
bartlett4	*49*
ewa_lanczos_radius4	50
lanczos3	51
ewa_lanczos3sharpest	52
hann4	53
cosine3	54
hamming4	*55*
welch3	56
ewa_lanczos4sharpest	57
lanczos4	58
cosine4	59
ewa_cubic_b_spline	60
welch4	61
cubic_b_spline	62
ewa_catmull_rom	63
nearest	*64*

Table 6.3: XYZ RMSE ranking of linear light upsampling methods as reconstructors of images obtained by linear light downsampling with nearest neighbour. Filters with gradient discontinuities are in italics.

upsampler	xyz_2
nohalo	4.9494
hamming4	4.9581
cosine3	4.9610
hann4	4.9610
lanczos4	4.9613
lanczos3	4.9614
bartlett4	4.9614
welch3	4.9621
cosine4	4.9635
welch4	4.9655
blackman4	4.9668
ewa_lanczos4sharpest	4.9670
bartlett3	4.9683
bohman4	4.9683
hamming3	4.9691
hann3	4.9696
ewa_lanczos3sharpest	4.9712
parzen4	4.9747
kaisersharp4	4.9760
kaiser4	4.9814
welch2	4.9825
kaisersoft4	4.9835
ewa_lanczos_radius3	4.9857
kaisersharp3	4.9896
cosine2	4.9904
ewa_lanczos_radius4	4.9925
blackman3	4.9970
kaiser3	5.0007
bohman3	5.0008
kaisersoft3	5.0060
ewa_robidouxsharp	5.0062
lanczos2	5.0082

upsampler	xyz_2
bartlett2	5.0083
catmull_rom	5.0124
kaisersharp2	5.0186
parzen3	5.0209
ewa_lanczos2sharpest	5.0295
ewa_lanczos_radius2	5.0302
ewa_lanczos3sharp	5.0311
hamming2	5.0316
ewa_lanczos4sharp	5.0317
kaiser2	5.0369
ewa_lanczos3	5.0403
ewa_mitchell_netravali	5.0423
ewa_lanczos4	5.0435
kaisersoft2	5.0599
ewa_lanczos2sharp	5.0606
hann2	5.0617
ewa_catmull_rom	5.0661
ewa_robidoux	5.0710
ewa_lanczos2	5.0945
blackman2	5.1043
bohman2	5.1147
parzen2	5.1328
mitchell_netravali	5.1537
cubic_hermite	5.1726
ewa_hermite	5.1885
ewa_teepee	5.1902
bilinear	5.2109
ewa_quadratic_b_spline	5.3794
quadratic_b_spline	5.4171
ewa_cubic_b_spline	5.5966
cubic_b_spline	5.6289
nearest	5.7086

Table 6.4: XYZ RMSE of linear light upsampling methods as reconstructors of images obtained by linear light downsampling (all downsamplers included). Filters with gradient discontinuities are in italics.

upsampler	xyz_2
welch4	4.7811
cosine4	4.7846
lanczos4	4.7909
ewa_catmull_rom	4.7935
hamming4	4.7943
welch3	4.7948
cosine3	4.7990
ewa_lanczos4sharpest	4.8010
hann4	4.8059
lanczos3	4.8092
bartlett4	4.8108
ewa_lanczos3sharpest	4.8200
blackman4	4.8311
bartlett3	4.8335
bohman4	4.8353
hann3	4.8356
hamming3	4.8358
ewa_lanczos_radius4	4.8498
ewa_lanczos_radius3	4.8501
parzen4	4.8504
nohalo	4.8506
kaisersharp4	4.8532
welch2	4.8571
kaiser4	4.8644
kaisersoft4	4.8685
cosine2	4.8767
kaisersharp3	4.8796
blackman3	4.8928
ewa_robidouxsharp	4.8973
bohman3	4.8989
kaiser3	4.8991
ewa_lanczos4sharp	4.9043

upsampler	xyz_2
kaisersoft3	4.9079
bartlett2	4.9089
lanczos2	4.9107
ewa_lanczos3sharp	4.9161
catmull_rom	4.9176
ewa_lanczos4	4.9198
kaisersharp2	4.9278
ewa_lanczos3	4.9283
parzen3	4.9313
ewa_lanczos2sharpest	4.9354
ewa_lanczos_radius2	4.9381
hamming2	4.9475
kaiser2	4.9551
ewa_mitchell_netravali	4.9592
ewa_lanczos2sharp	4.9840
kaisersoft2	4.9870
hann2	4.9906
ewa_robidoux	5.0012
ewa_lanczos2	5.0258
blackman2	5.0422
bohman2	5.0546
parzen2	5.0734
mitchell_netravali	5.1109
cubic_hermite	5.1257
ewa_hermite	5.1306
ewa_teepee	5.1512
bilinear	5.1817
ewa_quadratic_b_spline	5.3651
quadratic_b_spline	5.4052
nearest	5.5229
ewa_cubic_b_spline	5.5892
cubic_b_spline	5.6228

Table 6.5: XYZ RMSE of linear light upsampling methods as reconstructors of images obtained by linear light downsampling with effective low pass filters (nearest neighbour downsampling not included). Filters with gradient discontinuities are in italics.

upsampler	xyz_2
mitchell_netravali	5.3213
bilinear	5.3260
nohalo	5.3263
hann2	5.3364
ewa_robidoux	5.3411
kaisersoft2	5.3417
ewa_teepee	5.3436
blackman2	5.3456
bohman2	5.3480
kaiser2	5.3515
hamming2	5.3548
ewa_lanczos2sharp	5.3559
cubic_hermite	5.3560
ewa_lanczos2	5.3607
ewa_mitchell_netravali	5.3617
parzen2	5.3637
parzen3	5.3646
kaisersharp2	5.3666
catmull_rom	5.3748
kaisersoft3	5.3806
lanczos2	5.3808
ewa_lanczos_radius2	5.3830
bartlett2	5.3875
kaiser3	5.3879
bohman3	5.3892
ewa_lanczos2sharpest	5.3895
blackman3	5.3938
kaisersharp3	5.4071
ewa_hermite	5.4141
kaisersoft4	5.4191
ewa_robidouxsharp	5.4200
cosine2	5.4214

upsampler	xyz_2
kaiser4	5.4241
ewa_quadratic_b_spline	5.4360
kaisersharp4	5.4394
parzen4	5.4436
welch2	5.4554
quadratic_b_spline	5.4644
ewa_lanczos3	5.4653
ewa_lanczos3sharp	5.4667
bohman4	5.4683
hamming3	5.4699
hann3	5.4729
bartlett3	5.4744
blackman4	5.4762
ewa_lanczos_radius3	5.4948
ewa_lanczos4	5.5104
ewa_lanczos4sharp	5.5120
bartlett4	5.5231
ewa_lanczos_radius4	5.5266
lanczos3	5.5286
ewa_lanczos3sharpest	5.5349
hann4	5.5384
cosine3	5.5618
hamming4	5.5651
welch3	5.5817
ewa_lanczos4sharpest	5.5818
lanczos4	5.5911
cosine4	5.6224
ewa_cubic_b_spline	5.6264
welch4	5.6432
cubic_b_spline	5.6530
ewa_catmull_rom	6.0346
nearest	6.3979

Table 6.6: XYZ RMSE of linear light upsampling methods as reconstructors of images obtained by linear light nearest neighbour downsampling. Filters with gradient discontinuities are in italics.

6.3 Impact on upsampler ranks of various factors

In this and the following sections, we analyze the impact of various factors on upsampler rankings, extending the analysis performed on the data of Dumic et al. (§3.2) to our own.

The current section's Spearman rank correlations are obtained by slicing the data used to compute the overall ranks given in §6.2.1. Subsequent sections (§6.4–6.6) use additional data.

6.3.1 Impact on upsampler ranks of downsampler choice

As seen in Table 6.7, the Spearman rank correlations of every pair of downsamplers is above .942 except for the cross correlations involving nearest neighbour decimation. The conclusion to be drawn is that the choice of downsampler does not matter much provided it is a decent low-pass filter (which nearest neighbour decimation is not).

downsampler	box filtering	Gaussian blur	Lanczos 3-lobe filtering	EWA Lanczos 3-lobe filtering	nearest neighbour decimation
box filtering	**1**	**.964**	**.954**	**.947**	-.531
Gaussian blur	**.964**	**1**	**.942**	**.971**	-.615
Lanczos 3-lobe filtering	**.954**	**.942**	**1**	**.985**	-.511
EWA Lanczos 3-lobe filtering	**.947**	**.971**	**.985**	**1**	-.567
nearest neighbour decimation	-.531	-.615	-.511	-.567	**1**

Table 6.7: Downsamplers Spearman rank correlation matrix. Cross-correlations higher than .9 are in boldface.

6.3.2 Impact on upsampler ranks of the resampling ratio

As seen in Table 6.8, the Spearman rank correlation of every pair of downsampling ratios is above .887. The conclusion to be drawn is that more than one downsampling ratio should generally be used, taking into account the fact that low and high ratios are the least correlated when choosing ratios to test, but that in any case the rankings obtained with different ratios are fairly similar.

downsampling ratio	2	4	3	6	8	5	7
2	**1**	**.980**	**.966**	**.946**	**.920**	**.914**	.887
4	**.980**	**1**	**.989**	**.984**	**.968**	**.961**	**.936**
3	**.966**	**.989**	**1**	**.992**	**.979**	**.973**	**.952**
6	**.946**	**.984**	**.992**	**1**	**.993**	**.989**	**.972**
8	**.920**	**.968**	**.979**	**.993**	**1**	**.997**	**.987**
5	**.914**	**.961**	**.973**	**.989**	**.997**	**1**	**.993**
7	.887	**.936**	**.952**	**.972**	**.987**	**.993**	**1**

Table 6.8: Ratios Spearman rank correlation matrix.

6.3.3 Impact on upsampler ranks of test image choice

As seen in Table 6.9, the Spearman rank correlations of every pair of test images is above .728. The boy and shed images are the least correlated. In addition, if one takes the shed image out, all cross-correlations are above .849 and, if one also excludes the cat image, all correlations are above .887. The conclusion to be drawn is that the choice of test images matters quite a bit, but that the rankings obtained with different test images are generally somewhat similar.

This being said, note that all test images were obtained by careful downsampling of high quality digital photographs and of an archival scan. We expect different *types* of images (text, CG, destructively compressed, blurry, with significant chromatic aberration...) to give very different rankings.

test image	curios	baby	wave	apartments	dragon	garland	frog	wreck	man	paint	footbridge	cabins	tower	horse	cat	boy	shed
curios	**1**	**.994**	**.991**	**.987**	**.987**	**.985**	**.983**	**.977**	**.974**	**.974**	**.973**	**.969**	**.963**	**.960**	**.954**	**.921**	.811
baby	**.994**	**1**	**.986**	**.981**	**.984**	**.974**	**.978**	**.981**	**.974**	**.974**	**.979**	**.967**	**.966**	**.968**	**.963**	**.928**	.829
wave	**.991**	**.986**	**1**	**.993**	**.998**	**.977**	**.978**	**.984**	**.945**	**.985**	**.960**	**.937**	**.933**	**.943**	**.974**	.893	.839
apartments	**.987**	**.981**	**.993**	**1**	**.990**	**.981**	**.979**	**.968**	**.944**	**.983**	**.941**	**.930**	**.919**	**.923**	**.963**	.898	.822
dragon	**.987**	**.984**	**.998**	**.990**	**1**	**.972**	**.976**	**.986**	**.940**	**.987**	**.959**	**.930**	**.928**	**.941**	**.979**	.889	.848
garland	**.985**	**.974**	**.977**	**.981**	**.972**	**1**	**.989**	**.956**	**.964**	**.972**	**.953**	**.959**	**.947**	**.939**	**.928**	**.936**	.782
frog	**.983**	**.978**	**.978**	**.979**	**.976**	**.989**	**1**	**.964**	**.960**	**.988**	**.958**	**.952**	**.945**	**.944**	**.940**	**.954**	.810
wreck	**.977**	**.981**	**.984**	**.968**	**.986**	**.956**	**.964**	**1**	**.932**	**.976**	**.979**	**.935**	**.941**	**.966**	**.983**	.887	.888
man	**.974**	**.974**	**.945**	**.944**	**.940**	**.964**	**.960**	**.932**	**1**	**.932**	**.956**	**.989**	**.980**	**.949**	.894	**.953**	.731
paint	**.974**	**.974**	**.985**	**.983**	**.987**	**.972**	**.988**	**.976**	**.932**	**1**	**.949**	**.917**	**.916**	**.932**	**.972**	**.923**	.857
footbridge	**.973**	**.979**	**.960**	**.941**	**.959**	**.953**	**.958**	**.979**	**.956**	**.949**	**1**	**.971**	**.982**	**.995**	**.940**	**.917**	.839
cabins	**.969**	**.967**	**.937**	**.930**	**.930**	**.959**	**.952**	**.935**	**.989**	**.917**	**.971**	**1**	**.993**	**.968**	.884	**.942**	.741
tower	**.963**	**.966**	**.933**	**.919**	**.928**	**.947**	**.945**	**.941**	**.980**	**.916**	**.982**	**.993**	**1**	**.983**	.890	**.937**	.765
horse	**.960**	**.968**	**.943**	**.923**	**.941**	**.939**	**.944**	**.966**	**.949**	**.932**	**.995**	**.968**	**.983**	**1**	**.926**	**.914**	.838
cat	**.954**	**.963**	**.974**	**.963**	**.979**	**.928**	**.940**	**.983**	.894	**.972**	**.940**	.884	.890	**.926**	**1**	.849	**.921**
boy	**.921**	**.928**	.893	.898	.889	**.936**	**.954**	.887	**.953**	**.923**	**.917**	**.942**	**.937**	**.914**	.849	**1**	.728
shed	.811	.829	.839	.822	.848	.782	.810	.888	.731	.857	.839	.741	.765	.838	**.921**	.728	**1**

Table 6.9: Test images Spearman rank correlation matrix.

6.4 Impact on upsampler ranks of the choice of error metric

So far, all rankings and cross-correlations have been based solely on the XYZ RMSE (xyz_2) image difference metric. If another metric is used—XYZ AAE (xyz_1) or MAXABS (xyz_1) or sRGB RMSE (srgb_2) or MSSIM (mssim), for example—are the rankings similar, or are they very different, indicating a strong dependence of the rank on the error measure? In this chapter, we answer this question using, once again, a Spearman rank correlation matrix which shows the correlation of the rankings obtained with pairs of difference metrics: Table 6.10.

Some metrics are in close agreement. For example, srgb_1 (AVGABS, the average of the absolute value of the difference using sRGB pixel values) and srgb_2 (RMSE using sRGB pixel values) are almost perfectly aligned: their correlation is .993. On the other hand, xyz_2 (RMSE using XYZ pixel values) and cmc_∞ (MAX CMC 1:1 ΔE) are in near perfect *disagreement*, with a correlation of $-.783$. This is not particularly surprising: ∞-norms (MAXABS) favour different methods than 2-norms (RMSE) because they tend to "punish" methods on the basis of over- and undershoots without regard for much else. For this reason, the usefulness of MAXABS-type norms is highly questionable in the present context because it basically only measures one type of deviation. In addition, the CMC 1:1 ΔE distance between colours was designed to mimic perception in the context of manufacturing tolerance, whence the XYZ colour space is physically linear: one could say that these metrics are very different by design.

With the chosen ordering, namely one based on the correlation with the xyz_2 norm, rank correlations are close to being monotone with respect to distance from the matrix diagonal. This is reflected in the ranks shown in Tables 6.11–6.12. High ranks are generally clustered near the diagonal; low ranks, away from it. (The rank table's diagonal is slanted: the combined Tables 6.11–6.12 have 64 rows and 17 columns.) In other words, different metrics favour different groups of upsamplers. The one exception is the one nonlinear

filter tested, Nohalo-LBB, which takes first place with respect to fourteen of the seventeen metrics.

Basically, metrics fall into three groups:

- xyz_2, xyz_4, xyz_1, blur_1, blur_2, mssim, srgb_1 and srgb_2
- blur_4, cmc_1, srgb_4, cmc_2 and cmc_4
- the MAXABS metrics srgb_∞, xyz_∞, blur_∞ and cmc_∞.

Within each group, the metrics are reasonably well correlated. However, metrics from different groups are not, unless they are close to the transition from one group to the next. Consider the more commonly used error metrics xyz_2, xyz_1, mssim, srgb_1 and srgb_2. All of them belong to the first group. As mentioned earlier, srgb_1 and srgb_2 give nearly identical rankings. Likewise, xyz_2 and xyz_1 (and, actually, xyz_4) are in close agreement, with a cross-correlation of .958. mssim is fairly close to srgb_1 (.948) and srgb_2 (.943); it is more loosely correlated with xyz_1 (.818) and, especially, xyz_2 (.672). This is not particularly surprising given that mssim is based on luma, a luminance (light intensity) value obtained by averaging sRGB colour values. The XYZ metric most correlated to the commonly used sRGB-based metrics (including mssim) is based on the ℓ_1-norm (xyz_1), followed by ℓ_4 (xyz_4). The most distant pairs in the group of five are xyz_2 and srgb_2 (.510) and xyz_2 and srgb_1 (.505). These results make clear that rankings should involve more than one metric: a Spearman rank correlation of .5 or so does not indicate strong agreement. It also makes clear that the colour space in which RMSE is computed matters a lot, even though it is rarely, if ever, specified in publications that contain a quantitative comparison of image resampling methods.

The blur metrics were included in EXQUIRES to investigate the dependence of the MSSIM ranking on its blur and crop components (leaving out the "correlation" terms of SSIM). Although one may be tempted to say "a fair amount" given that the cross-correlation of blur_2 with mssim is .872, the fact that the cross-correlations of mssim with srgb_1 and

srgb_2 are even larger suggests that a key ingredient is SSIM's formulation in terms of luma ("sRGB luminance"). This, however, may only hold in the context of re-enlargement tests: Images that result from enlargement are generally smoother (locally) and less noisy than those for which SSIM was designed. Consequently, blurring and cropping have less of an effect on them than they do on the types of images that SSIM was designed to evaluate. (Note that the blur metrics will be based on the XYZ colour space instead of the sRGB colour space in future releases of EXQUIRES, because averaging convolutions are generally considered to be better performed in linear light [73, 143].)

Warning: The reported metric cross-correlations may depend strongly on the specifics of the context in which they were measured, namely an image re-enlargement experiment. Metric cross-correlations may, or may not, change significantly if the metrics are used to measure JPEG compression artifacts, for example.

	xyz_2	xyz_1	xyz_4	$blur_1$	$blur_2$	mssim	$srgb_2$	$srgb_1$	$blur_4$	cmc_1	$srgb_4$	cmc_2	cmc_4	$srgb_\infty$	xyz_∞	$blur_\infty$	cmc_∞
xyz_2	**1**	**.958**	**.911**	.831	.687	.672	.510	.505	.224	.173	.035	−.045	−.294	−.636	−.698	−.757	−.783
xyz_1	**.958**	**1**	**.912**	**.919**	.814	.818	.663	.661	.380	.338	.195	.111	−.148	−.519	−.654	−.643	−.675
xyz_4	**.911**	**.912**	**1**	.810	.691	.659	.505	.494	.217	.144	.040	−.075	−.311	−.665	−.615	−.783	−.800
$blur_1$	.831	**.919**	.810	**1**	**.948**	.824	.701	.697	.557	.381	.249	.156	−.105	−.471	−.630	−.498	−.584
$blur_2$	.687	.814	.691	**.948**	**1**	.872	.817	.811	.750	.539	.434	.321	.061	−.329	−.529	−.350	−.417
mssim	.672	.818	.659	.824	.872	**1**	**.943**	**.948**	.738	.752	.634	.555	.291	−.134	−.417	−.278	−.281
$srgb_2$	.510	.663	.505	.701	.817	**.943**	**1**	**.993**	.879	.877	.817	.722	.487	.024	−.296	−.116	−.103
$srgb_1$	.505	.661	.494	.697	.811	**.948**	**.993**	**1**	.877	.894	.812	.740	.501	.034	−.301	−.101	−.090
$blur_4$	.224	.380	.217	.557	.750	.738	.879	.877	**1**	.861	.833	.760	.571	.107	−.214	.094	.078
cmc_1	.173	.338	.144	.381	.539	.752	.877	.894	.861	**1**	**.939**	**.956**	.809	.339	−.054	.224	.246
$srgb_4$	.035	.195	.040	.249	.434	.634	.817	.812	.833	**.939**	**1**	**.940**	.853	.404	.085	.292	.338
cmc_2	−.045	.111	−.075	.156	.321	.555	.722	.740	.760	**.956**	**.940**	**1**	**.940**	.508	.127	.408	.437
cmc_4	−.294	−.148	−.311	−.105	.061	.291	.487	.501	.571	.809	.853	**.940**	**1**	.666	.354	.596	.629
$srgb_\infty$	−.636	−.519	−.665	−.471	−.329	−.134	.024	.034	.107	.339	.404	.508	.666	**1**	.683	**.927**	**.949**
xyz_∞	−.698	−.654	−.615	−.630	−.529	−.417	−.296	−.301	−.214	−.054	.085	.127	.354	.683	**1**	.643	.730
$blur_\infty$	−.757	−.643	−.783	−.498	−.350	−.278	−.116	−.101	.094	.224	.292	.408	.596	**.927**	.643	**1**	**.965**
cmc_∞	−.783	−.675	−.800	−.584	−.417	−.218	−.103	−.090	.078	.246	.338	.437	.629	**.949**	.730	**.965**	**1**

Table 6.10: Metrics Spearman rank correlation matrix.

upsampler	xyz_2	xyz_4	xyz_1	$blur_1$	$blur_2$	mssim	$srgb_1$	$srgb_2$	$blur_4$	cmc_1	$srgb_4$	cmc_2	cmc_4	$srgb_\infty$	xyz_∞	$blur_\infty$	cmc_∞
nohalo	**1**	**1**	46	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	21	**11**	**7**
hamming4	**2**	**4**	**7**	**13**	32	27	36	34	47	42	*53*	47	*55*	*60.5*	*58*	*59*	*60*
cosine3	**3**	**13**	**11**	27	33	32	40	37	*49*	44	*54*	*49*	*56*	*60.5*	*57*	*58*	*56*
hann4	**4**	**11**	**2**	17	27	23	32	33	41	40	45	43	*51*	47	*55*	*54*	*57*
lanczos4	**5**	**16**	**13**	26	35	37	42	40	*51*	48	*55*	*52*	*57*	*60.5*	*61*	*61*	*61*
lanczos3	**6**	**7**	**6**	**16**	26	25	33	31	40	39	46	44	*52*	47	*52*	*53*	*55*
bartlett4	**7**	**2**	**5**	**8**	**13**	18	22	26	35	34	38	40	42	43	*53*	*52*	*59*
welch3	**8**	18	**14**	33	37	38	44	45	*52*	*50*	*56*	*54*	*58*	*60.5*	*59*	*60*	*58*
cosine4	**9**	20	18	31	39	40	47	46	*55*	*51*	*57*	*56*	*59*	*60.5*	*62*	*62*	*63*
welch4	**10**	24	24	36	42	44	48	47	*56*	*53*	*58*	*58*	*60*	*60.5*	*63*	*63*	*64*
blackman4	**11**	**5**	**1**	**5**	**10**	**12**	20	20	30	32	34	37	39	43	*50*	47	47
ewa_lanczos4sharpest	**12**	21	23	**14**	29	34	39	41	44	*49*	*51*	48	*54*	*52.5*	*54*	48	*62*
bartlett3	**13**	**3**	**12**	**4**	**2**	**6**	**13**	**15**	25	26	30	31	36	43	*51*	44	*49*
bohman4	**14**	**6**	**4**	**3**	**7**	**8**	19	18	27	27	31	33	37	43	48	45	45
hamming3	**15**	**9**	**3**	**11**	**11**	**15**	21	21	28	31	32	35	38	43	47	46	46
hann3	**16**	**8**	**9**	**10**	17	17	26	22	31	33	36	39	40	47	45	43	44
ewa_lanczos3sharpest	17	17	26	**6**	21	26	34	35	38	41	47	45	47	*52.5*	*56*	39	*53*
parzen4	18	**10**	**8**	**2**	**3**	**4**	**14**	**13**	22	23	25	29	34	39	23	42	42
kaisersharp4	19	**12**	**10**	**7**	**4**	**5**	**12**	**12**	20	22	23	28	33	34.5	22	41	41
kaiser4	20	**14**	**15**	**9**	**5**	**3**	**9**	**9**	17	18	21	24	32	34.5	26	38	37
welch2	21	23	**16**	22	22	35	31	32	29	38	37	41	41	40	*49*	40	43
kaisersoft4	22	**15**	17	**12**	**6**	**2**	**7**	**8**	**15**	**15**	19	23	30	34.5	27	37	36
ewa_lanczosradius3	23	31	21	35	36	42	41	44	37	45	40	46	45	*52.5*	17	*51*	48
kaisersharp3	24	19	22	**15**	**8**	**7**	**8**	**7**	**11**	**14**	**16**	22	28	34.5	38	35	33
cosine2	25	27	20	19	18	29	25	25	21	30	27	32	35	34.5	46	36	38
ewa_lanczosradius4	26	38	19	40	41	*49*	46	48	43	*52*	48	*51*	*49*	*52.5*	20	*56*	*54*
blackman3	27	22	25	18	**12**	**9**	**4**	**6**	**6**	**12**	**13**	**16**	24	34.5	37	34	32
kaiser3	28	26	27	25	**16**	**10**	**2**	**3**	**4**	**9**	**9**	**13**	21	23	35	33	30
bohman3	29	25	28	21	**14**	**11**	**6**	**4**	**5**	**10**	**11**	**15**	23	34.5	40	32	31
kaisersoft3	30	29	30	28	20	**13**	**3**	**5**	**3**	**6**	**6**	**10**	**16**	23	33	31	28
ewa_robidouxsharp	31	30	35	23	19	28	27	28	19	29	26	30	31	34.5	44	29	34
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 6.11: Upsampler ranks by metric (part 1). Filters with gradient discontinuities are in italics. Ranks in the top quartile are in boldface, in the bottom quartile, in italics.

upsampler	xyz_2	xyz_4	xyz_1	$blur_1$	$blur_2$	mssim	$srgb_1$	$srgb_2$	$blur_4$	cmc_1	$srgb_4$	cmc_2	cmc_4	$srgb_\infty$	xyz_∞	$blur_\infty$	cmc_∞
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
lanczos2	32	33	29	30	23	24	17	19	**12**	17	**14**	20	25	23	43	30	29
bartlett2	33	28	34	24	**15**	**14**	**5**	**2**	**2**	**3**	**12**	**9**	17	23	42	27	27
catmull_rom	34	32	31	32	25	19	**11**	**11**	**9**	**11**	**7**	**11**	**14**	23	41	28	26
kaisersharp2	35	34	36	37	30	20	**15**	**14**	**10**	**8**	**5**	**8**	**11**	23	36	26	20.5
parzen3	36	35	37	38	31	**16**	**10**	**10**	**8**	**2**	**2**	**4**	**9**	23	29	25	20.5
ewa_lanczos2sharpest	37	37	44	29	24	31	23	24	**13**	20	20	18	20	23	31	20	20.5
ewa_lanczosradius2	38	39	41	34	28	33	24	27	**14**	21	18	19	18	23	32	21	20.5
ewa_lanczos3sharp	39	44	33	46	47	*51*	*50*	*52*	45	*55*	42	*50*	43	*52.5*	**9.5**	*49*	39
hamming2	40	36	40	39	34	21	**16**	**16**	**16**	**4**	**3**	**5**	**8**	23	25	19	20.5
ewa_lanczos4sharp	41	46	32	47	48	*56*	*52*	*54*	*50*	*57*	*49*	*55*	48	*52.5*	**9.5**	*55*	*50*
kaiser2	42	40	43	41	38	22	18	17	18	**5**	**4**	**3**	**6**	23	24	18	20.5
ewa_lanczos3	43	47	38	*49*	*49*	*55*	*51*	*53*	48	*56*	43	*53*	44	*52.5*	**9.5**	*50*	40
ewa_mitchell_netravali	44	41	42	42	40	39	30	30	23	25	**15**	17	**12**	23	39	23	20.5
ewa_lanczos4	45	*49*	39	*50*	*51*	*58*	*56*	*57*	*54*	*58*	*52*	*57*	*53*	*52.5*	**9.5**	*57*	*51*
kaisersoft2	46	42	*49*	43	43	30	28	23	24	**7**	**10**	**2**	**2**	**12.5**	19	**15**	**16**
ewa_lanczos2sharp	47	45	45	44	45	48	37	42	32	35	22	26	19	23	34	24	20.5
hann2	48	43	48	45	44	36	29	29	26	**13**	**8**	**6**	**4**	**12.5**	18	**14**	**15**
ewa_catmull_rom	*49*	*52*	*58*	*60*	*60*	45	*60*	*59*	*62*	*62*	*63*	*64*	*64*	*60.5*	*64*	*64*	*52*
ewa_robidoux	*50*	48	47	48	46	47	35	39	33	28	17	21	**10**	23	28	17	**14**
ewa_lanczos2	*51*	*53*	*50*	*52*	*54*	*52*	*49*	*49*	42	43	29	34	26	23	30	22	25
blackman2	*52*	*50*	*51*	*51*	*50*	41	38	36	34	**16**	24	**7**	**3**	**12.5**	**15**	**13**	**11.5**
bohman2	*53*	*51*	*53*	*53*	*52*	43	43	38	36	19	28	**12**	**5**	**12.5**	**14**	**12**	**11.5**
parzen2	*54*	*54*	*54*	*54*	*53*	46	45	43	39	24	35	**14**	**7**	**12.5**	**13**	**9**	**10**
mitchell_netravali	*55*	*57*	*52*	*57*	*57*	*54*	*55*	*55*	*57*	46	33	36	22	**12.5**	**16**	**7**	**13**
cubic_hermite	*56*	*55*	*56*	*56*	*56*	*50*	*53*	*51*	*53*	37	39	27	**13**	**7**	**7**	**8**	**6**
ewa_hermite	*57*	*56*	*59*	*55*	*55*	*53*	*54*	*50*	46	36	*50*	25	**15**	**6**	**12**	**10**	**9**
ewa_teepee	*58*	*58*	*55*	*58*	*58*	*57*	*57*	*56*	*58*	47	41	38	27	**8**	**6**	**6**	**5**
bilinear	*59*	*59*	*57*	*59*	*59*	*59*	*58*	*58*	*59*	*54*	44	42	29	**9**	**5**	**5**	**8**
ewa_quadratic_b_spline	*60*	*60*	*60*	*61*	*61*	*60*	*59*	*61*	*60*	*60*	*59*	*59*	46	**5**	**4**	**4**	**4**
quadratic_b_spline	*61*	*61*	*61*	*62*	*62*	*61*	*61*	*62*	*61*	*61*	*60*	*60*	*50*	**4**	**3**	**3**	**3**
ewa_cubic_b_spline	*62*	*63*	*62*	*63*	*63*	*62*	*63*	*63*	*63*	*63*	*61*	*62*	*61*	**3**	**2**	**2**	**2**
cubic_b_spline	*63*	*64*	*63*	*64*	*64*	*63*	*64*	*64*	*64*	*64*	*62*	*63*	*62*	**2**	**1**	**1**	**1**
nearest	*64*	*62*	*64*	20	**9**	*64*	*62*	*60*	**7**	*59*	*64*	*61*	*63*	*60.5*	*60*	**16**	35

Table 6.12: Upsampler ranks by metric (part 2).

6.5 Impact on upsampler ranks of filtering sRGB values without converting to and from linear light

Still re-enlarging images obtained by downsampling through linear light, let us consider the rankings obtained if one filters using sRGB values directly (without going through linear light). If one compares the rank obtained by the same filter in both configurations, that is, if one compares the ranks shown in Table 6.1 (through linear light) to those shown in Table 6.13 (straight through sRGB), one sees that they are barely different. Indeed, the Spearman rank cross-correlation between the two rankings is .984. The "straight through sRGB" ranks are obtained from the XYZ RMSE errors obtained when enlarging straight through sRGB, shown in Table 6.14. Interestingly, the popular Lanczos 3-lobe earns top rank among C^1 methods.

upsampler	rank
hamming4	*1*
bartlett4	*2*
lanczos3	3
hann4	4
cosine3	5
lanczos4	6
welch3	7
blackman4	8
ewa_lanczos4sharpest	9
bartlett3	*10*
bohman4	11
cosine4	12
hamming3	*13*
hann3	14
ewa_lanczos3sharpest	15
welch4	16
parzen4	17
kaisersharp4	*18*
kaiser4	*19*
ewa_lanczos_radius3	20
kaisersoft4	*21*
welch2	22
nohalo	23
kaisersharp3	*24*
ewa_lanczos_radius4	25
cosine2	26
blackman3	27
bohman3	28
kaiser3	*29*
ewa_robidouxsharp	30
kaisersoft3	*31*
bartlett2	*32*

upsampler	rank
lanczos2	33
catmull_rom	34
kaisersharp2	*35*
parzen3	36
ewa_lanczos3sharp	37
ewa_lanczos4sharp	38
ewa_lanczos2sharpest	39
ewa_lanczos_radius2	40
hamming2	*41*
ewa_lanczos3	42
ewa_lanczos4	43
kaiser2	*44*
ewa_mitchell_netravali	45
ewa_lanczos2sharp	46
kaisersoft2	*47*
hann2	48
ewa_robidoux	49
ewa_lanczos2	50
blackman2	51
ewa_catmull_rom	52
bohman2	53
parzen2	54
mitchell_netravali	55
cubic_hermite	56
ewa_hermite	57
ewa_teepee	*58*
bilinear	*59*
ewa_quadratic_b_spline	60
quadratic_b_spline	61
ewa_cubic_b_spline	62
cubic_b_spline	63
nearest	*64*

Table 6.13: XYZ RMSE ranking of sRGB upsampling methods as reconstructors of images obtained by linear light downsampling. Filters with gradient discontinuities are in italics.

upsampler	xyz_2
hamming4	4.9132
bartlett4	4.9151
lanczos3	4.9156
hann4	4.9158
cosine3	4.9168
lanczos4	4.9182
welch3	4.9191
blackman4	4.9200
ewa_lanczos4sharpest	4.9205
bartlett3	4.9211
bohman4	4.9214
cosine4	4.9222
hamming3	4.9225
hann3	4.9227
ewa_lanczos3sharpest	4.9232
welch4	4.9255
parzen4	4.9280
kaisersharp4	4.9294
kaiser4	4.9353
ewa_lanczos_radius3	4.9375
kaisersoft4	4.9376
welch2	4.9379
nohalo	4.9438
kaisersharp3	4.9444
ewa_lanczos_radius4	4.9455
cosine2	4.9460
blackman3	4.9529
bohman3	4.9572
kaiser3	4.9572
ewa_robidouxsharp	4.9598
kaisersoft3	4.9634
bartlett2	4.9656

upsampler	xyz_2
lanczos2	4.9664
catmull_rom	4.9701
kaisersharp2	4.9783
parzen3	4.9809
ewa_lanczos3sharp	4.9849
ewa_lanczos4sharp	4.9864
ewa_lanczos2sharpest	4.9882
ewa_lanczos_radius2	4.9888
hamming2	4.9935
ewa_lanczos3	4.9946
ewa_lanczos4	4.9988
kaiser2	5.0000
ewa_mitchell_netravali	5.0014
ewa_lanczos2sharp	5.0214
kaisersoft2	5.0273
hann2	5.0300
ewa_robidoux	5.0351
ewa_lanczos2	5.0582
blackman2	5.0788
ewa_catmull_rom	5.0872
bohman2	5.0908
parzen2	5.1107
mitchell_netravali	5.1344
cubic_hermite	5.1575
ewa_hermite	5.1710
ewa_teepee	5.1780
bilinear	5.2038
ewa_quadratic_b_spline	5.3909
quadratic_b_spline	5.4334
ewa_cubic_b_spline	5.6289
cubic_b_spline	5.6648
nearest	5.7086

Table 6.14: XYZ RMSE of sRGB upsampling methods as reconstructors of images obtained by linear light downsampling (all downsamplers included). Filters with gradient discontinuities are in italics.

6.6 Don't assume that enlarging through linear light is better

Is it better to enlarge an sRGB image taking the sRGB values as is, or better to enlarge them through linear RGB with sRGB primaries, that is, by first converting the sRGB values to linear RGB, applying the resizing filter, and finally converting back to sRGB? (Note that in the case of Nohalo-LBB, the linear light space is not linear RGB with sRGB primaries, it is XYZ.)

If processing is performed in 8-bit from end to end, the answer is "Definitely not." The reason for this is that the conversion into and out of linear RGB is very lossy in 8-bit. (Note that 8-bit output images can be produced from 8-bit input images with 16-bit or floating point intermediate images. "Definitely not in 8-bit" concerns key stages within the toolchain, not the input and output images.)

What if, however, the colour space conversions are performed so that the introduced error is inconsequential, as is the case with EXQUIRES, in which stored results are 16-bit and all calculations are performed in floating point? The surprising answer is: "If you use a filter with negative lobes, don't enlarge through linear RGB with sRGB primaries. Use the sRGB values 'as is'." At least if your image was not obtained by downsampling using sRGB values directly, as is common for web images: If the image was obtained by downsampling sRGB values directly, you probably are better off enlarging through linear light. The quantitative basis for these rather surprising conclusions ("If downsampling in linear light, re-enlarge in sRGB; if downsampling in sRGB, re-enlarge in linear light.") is presented in the remainder of this chapter. Additional discussion is found in the ImageMagick forum threads [144–146].

6.6.1 Re-enlarging images obtained by linear light downsampling

In Table 6.15, enlarging by filtering the sRGB values directly is indicated by italics, and a roman font indicates enlargements through linear RGB with sRGB primaries.

Filtering sRGB values directly almost always leads to a smaller error than filtering through linear light. There are five exceptions: EWA Catmull-Rom, and the tensor and EWA quadratic and cubic B-spline smoothing filters, filters without negative lobes. Not only do "sRGB" almost always rank above "linear light" enlargements, but their xyz_2 errors are significantly smaller: Compare Table 6.14 with Table 6.4.

6.6.2 Re-enlarging images obtained by direct sRGB downsampling

Many images, in particular those found on the web, are the result of downsizing by filtering the sRGB values directly. For this reason, we separately consider the re-enlargement of such images. The results are shown in Table 6.16.

In almost every case, re-enlarging the result of direct sRGB downsampling gives a more accurate result if the enlargement is performed through linear light. There are only two exceptions: nearest neighbour upsampling gives rise to a tie between its two variants (as should be), and Nohalo-LBB, which uses XYZ instead of linear RGB with sRGB primaries.

upsampler	rank
hamming4	*1*
bartlett4	*2*
lanczos3	*3*
hann4	*4*
cosine3	*5*
lanczos4	*6*
welch3	*7*
blackman4	*8*
ewa_lanczos4sharpest	*9*
bartlett3	*10*
bohman4	*11*
cosine4	*12*
hamming3	*13*
hann3	*14*
ewa_lanczos3sharpest	*15*
welch4	*16*
parzen4	*17*
kaisersharp4	*18*
kaiser4	*19*
ewa_lanczos_radius3	*20*
kaisersoft4	*21*
welch2	*22*
nohalo	*23*
kaisersharp3	*24*
ewa_lanczos_radius4	*25*
cosine2	*26*
nohalo	27
blackman3	*28*
bohman3	*29*
kaiser3	*30*
hamming4	31
ewa_robidoux_sharp	*32*
cosine3	33
hann4	34
lanczos4	35
lanczos3	36
bartlett4	37
welch3	38
cosine4	39
kaisersoft3	*40*
welch4	41
bartlett2	*42*
lanczos2	*43*

upsampler	rank
blackman4	44
ewa_lanczos4sharpest	45
bartlett3	46
bohman4	47
hamming3	48
hann3	49
catmull_rom	*50*
ewa_lanczos3sharpest	51
parzen4	52
kaisersharp4	53
kaisersharp2	*54*
parzen3	*55*
kaiser4	56
welch2	57
kaisersoft4	58
ewa_lanczos3sharp	*59*
ewa_lanczos_radius3	60
ewa_lanczos4sharp	*61*
ewa_lanczos2sharpest	*62*
ewa_lanczos_radius2	*63*
kaisersharp3	64
cosine2	65
ewa_lanczos_radius4	66
hamming2	*67*
ewa_lanczos3	*68*
blackman3	69
ewa_lanczos4	*70*
kaiser2	*71*
kaiser3	72
bohman3	73
ewa_mitchell_netravali	*74*
kaisersoft3	75
ewa_robidoux_sharp	76
lanczos2	77
bartlett2	78
catmull_rom	79
kaisersharp2	80
parzen3	81
ewa_lanczos2sharp	*82*
kaisersoft2	*83*
ewa_lanczos2sharpest	84
hann2	*85*
ewa_lanczos_radius2	86

upsampler	rank
ewa_lanczos3sharp	87
hamming2	88
ewa_lanczos4sharp	89
ewa_robidoux	*90*
kaiser2	91
ewa_lanczos3	92
ewa_mitchell_netravali	93
ewa_lanczos4	94
ewa_lanczos2	*95*
kaisersoft2	96
ewa_lanczos2sharp	97
hann2	98
ewa_catmull_rom	99
ewa_robidoux	100
blackman2	*101*
ewa_catmull_rom	*102*
bohman2	*103*
ewa_lanczos2	104
blackman2	105
parzen2	*106*
bohman2	107
parzen2	108
mitchell_netravali	*109*
mitchell_netravali	110
cubic_hermite	*111*
ewa_hermite	*112*
cubic_hermite	113
ewa_teepee	*114*
ewa_hermite	115
ewa_teepee	116
bilinear	*117*
bilinear	118
ewa_quadratic_b_spline	119
ewa_quadratic_b_spline	*120*
quadratic_b_spline	121
quadratic_b_spline	*122*
ewa_cubic_b_spline	123
ewa_cubic_b_spline	*124*
cubic_b_spline	125
cubic_b_spline	*126*
nearest	*127.5*
nearest	127.5

Table 6.15: XYZ RMSE ranking of linear light and sRGB upsampling methods as reconstructors of images obtained by linear light downsampling. Enlargement by direct filtering of sRGB values is shown in italics; roman font indicates enlargement through linear light.

upsampler	rank	upsampler	rank	upsampler	rank
hamming4	1	*welch2*	*44*	ewa_lanczos2sharp	87
lanczos3	2	ewa_lanczos_radius4	45	kaisersoft2	88
hann4	3	blackman3	46	hann2	89
bartlett4	4	*kaiser4*	*47*	*ewa_lanczos3*	*90*
cosine3	5	nohalo	48	*kaiser2*	*91*
lanczos4	6	*kaisersoft4*	*49*	*ewa_mitchell_netravali*	*92*
welch3	7	kaiser3	50	*ewa_lanczos4*	*93*
blackman4	8	bohman3	51	ewa_robidoux	94
bohman4	9	*ewa_lanczos_radius3*	*52*	ewa_catmull_rom	95
cosine4	10	kaisersoft3	53	*ewa_lanczos2sharp*	*96*
bartlett3	11	*kaisersharp3*	*54*	ewa_lanczos2	97
hann3	12	*cosine2*	*55*	*kaisersoft2*	*98*
hamming3	13	ewa_robidouxsharp	56	*hann2*	*99*
hamming4	*14*	lanczos2	57	blackman2	100
welch4	15	bartlett2	58	*ewa_robidoux*	*101*
lanczos3	*16*	*ewa_lanczos_radius4*	*59*	bohman2	102
ewa_lanczos4sharpest	17	catmull_rom	60	*ewa_catmull_rom*	*103*
cosine3	*18*	*blackman3*	*61*	*ewa_lanczos2*	*104*
parzen4	19	kaisersharp2	62	parzen2	105
bartlett4	*20*	*bohman3*	*63*	*blackman2*	*106*
hann4	*21*	*kaiser3*	*64*	mitchell_netravali	107
ewa_lanczos3sharpest	22	parzen3	65	*bohman2*	*108*
kaisersharp4	23	*ewa_robidouxsharp*	*66*	cubic_hermite	109
lanczos4	*24*	*kaisersoft3*	*67*	*parzen2*	*110*
welch3	*25*	*bartlett2*	*68*	ewa_teepee	111
cosine4	*26*	*lanczos2*	*69*	ewa_hermite	112
blackman4	*27*	ewa_lanczos2sharpest	70	bilinear	113
hann3	*28*	ewa_lanczos_radius2	71	*mitchell_netravali*	*114*
ewa_lanczos4sharpest	*29*	hamming2	72	*cubic_hermite*	*115*
kaiser4	30	ewa_lanczos3sharp	73	*ewa_hermite*	*116*
bartlett3	*31*	*catmull_rom*	*74*	*ewa_teepee*	*117*
bohman4	*32*	ewa_lanczos4sharp	75	*bilinear*	*118*
welch4	*33*	kaiser2	76	ewa_quadratic_b_spline	119
welch2	34	ewa_lanczos3	77	quadratic_b_spline	120
ewa_lanczos3sharpest	*35*	ewa_mitchell_netravali	78	*ewa_quadratic_b_spline*	*121*
hamming3	*36*	*kaisersharp2*	*79*	*quadratic_b_spline*	*122*
kaisersoft4	37	*parzen3*	*80*	ewa_cubic_b_spline	123
nohalo	*38*	ewa_lanczos4	81	cubic_b_spline	124
ewa_lanczos_radius3	39	*ewa_lanczos2sharpest*	*82*	*ewa_cubic_b_spline*	*125*
parzen4	*40*	*ewa_lanczos3sharp*	*83*	*nearest*	*126.5*
kaisersharp3	41	*ewa_lanczos_radius2*	*84*	nearest	126.5
cosine2	42	*ewa_lanczos4sharp*	*85*	*cubic_b_spline*	*128*
kaisersharp4	*43*	*hamming2*	*86*		

Table 6.16: XYZ RMSE ranking of linear light and sRGB upsampling methods as reconstructors of images obtained by downsampling by direct filtering of sRGB values. Enlargement by direct filtering of sRGB values is shown in italics; roman font indicates enlargement through linear light.

7 Conclusions

EXQUIRES is a powerful platform on which to build a test suite measuring the accuracy of various image resampling methods as well as the impact of various factors on the results. It is Free/Libre and Open Source Software, extensible, highly configurable and well-documented.

Results produced by EXQUIRES establish that the accuracy ranking of an upsampling ("enlargement") image filter depends strongly on the characteristics of the image being enlarged.

When enlarging images which correspond to what one would expect from downsampling, through linear light, high quality sRGB digital photographs with a good quality low pass filter, the most accurate results are likely to be obtained with Welch-, Cosine-, Sinc-(Lanczos) and Hamming Sinc 4-lobe filtering and with the novel Elliptical Weighted Averaging (EWA) with the Catmull-Rom cubic kernel. With such images, 4-lobe methods generally rank higher than 3-lobe methods, which themselves rank higher than 2-lobe or halo-free methods. In addition, tensor methods generally rank higher than EWA methods.

When enlarging images which correspond to what one would expect from downsampling high quality sRGB digital photographs with nearest neighbour interpolation, the above methods are likely to give relatively inaccurate results. In fact, the rankings are nearly reversed. With such images, Mitchell-Netravali cubic spline smoothing, bilinear interpolation, the novel interpolation method Nohalo-LBB (Nohalo face split subdivision with Locally Bounded Bicubic finishing scheme), Hann-windowed Sinc 2-lobe filtering

and filtering with the novel EWA Robidoux tuned Keys cubic are likely to give the most accurate results. When re-enlarging such images, no method with more than two lobes shows up in the top quartile.

Although the rankings obtained with nearest neighbour are extremely different from the rankings obtained with the other downsamplers, namely box filtering, Gaussian blur, tensor Sinc-windowed Sinc filtering and EWA Jinc-windowed Jinc 3-lobe filtering, the rankings obtained with these other downsamplers are well correlated.

The above rankings are based on deviations measured with the RMSE (Root Mean Square Error) in the XYZ linear light colour space. Although commonly used image difference metrics give fairly well correlated rankings, the choice of difference metric significantly affects the rank of an image upsampler, to the extent that one can push most image upsampling methods to a high rank with a "judicious" choice of image difference metric. Only one upsampling method, namely Nohalo-LBB, manages to have a high rank with respect to most image difference metrics, at least when results of re-enlarging images obtained with all the downsamplers, including nearest neighbour, are considered.

The rankings are mostly independent of the choice of resampling ratio as well as of the subject matter of test images, at least with the carefully downsampled high-quality digital photographs and archival quality scan used for testing. Results obtained with different types of image (text, computer graphics, poor quality...) would be interesting to compare.

Whether resampling is performed through linear light or directly using sRGB pixel values has insignificant impact on rankings. However, re-enlarging an image downsampled through linear light using the sRGB pixel values directly gives significantly more accurate results than re-enlarging through linear light. Conversely, re-enlarging through linear light an image downsampled using the sRGB pixel values directly gives significantly more accurate results. In other words, mixed "roundtrip" linear RGB/sRGB toolchains appear to be more accurate than toolchains that resample both down and up using either linear light or sRGB pixel values, at least when used to process and produce sRGB images. Interest-

ingly, the popular Lanczos 3-lobe method rises to the top in both mixed toolchains among methods which have a continuous gradient. Hamming-, Bartlett- and Hann-windowed Sinc 4-lobe filtering, and Cosine-windowed 3-lobe filtering, also rank highly. This all suggests that the choice of colour space through which to enlarge, in relation to the colour space of the input image and output result, is a particularly worthy topic of investigation.

Given that 4-lobe methods rank highly in many of the tests, it would be interesting to add 5- and 6-lobe windowed Sinc to the suite, if only to see how the ranking of windowing functions depends on the number of lobes. Besides testing additional upsampling methods, it would also be interesting to measure the performance of the various methods when the downsized images that are re-enlarged are JPEG compressed or otherwise "corrupted". Nonetheless, the topic most deserving of investigation is the interaction of resampling and colour space choice, with careful consideration given to the machinery used to perform colour space transformations.

A EXQUIRES modules: source code

The following code was written by Adam Turcotte with minor contributions from Nicolas Robidoux. In this and the following appendix, every source file of version 0.9.9.3 of the EXQUIRES test suite is listed. An up-to-date version of the code is in the GitHub repository [147].

A.1 __init__.py

```
#!/usr/bin/env python
# coding: utf-8
#
#  Copyright (c) 2012, Adam Turcotte (adam.turcotte@gmail.com)
#                      Nicolas Robidoux (nicolas.robidoux@gmail.com)
#  License: BSD 2-Clause License
#
#  This file is part of the
#  EXQUIRES (EXtensible QUantitative Image RESampling) test suite
#

"""Package file for **EXQUIRES**."""

__version__ = '0.9.9.3'
```

A.2 aggregate.py

```
#!/usr/bin/env python
# coding: utf-8
#
#  Copyright (c) 2012, Adam Turcotte (adam.turcotte@gmail.com)
#                      Nicolas Robidoux (nicolas.robidoux@gmail.com)
#  License: BSD 2-Clause License
#
#  This file is part of the
#  EXQUIRES (EXtensible QUantitative Image RESampling) test suite
#

"""Aggregate a list of numbers using the specified method.

  **Aggregators:**

    ====== ============================================
```

```
    NAME   DESCRIPTION
    ====== ============================================
    l_1    return the average
    l_2    average the squares and return the square root
    l_4    average the quads and return the fourth root
    l_inf  return the maximum
    ====== ============================================

"""

import inspect

import numpy

from exquires import parsing


class Aggregate(object):

    """This class provide various ways of aggregating error data.

    :param values: numbers to aggregate
    :type values:  `list of numbers`

    """

    def __init__(self, values):
        """Create a new :class:`Aggregate` object."""
        self.values = values

    def l_1(self):
        """Return the average.

        :return: the average
        :rtype:  `float`

        """
        return numpy.average(self.values)

    def l_2(self):
        """Average the squares and return the square root.

        :return: the square root of the average of the squares
        :rtype:  `float`

        """
        return numpy.average(numpy.power(self.values, 2)) ** 0.5

    def l_4(self):
        """Average the quads and return the fourth root.

        :return: the fourth root of the average of the quads
        :rtype:  `float`

        """
        return numpy.average(numpy.power(self.values, 4)) ** 0.25

    def l_inf(self):
        """Return the maximum.

        :return: the maximum
        :rtype:  `float`

        """
        return max(self.values)


def main():
    """Run :ref:`exquires-aggregate`."""
```

```
    # Obtain a list of aggregation methods that can be called.
    aggregators = []
    methods = inspect.getmembers(Aggregate, predicate=inspect.ismethod)
    for method in methods[1:]:
        aggregators.append(method[0])

    # Define the command-line argument parser.
    parser = parsing.ExquiresParser(description=__doc__)
    parser.add_argument('method', metavar='METHOD', choices=aggregators,
                        help='the type of aggregation to use')
    parser.add_argument('values', metavar='NUM', type=float, nargs='+',
                        help='number to include in aggregation')

    # Attempt to parse the command-line arguments.
    args = parser.parse_args()

    # Print the result with 15 digits after the decimal.
    aggregation = Aggregate(args.values)
    print '%.15f' % getattr(aggregation, args.method)()

if __name__ == '__main__':
    main()
```

A.3 compare.py

```
#!/usr/bin/env python
# coding: utf-8
#
#  Copyright (c) 2012, Adam Turcotte (adam.turcotte@gmail.com)
#                      Nicolas Robidoux (nicolas.robidoux@gmail.com)
#  License: BSD 2-Clause License
#
#  This file is part of the
#  EXQUIRES (EXtensible QUantitative Image RESampling) test suite
#

"""Print the result of calling a difference metric on two image files.

  **Difference Metrics:**

    =========== ================================================
    NAME        DESCRIPTION
    =========== ================================================
    srgb_1      :math:`\ell_1` norm in sRGB colour space
    srgb_2      :math:`\ell_2` norm in sRGB colour space
    srgb_4      :math:`\ell_4` norm in sRGB colour space
    srgb_inf    :math:`\ell_\infty` norm in sRGB colour space
    cmc_1       :math:`\ell_1` norm using the CMC(1:1) colour difference
    cmc_2       :math:`\ell_2` norm using the CMC(1:1) colour difference
    cmc_4       :math:`\ell_4` norm using the CMC(1:1) colour difference
    cmc_inf     :math:`\ell_\infty` norm using the CMC(1:1) colour difference
    xyz_1       :math:`\ell_1` norm in XYZ colour space
    xyz_2       :math:`\ell_2` norm in XYZ colour space
    xyz_4       :math:`\ell_4` norm in XYZ colour space
    xyz_inf     :math:`\ell_\infty` norm in XYZ colour space
    blur_1      MSSIM-inspired :math:`\ell_1` norm
    blur_2      MSSIM-inspired :math:`\ell_2` norm
    blur_4      MSSIM-inspired :math:`\ell_4` norm
    blur_inf    MSSIM-inspired :math:`\ell_\infty` norm
    mssim       Mean Structural Similarity Index (MSSIM)
    =========== ================================================

"""

import inspect
```

```
import os
from math import exp

from exquires import parsing


class Metrics(object):

    """This class contains error metrics to be used on sRGB images.

    The :math:`\ell_1`, :math:`\ell_2`, :math:`\ell_4`, and :math:`\ell_\infty`
    metrics are normalized by L, the largest possible pixel value of the input
    images (the lowest is assumed to be 0). The range of output for these
    metrics is [0, 100].

    The MSSIM metric produces output in the range [-1, 1], but it is unlikely
    that a negative value will be produced, as the image features must differ
    greatly. For instance, a pure white image compared with a pure black image
    produces a result slightly greater than 0.

    The CMC and XYZ errors can be slightly outside the range [0, 100], but this
    will not occur for most image pairs.

    .. note::

        By default, a :class:`Metrics` object is configured to operate on
        16-bit images.

    :param image1: first image to compare (reference image)
    :param image2: second image to compare (test image)
    :param L:      highest possible pixel value (default=65535)
    :type image1:  'path'
    :type image2:  'path'
    :type L:       'integer'

    """

    def __init__(self, image1, image2, maxval=65535):
        """Create a new :class:`Metrics` object."""
        vipscc = __import__('vipsCC', globals(), locals(),
                            ['VImage', 'VMask'], -1)
        self.vmask = vipscc.VMask
        self.im1 = vipscc.VImage.VImage(image1)
        self.im2 = vipscc.VImage.VImage(image2)
        self.maxval = maxval
        self.srgb_profile = os.path.join(os.path.dirname(__file__),
                                         'sRGB_IEC61966-2-1_black_scaled.icc')
        self.intent = 1    # IM_INTENT_RELATIVE_COLORIMETRIC

    def srgb_1(self):
        """Compute :math:`\ell_1` error in sRGB colour space.

        The equation for the :math:`\ell_1` error, aka Average Absolute Error
        (AAE), is

        .. math::
            :label: l_1

            \ell_1(x,y) = \\frac{1}{N} \sum_{i=1}^{N} |x_i - y_i|

        where :math:`x` and :math:`y` are the images to compare, each
        consisting of :math:`N` pixels.

        :return: :math:`\ell_1` error
        :rtype:  'float'

        """
        diff = self.im1.subtract(self.im2).abs().avg() / self.maxval
        return diff * 100
```

```
def srgb_2(self):
    """Compute :math:`\ell_2` error in sRGB colour space.

    The equation for the :math:`\ell_2` error, aka Root Mean Squared Error
    (RMSE), is

    .. math::
        :label: l_2

        \ell_2(x,y) = \sqrt{\\frac{1}{N} \sum_{i=1}^{N} (x_i - y_i)^2}

    where :math:`x` and :math:`y` are the images to compare, each
    consisting of :math:`N` pixels.

    :return: :math:`\ell_2` error
    :rtype:  `float`

    """
    diff = self.im1.subtract(self.im2).pow(2).avg() ** 0.5 / self.maxval
    return diff * 100

def srgb_4(self):
    """Compute :math:`\ell_4` error in sRGB colour space.

    The equation for the :math:`\ell_4` error is

    .. math::
        :label: l_4

        \ell_4(x,y) = \sqrt[4]{\\frac{1}{N} \sum_{i=1}^{N} (x_i - y_i)^4}

    where :math:`x` and :math:`y` are the images to compare, each
    consisting of :math:`N` pixels.

    :return: :math:`\ell_4` error
    :rtype:  `float`

    """
    diff = self.im1.subtract(self.im2).pow(4).avg() ** 0.25 / self.maxval
    return diff * 100

def srgb_inf(self):
    """Compute :math:`\ell_\infty` error in sRGB colour space.

    The equation for the :math:`\ell_\infty` error, aka Maximum Absolute
    Error (MAE), is

    .. math::
        :label: l_inf

        \ell_\infty(x,y) = \max_{1 \le i \le N} |x_i - y_i|

    where :math:`x` and :math:`y` are the images to compare, each
    consisting of :math:`N` pixels.

    :return: :math:`\ell_\infty` error
    :rtype:  `float`

    """
    diff = self.im1.subtract(self.im2).abs().max() / self.maxval
    return diff * 100

def mssim(self):
    """Compute the Mean Structural Similarity Index (MSSIM).

    The equation for SSIM is

    .. math::
```

```
        :label: ssim

        SSIM(x,y) = \\frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}
                {(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}

    where :math:`\mu_x` and :math:`\mu_y` are the sample means,
    :math:`\sigma_x` and :math:`\sigma_y` are the standard deviations, and
    :math:`\sigma_{xy}` is the correlation coefficient between images
    :math:`x` and :math:`y`.

    Once the SSIM map is computed, the border is trimmed by 5 pixels and
    the mean is returned.

    This version is slightly more efficient than the method proposed by
    Wang et. al. because it reduces the number of Gaussian blurs from 5 to
    4.

    .. note::

        The images are converted to grayscale before applying
        Gaussian blur. The grayscale conversion is equivalent to taking
        the Y channel in YIQ colour space.

    :return: mean SSIM
    :rtype:  'float'

    """
    # Compute the SSIM constants from the highest possible pixel value.
    const1 = (0.01 * self.maxval) ** 2
    const_sum = const1 + (0.03 * self.maxval) ** 2

    # Create the Gaussian blur mask.
    blur = self.vmask.VDMask(11, 1, 1.0, 0, _get_blurlist())

    # Compute a mask for converting the image to grayscale.
    # Note that the result is equivalent to the Y channel of YIQ.
    rgb2gray = self.vmask.VDMask(3, 1, 1, 0, [0.299, 0.587, 0.114])

    # Convert the image to grayscale using Matlab's approach.
    im1_g = self.im1.recomb(rgb2gray)
    im2_g = self.im2.recomb(rgb2gray)

    # Apply Gaussian blur to the grayscale images.
    im1_b = im1_g.convsep(blur)
    im2_b = im2_g.convsep(blur)

    # Compute the SSIM map.
    tmp1 = im1_g.multiply(im2_g).convsep(blur).lin(2, const_sum)
    tmp2 = im1_g.pow(2).add(im2_g.pow(2)).convsep(blur).lin(1, const_sum)
    tmp3 = im1_b.multiply(im2_b).lin(2, const1)
    tmp4 = im2_b.subtract(im1_b).pow(2).add(tmp3)
    tmp5 = tmp3.multiply(tmp1.subtract(tmp3))
    ssim = tmp5.divide(tmp2.subtract(tmp4).multiply(tmp4))

    # Crop the SSIM map and return the average.
    return ssim.extract_area(5, 5,
                             im1_g.Xsize() - 10, im1_g.Ysize() - 10).avg()

def blur_1(self):
    """Compute MSSIM-inspired :math:`\ell_1` error.

    This method performs the same greyscale conversion, Gaussian blur, and
    cropping as MSSIM, but returns the :math:`\ell_1` error of the cropped
    image.

    See :eq:`l_1` for details on how the blurred images are compared.

    .. note::
```

```
        The images are converted to grayscale before applying
        Gaussian blur. The grayscale conversion is equivalent to taking
        the Y channel in YIQ colour space.

    :return: MSSIM-inspired :math:`\ell_1` error
    :rtype:  `float`

    """

    # Create the Gaussian blur mask.
    blur = self.vmask.VDMask(11, 1, 1.0, 0, _get_blurlist())

    # Compute a mask for converting the image to grayscale.
    # Note that the result is equivalent to the Y channel of YIQ.
    rgb2gray = self.vmask.VDMask(3, 1, 1, 0, [0.299, 0.587, 0.114])

    # Convert the image to grayscale using Matlab's approach.
    im1_g = self.im1.recomb(rgb2gray)
    im2_g = self.im2.recomb(rgb2gray)

    # Apply Gaussian blur, crop the difference and return the l_1 error.
    diff = im1_g.convsep(blur).subtract(im2_g.convsep(blur))
    crop = diff.extract_area(5, 5, im1_g.Xsize() - 10, im1_g.Ysize() - 10)
    return (crop.abs().avg() / self.maxval) * 100

def blur_2(self):
    """Compute MSSIM-inspired :math:`\ell_2` error.

    This method performs the same greyscale conversion, Gaussian blur, and
    cropping as MSSIM, but returns the :math:`\ell_2` error of the cropped
    image.

    See :eq:`l_2` for details on how the blurred images are compared.

    .. note::

        The images are converted to grayscale before applying
        Gaussian blur. The grayscale conversion is equivalent to taking
        the Y channel in YIQ colour space.

    :return: MSSIM-inspired :math:`\ell_2` error
    :rtype:  `float`

    """

    # Create the Gaussian blur mask.
    blur = self.vmask.VDMask(11, 1, 1.0, 0, _get_blurlist())

    # Compute a mask for converting the image to grayscale.
    # Note that the result is equivalent to the Y channel of YIQ.
    rgb2gray = self.vmask.VDMask(3, 1, 1, 0, [0.299, 0.587, 0.114])

    # Convert the image to grayscale using Matlab's approach.
    im1_g = self.im1.recomb(rgb2gray)
    im2_g = self.im2.recomb(rgb2gray)

    # Apply Gaussian blur, crop the difference and return the l_2 error.
    diff = im1_g.convsep(blur).subtract(im2_g.convsep(blur))
    crop = diff.extract_area(5, 5, im1_g.Xsize() - 10, im1_g.Ysize() - 10)
    return (crop.pow(2).avg() ** 0.5 / self.maxval) * 100

def blur_4(self):
    """Compute MSSIM-inspired :math:`\ell_4` error.

    This method performs the same greyscale conversion, Gaussian blur, and
    cropping as MSSIM, but returns the :math:`\ell_4` error of the cropped
    image.

    See :eq:`l_4` for details on how the blurred images are compared.
```

```
    .. note::

        The images are converted to grayscale before applying
        Gaussian blur. The grayscale conversion is equivalent to taking
        the Y channel in YIQ colour space.

    :return: MSSIM-inspired :math:`\ell_4` error
    :rtype:  `float`

    """

    # Create the Gaussian blur mask.
    blur = self.vmask.VDMask(11, 1, 1.0, 0, _get_blurlist())

    # Compute a mask for converting the image to grayscale.
    # Note that the result is equivalent to the Y channel of YIQ.
    rgb2gray = self.vmask.VDMask(3, 1, 1, 0, [0.299, 0.587, 0.114])

    # Convert the image to grayscale using Matlab's approach.
    im1_g = self.im1.recomb(rgb2gray)
    im2_g = self.im2.recomb(rgb2gray)

    # Apply Gaussian blur, crop the difference and return the l_4 error.
    diff = im1_g.convsep(blur).subtract(im2_g.convsep(blur))
    crop = diff.extract_area(5, 5, im1_g.Xsize() - 10, im1_g.Ysize() - 10)
    return (crop.pow(4).avg() ** 0.25 / self.maxval) * 100

def blur_inf(self):
    """Compute MSSIM-inspired :math:`\ell_\infty` error.

    This method performs the same greyscale conversion, Gaussian blur, and
    cropping as MSSIM, but returns the :math:`\ell_\infty` error of the
    cropped image.

    See :eq:`l_inf` for details on how the blurred images are compared.

    .. note::

        The images are converted to grayscale before applying
        Gaussian blur. The grayscale conversion is equivalent to taking
        the Y channel in YIQ colour space.

    :return: MSSIM-inspired :math:`\ell_\infty` error
    :rtype:  `float`

    """

    # Create the Gaussian blur mask.
    blur = self.vmask.VDMask(11, 1, 1.0, 0, _get_blurlist())

    # Compute a mask for converting the image to grayscale.
    # Note that the result is equivalent to the Y channel of YIQ.
    rgb2gray = self.vmask.VDMask(3, 1, 1, 0, [0.299, 0.587, 0.114])

    # Convert the image to grayscale using Matlab's approach.
    im1_g = self.im1.recomb(rgb2gray)
    im2_g = self.im2.recomb(rgb2gray)

    # Apply Gaussian blur, crop the difference and return the l_inf error.
    diff = im1_g.convsep(blur).subtract(im2_g.convsep(blur))
    crop = diff.extract_area(5, 5, im1_g.Xsize() - 10, im1_g.Ysize() - 10)
    return (crop.abs().max() / self.maxval) * 100

def cmc_1(self):
    """Compute :math:`\ell_1` error in Uniform Colour Space (UCS).

    This method imports the images into Lab colour space, then calculates
    delta-E CMC(1:1) and returns the average.
```

```
        See :eq:`l_1` for details on how the standard :math:`\ell_1` norm is
        computed.

        :return: :math:`\ell_1` error in Uniform Colour Space (UCS)
        :rtype:  `float`

        """
        lab1 = self.im1.icc_import(self.srgb_profile, self.intent)
        lab2 = self.im2.icc_import(self.srgb_profile, self.intent)
        return lab1.dECMC_fromLab(lab2).avg()

    def cmc_2(self):
        """Compute :math:`\ell_2` error in Uniform Colour Space (UCS).

        This method imports the images into Lab colour space, then calculates
        delta-E CMC(1:1) and returns the :math:`\ell_2` norm.

        See :eq:`l_2` for details on how the standard :math:`\ell_2` norm is
        computed.

        :return: :math:`\ell_2` error in Uniform Colour Space (UCS)
        :rtype:  `float`

        """
        lab1 = self.im1.icc_import(self.srgb_profile, self.intent)
        lab2 = self.im2.icc_import(self.srgb_profile, self.intent)
        return lab1.dECMC_fromLab(lab2).pow(2).avg() ** 0.5

    def cmc_4(self):
        """Compute :math:`\ell_4` error in Uniform Colour Space (UCS).

        This method imports the images into Lab colour space, then calculates
        delta-E CMC(1:1) and returns the :math:`\ell_4` norm.

        See :eq:`l_4` for details on how the standard :math:`\ell_4` norm is
        computed.

        :return: :math:`\ell_4` error in Uniform Colour Space (UCS)
        :rtype:  `float`

        """
        lab1 = self.im1.icc_import(self.srgb_profile, self.intent)
        lab2 = self.im2.icc_import(self.srgb_profile, self.intent)
        return lab1.dECMC_fromLab(lab2).pow(4).avg() ** 0.25

    def cmc_inf(self):
        """Compute :math:`\ell_\infty` error in Uniform Colour Space (UCS).

        This method imports the images into Lab colour space, then calculates
        delta-E CMC(1:1) and returns the :math:`\ell_\infty` norm.

        See :eq:`l_inf` for details on how the standard :math:`\ell_\infty`
        norm is computed.

        :return: :math:`\ell_\infty` error in Uniform Colour Space (UCS)
        :rtype:  `float`

        """
        lab1 = self.im1.icc_import(self.srgb_profile, self.intent)
        lab2 = self.im2.icc_import(self.srgb_profile, self.intent)
        return lab1.dECMC_fromLab(lab2).max()

    def xyz_1(self):
        """Compute :math:`\ell_1` error in XYZ Colour Space.

        This method imports the images into XYZ colour space, then calculates
        the :math:`\ell_1` error.
```

```
        See :eq:`l_1` for details on how the standard :math:`\ell_1` norm is
        computed.

        :return: :math:`\ell_1` error in XYZ Colour Space
        :rtype:  `float`

        """
        xyz1 = self.im1.icc_import(self.srgb_profile, self.intent).Lab2XYZ()
        xyz2 = self.im2.icc_import(self.srgb_profile, self.intent).Lab2XYZ()
        return xyz1.subtract(xyz2).abs().avg()

    def xyz_2(self):
        """Compute :math:`\ell_2` error in XYZ Colour Space.

        This method imports the images into XYZ colour space, then calculates
        the :math:`\ell_2` error.

        See :eq:`l_2` for details on how the standard :math:`\ell_2` norm is
        computed.

        :return: :math:`\ell_2` error in XYZ Colour Space
        :rtype:  `float`

        """
        xyz1 = self.im1.icc_import(self.srgb_profile, self.intent).Lab2XYZ()
        xyz2 = self.im2.icc_import(self.srgb_profile, self.intent).Lab2XYZ()
        return xyz1.subtract(xyz2).pow(2).avg() ** 0.5

    def xyz_4(self):
        """Compute :math:`\ell_4` error in XYZ Colour Space.

        This method imports the images into XYZ colour space, then calculates
        the :math:`\ell_4` error.

        See :eq:`l_4` for details on how the standard :math:`\ell_4` norm is
        computed.

        :return: :math:`\ell_4` error in XYZ Colour Space
        :rtype:  `float`

        """
        xyz1 = self.im1.icc_import(self.srgb_profile, self.intent).Lab2XYZ()
        xyz2 = self.im2.icc_import(self.srgb_profile, self.intent).Lab2XYZ()
        return xyz1.subtract(xyz2).pow(4).avg() ** 0.25

    def xyz_inf(self):
        """Compute :math:`\ell_\infty` error in XYZ Colour Space.

        This method imports the images into XYZ colour space, then calculates
        the :math:`\ell_\infty` error.

        See :eq:`l_inf` for details on how the standard :math:`\ell_\infty`
        norm is computed.

        :return: :math:`\ell_\infty` error in XYZ Colour Space
        :rtype:  `float`

        """
        xyz1 = self.im1.icc_import(self.srgb_profile, self.intent).Lab2XYZ()
        xyz2 = self.im2.icc_import(self.srgb_profile, self.intent).Lab2XYZ()
        return xyz1.subtract(xyz2).abs().max()


def _get_blurlist():
    """Private method to return a Gaussian blur mask.

    .. note::

        This is a private function called by :meth:`~Metrics.blur_1`,
```

```
        :meth:`~Metrics.blur_2`, :meth:`~Metrics.blur_4`,
        :meth:`~Metrics.blur_inf`, and :meth:`~Metrics.mssim`.

    """

    # Compute the raw Gaussian blur coefficients.
    blur_sigma = 1.5
    blur_divisor = 2 * blur_sigma * blur_sigma
    rawblur1 = exp(-1 / blur_divisor)
    rawblur2 = exp(-4 / blur_divisor)
    rawblur3 = exp(-9 / blur_divisor)
    rawblur4 = exp(-16 / blur_divisor)
    rawblur5 = exp(-25 / blur_divisor)

    # Normalize the raw Gaussian blur coefficients.
    rawblursum = rawblur1 + rawblur2 + rawblur3 + rawblur4 + rawblur5
    blur_normalizer = 2 * rawblursum + 1
    blur0 = 1 / blur_normalizer
    blur1 = rawblur1 / blur_normalizer
    blur2 = rawblur2 / blur_normalizer
    blur3 = rawblur3 / blur_normalizer
    blur4 = rawblur4 / blur_normalizer
    blur5 = rawblur5 / blur_normalizer

    # Return the Gaussian blur mask as a list.
    return [blur5, blur4, blur3, blur2, blur1,
            blur0, blur1, blur2, blur3, blur4, blur5]


def main():
    """Run :ref:`exquires-compare`."""

    # Obtain a list of error metrics that can be called.
    metrics = []
    methods = inspect.getmembers(Metrics, predicate=inspect.ismethod)
    for method in methods[1:]:
        metrics.append(method[0])

    # Define the command-line argument parser.
    parser = parsing.ExquiresParser(description=__doc__)
    parser.add_argument('metric', type=str, metavar='METRIC', choices=metrics,
                        help='the difference metric to use')
    parser.add_argument('image1', type=str, metavar='IMAGE_1',
                        help='the first image to compare')
    parser.add_argument('image2', type=str, metavar='IMAGE_2',
                        help='the second image to compare')
    parser.add_argument('-m', '--maxval', type=int, metavar='MAX_LEVEL',
                        default=65535,
                        help='the maximum pixel value (default: 65535)')

    # Attempt to parse the command-line arguments.
    args = parser.parse_args()

    # Attempt to call the chosen metric on the specified images.
    vipscc = __import__('vipsCC', globals(), locals(), ['VError'], -1)
    try:
        # Print the result with 15 digits after the decimal.
        metric = Metrics(args.image1, args.image2, args.maxval)
        print '%.15f' % getattr(metric, args.metric)()
    except vipscc.VError.VError, error:
        parser.error(str(error))

if __name__ == '__main__':
    main()
```

A.4 correlate.py

```
#!/usr/bin/env python
# coding: utf-8
#
#  Copyright (c) 2012, Adam Turcotte (adam.turcotte@gmail.com)
#                      Nicolas Robidoux (nicolas.robidoux@gmail.com)
#  License: BSD 2-Clause License
#
#  This file is part of the
#  EXQUIRES (EXtensible QUantitative Image RESampling) test suite
#

"""Produce a Spearman's rank cross-correlation matrix for the specified group.

By default, the :option:`-M`/:option:`--metric` option is selected.
You can select one of the following cross-correlation groups:

    * :option:`-I`/:option:`--image`
    * :option:`-D`/:option:`--down`
    * :option:`-R`/:option:`--ratio`
    * :option:`-M`/:option:`--metric`

You can also select which upsamplers to consider when computing the matrix
by using the :option:`-U`/:option:`--up` option.

"""

import argparse

import numpy

from exquires import database, parsing, stats


def _get_group_and_ranks(args):
    """Return the correlation group and ranks.

    .. note::

        This is a private function called by :func:`_print_matrix`.

    :param args:             arguments
    :param args.dbase_file:  database file
    :param args.image:       selected image names
    :param args.down:        selected downsampler names
    :param args.ratio:       selected ratios
    :param args.up:          selected upsampler names
    :param args.metric:      selected metric names
    :param args.metrics_d:   all metric names
    :param args.file:        output file
    :param args.digits:      number of digits to print
    :param args.latex:       `True` if printing a LaTeX-formatted table
    :param args.key:         key for the correlation group
    :type args:              :class:`argparse.Namespace`
    :type args.dbase_file:   `path`
    :type args.image:        `list of strings`
    :type args.down:         `list of strings`
    :type args.ratio:        `list of strings`
    :type args.up:           `list of strings`
    :type args.metric:       `list of strings`
    :type args.metrics_d:    `dict`
    :type args.file:         `path`
    :type args.digits:       `integer`
    :type args.latex:        `boolean`
    :type args.key:          `string`

    :return:                 the group and ranks
    :rtype:                  `string`, `list of lists`
```

```
    """
    # Create a list of the sorting options for each metric.
    metrics_desc = []
    for metric in args.metric:
        metrics_desc.append(int(args.metrics_d[metric][2]))

    # See if the config file has been poorly edited by the user.
    if not (len(args.image) or len(args.down) or
            len(args.ratio) or len(args.up) or len(args.metric)):
        return

    # Open the database connection.
    dbase = database.Database(args.dbase_file)

    # Determine which cross-correlation to perform.
    group = getattr(args, args.key)
    ranks = []
    if args.key in ('image', 'down', 'ratio'):
        # Setup table arguments.
        table_args = argparse.Namespace()
        table_args.image = None
        table_args.down = None
        table_args.ratio = None

        for item in group:
            # Setup the tables to access.
            setattr(table_args, args.key, [item])
            agg_table = stats.get_aggregate_table(
                dbase, args.up, args.metrics_d, dbase.get_tables(table_args)
            )

            if ranks:
                col = stats.get_merged_ranks(agg_table, metrics_desc, 0)
                for i in range(0, len(args.up)):
                    ranks[i].append(col[i][1])
            else:
                ranks = stats.get_merged_ranks(agg_table, metrics_desc, 0)

    else:  # Cross-correlation group is 'metric'
        # Get the rank table.
        agg_table = stats.get_aggregate_table(
            dbase, args.up, args.metrics_d, dbase.get_tables(args)
        )
        ranks = stats.get_ranks(agg_table, metrics_desc, 0)

    # Close the database connection.
    dbase.close()

    # Return the group and ranks.
    return group, ranks


def _print_matrix(args):
    """Print a cross-correlation matrix from aggregate image comparison data.

    .. note::

        This is a private function called by :func:`main`.

    :param args:             arguments
    :param args.dbase_file: database file
    :param args.image:       selected image names
    :param args.down:        selected downsampler names
    :param args.ratio:       selected ratios
    :param args.up:          selected upsampler names
    :param args.metric:      selected metric names
    :param args.metrics_d:   all metric names
    :param args.file:        output file
    :param args.digits:      number of digits to print
```

```
    :param args.latex:      'True' if printing a LaTeX-formatted table
    :param args.key:        key for the correlation group
    :param args.anchor:     row/column to order the matrix by
    :type args:             :class:'argparse.Namespace'
    :type args.dbase_file:  'path'
    :type args.image:       'list of strings'
    :type args.down:        'list of strings'
    :type args.ratio:       'list of strings'
    :type args.up:          'list of strings'
    :type args.metric:      'list of strings'
    :type args.metrics_d:   'dict'
    :type args.file:        'path'
    :type args.digits:      'integer'
    :type args.latex:       'boolean'
    :type args.key:         'string'
    :type args.anchor:      'string'

    """
    # Get the correlation group and ranks table.
    group, ranks = _get_group_and_ranks(args)

    # Compute the correlation coefficient matrix.
    matrix = numpy.identity(len(group))
    xbar = (len(args.up) + 1) * 0.5
    for i, mrow1 in enumerate(matrix):
        for j, mrow2 in enumerate(matrix[i + 1:], i + 1):
            # Compute the numerator and denominator.
            coeff = [0, 0, 0]
            for row in [rank_row[1:] for rank_row in ranks]:
                coeff[0] += (row[i] - xbar) * (row[j] - xbar)
                coeff[1] += (row[i] - xbar) ** 2
                coeff[2] += (row[j] - xbar) ** 2

            # Compute the correlation coefficient.
            mrow1[j] = mrow2[i] = (
                    coeff[0] / ((coeff[1] * coeff[2]) ** 0.5)
            )

    # Deal with -a/--anchor option.
    if args.anchor:
        sort_order = matrix[group.index(args.anchor)].argsort()[::-1]
        group = [group[i] for i in sort_order]
        matrix_sorted = numpy.identity(len(group))
        for i, row in enumerate(sort_order):
            for j, col in enumerate(sort_order):
                matrix_sorted[i, j] = matrix[row, col]
        matrix = matrix_sorted

    # Pass the coefficient matrix to the appropriate table printer.
    if args.latex:
        stats.print_latex(matrix, args, group, True)
    else:
        stats.print_normal(matrix, args, group, True)


def main():
    """Run :ref:'exquires-correlate'.

    Parse the command-line arguments and print the cross-correlation matrix.

    """
    _print_matrix(parsing.StatsParser(__doc__, True).parse_args())

if __name__ == '__main__':
    main()
```

A.5 database.py

```
#!/usr/bin/env python
# coding: utf-8
#
#  Copyright (c) 2012, Adam Turcotte (adam.turcotte@gmail.com)
#                      Nicolas Robidoux (nicolas.robidoux@gmail.com)
#  License: BSD 2-Clause License
#
#  This file is part of the
#  EXQUIRES (EXtensible QUantitative Image RESampling) test suite
#

"""Provides an interface to the sqlite3 image error database."""

import sqlite3


class Database:

    """This class provides an interface to the sqlite3 image error database.

    The database stores error data computed by :ref:`exquires-run` and
    :ref:`exquires-update`. This data is retrieved and used to compute the
    output given by :ref:`exquires-report` and :ref:`exquires-correlate`.

    :param dbasefile: database file to connect to
    :type dbasefile:  `path`

    """

    def __init__(self, dbasefile):
        """Create a new :class:`Database` object."""
        self.dbase = sqlite3.connect(dbasefile,
                                     detect_types=sqlite3.PARSE_DECLTYPES)
        self.dbase.row_factory = sqlite3.Row
        self.dbase.text_factory = str
        self.sql_do('CREATE TABLE IF NOT EXISTS TABLEDATA (name TEXT PRIMARY'
                    ' KEY, image TEXT, downsampler TEXT, ratio TEXT )')

    def sql_do(self, sql, params=()):
        """Perform an operation on the database and commit the changes.

        :param sql:    SQL statement to execute and commit
        :param params: values to fill wildcards in the SQL statement
        :type sql:     `string`
        :type params:  `list of values`

        """
        self.dbase.execute(sql, params)
        self.dbase.commit()

    def __create_table(self, name, metrics):
        """Private method used to create a new database table.

        .. note::

            This is a private method called by :meth:`add_table` and
            :meth:`backup_table`.

        :param name:    name of the table to create
        :param metrics: error metrics to compute (the table columns)
        :type name:     `string`
        :type metrics:  `list of strings`

        """
        sql = 'CREATE TABLE {} ( upsampler TEXT PRIMARY KEY'.format(name)
```

```
        for metric in metrics:
            sql += ', {} DOUBLE'.format(metric)
        sql += ' )'
        self.sql_do(sql)

    def add_table(self, image, downsampler, ratio, metrics):
        """Add a new table to the database.

        Each table is defined in the following way:
            1. image, downsampler, and ratio define the table name
            2. metrics define the columns of the table

        To keep track of each table in terms of the image, downsampler, and
        ratio that defines it, an entry is created in the TABLEDATA table.

        :param image:       name of the image
        :param downsampler: name of the downsampler
        :param ratio:       resampling ratio
        :param metrics:     names of the metrics
        :type image:        `string`
        :type downsampler:  `string`
        :type ratio:        `string`
        :type metrics:      `list of strings`

        :return:            the table name
        :rtype:             `string`

        """
        # Create table.
        name = '_'.join([image, downsampler, ratio])
        self.__create_table(name, metrics)

        # Add table details to master table (TABLEDATA).
        row = dict(name=name, image=image,
                   downsampler=downsampler, ratio=ratio)
        self.insert('TABLEDATA', row)
        self.dbase.commit()
        return name

    def backup_table(self, name, metrics):
        """Backup an existing image error table.

        :param name:    name of the table to backup
        :param metrics: error metrics (columns) to backup
        :type name:     `string`
        :type metrics:  `list of strings`

        :return:        name of the backup table
        :rtype:         `string`

        """
        backup_name = '_'.join([name, 'bak'])
        self.sql_do('ALTER TABLE {} RENAME TO {}'.format(name, backup_name))
        self.__create_table(name, metrics)
        return backup_name

    def get_tables(self, args):
        """Return table names for these images, downsamplers, and ratios.

        :param args:       arguments
        :param args.image: names of images
        :param args.down:  names of downsamplers
        :param args.ratio: resampling ratios
        :type args:        :class:`argparse.Namespace`
        :type args.image:  `list of strings`
        :type args.down:   `list of strings`
        :type args.ratio:  `list of strings`

        :return:           names of the tables
```

```
    :rtype:             `list of strings`

    """
    # Start assembling an SQL query for the specified tables.
    query = 'SELECT name FROM TABLEDATA WHERE ('

    # Append the image names.
    if args.image:
        for image in args.image:
            query = ' '.join([query, 'image = \'{}\' OR'.format(image)])
        query = ''.join([query.rstrip(' OR'), ')'])

    # Append the downsampler names.
    if args.down:
        if args.image:
            query = ' '.join([query, 'AND ('])
        for downsampler in args.down:
            downsampler_str = 'downsampler = \'{}\' OR'.format(downsampler)
            query = ' '.join([query, downsampler_str])
        query = ''.join([query.rstrip(' OR'), ')'])

    # Append the ratios.
    if args.ratio:
        if (args.image or args.down):
            query = ' '.join([query, 'AND ('])
        for ratio in args.ratio:
            query = ' '.join([query, 'ratio = \'{}\' OR'.format(ratio)])
        query = ''.join([query.rstrip(' OR'), ')'])

    # Return the table names.
    return [table[0] for table in self.sql_fetchall(query)]

def drop_backup(self, name):
    """Drop a backup table once it is no longer needed.

    :param name: name of the backup table to drop
    :type name:  `string`

    """
    self.sql_do('DROP TABLE {}'.format(name))

def drop_tables(self, images, downsamplers, ratios):
    """Drop database tables.

    All tables defined by any of the images, downsamplers, or ratios are
    dropped. The TABLEDATA table is updated to reflect these changes.

    :param images:       names of the images
    :param downsamplers: names of the downsamplers
    :param ratios:       resampling ratios
    :type images:        `list of strings`
    :type downsamplers:  `list of strings`
    :type ratios:        `list of strings`

    """
    if len(images) or len(downsamplers) or len(ratios):
        query = 'SELECT name FROM TABLEDATA WHERE'
        for image in images:
            query = query + ' image = \'{}\' OR'.format(image)
        for downsampler in downsamplers:
            query = query + ' downsampler = \'{}\' OR'.format(downsampler)
        for ratio in ratios:
            query = query + ' ratio = \'{}\' OR'.format(ratio)
        query = query.rstrip(' OR')
        cursor = self.dbase.cursor()
        cursor.execute(query)
        names = [(table[0],) for table in cursor.fetchall()]

        # Delete the rows from TABLEDATA and drop the tables by name.
```

```
        self.dbase.executemany('DELETE FROM TABLEDATA WHERE name = ?',
                               names)
        for name in names:
            self.dbase.execute(' '.join(['DROP TABLE', name[0]]))
        self.dbase.commit()

def sql_fetchall(self, sql, params=()):
    """Fetch all rows for the specified SQL query.

    :param sql:    SQL query to execute
    :param params: values to fill wildcards in the SQL statement
    :type sql:     'string'
    :type params:  'list of values'

    :return:       rows specified by the SQL query
    :rtype:        'list of dicts'

    """
    cursor = self.dbase.execute(sql, params)
    return cursor.fetchall()

def get_error_data(self, table, upsampler, metrics_str):
    """Return a filtered row of error data.

    For the upsampler row in the table, return a dictionary containing only
    the error data for the specified metrics.

    :param table:       name of the table to query
    :param upsampler:   name of the upsampler (row) to acquire data from
    :param metrics_str: metrics to return error data for (comma-separated)
    :type table:        'string'
    :type upsampler:    'string'
    :type metrics_str:  'string'

    :return:            the filtered row of error data
    :rtype:             'dict'

    """
    sql = 'SELECT upsampler,{} FROM {} WHERE upsampler=\'{}\''
    query = sql.format(metrics_str, table, upsampler)
    cursor = self.dbase.execute(query)
    return dict(cursor.fetchone())

def insert(self, table, row):
    """Insert a single row into the table, or update if it exists.

    :param table: name of the table
    :param row:   row data to insert
    :type table:  'string'
    :type row:    'dict'

    """
    keys = sorted(row.keys())
    values = [row[v] for v in keys]
    query = 'INSERT OR REPLACE INTO {} ({}) VALUES ({})'.format(
        table, ', '.join(keys), ', '.join('?' for i in range(len(values))))
    self.dbase.execute(query, values)
    self.dbase.commit()

def delete(self, table, upsampler):
    """Delete a row from the table.

    :param table:     name of the table to delete from
    :param upsampler: upsampler (row) to remove from the table
    :type table:      'string'
    :type upsampler:  'string'

    """
    query = 'DELETE FROM {} where upsampler = ?'.format(table)
```

```
        self.dbase.execute(query, [upsampler])
        self.dbase.commit()

    def close(self):
        """Close the connection to the database."""
        self.dbase.close()
```

A.6 new.py

```
#!/usr/bin/env python
# coding: utf-8
#
#  Copyright (c) 2012, Adam Turcotte (adam.turcotte@gmail.com)
#                      Nicolas Robidoux (nicolas.robidoux@gmail.com)
#  License: BSD 2-Clause License
#
#  This file is part of the
#  EXQUIRES (EXtensible QUantitative Image RESampling) test suite
#

"""Generate a new project file to use with :ref:`exquires-run`.

The project file is used to specify the following components of the suite:

    * Images ( sRGB TIFF | 16 bits/sample (48/pixel) | 840x840 pixels )
    * Downsamplers
    * Resampling Ratios
    * Upsamplers
    * Difference Metrics

For the specified project name and list of images, a default project file will
be created with the name :file:`PROJECT.ini`, where :file:`PROJECT` is a name
specified using the :option:`-p`\:option:`--proj` option. If a name is not
specified, the default name is :file:`project1`.

Use the :option:`-I`\:option:`--image` option to provide a list of images to
include in the project file. If no images are specified, a default image
(:file:`wave.tif`) is included in the project file.

Manually edit this file to customize your project.

"""

from os import path

from configobj import ConfigObj

from exquires import parsing


def _magick(method, **kwargs):
    """Return an ImageMagick resize command as a string.

    Blur and Kaiser beta values are passed as strings to avoid truncation.

    .. note::

        This is a private function called by :func:`_add_default_downsamplers`,
        :func:`_std_int_lin_tensor_mtds_1`, :func:`_std_int_lin_tensor_mtds_2`,
        :func:`_novel_int_lin_flt_mtds`, :func:`_std_nonint_lin_tensor_mtds`,
        :func:`_std_int_ewa_lin_flt_mtds`,
        :func:`_std_nonint_ewa_lin_flt_mtds`, and
        :func:`_novel_nonint_ewa_lin_flt_mtds`.

    :param method: method to use with `-resize` or `-distort Resize`
    :param lin:    `True` if using a linear method
```

```
    :param dist:    `True` if using a `-distort Resize` method
    :param lobes:   number of lobes
    :param blur:    blur value
    :param beta:    beta value for Kaiser method
    :type method:   `string`
    :type lin:      `boolean`
    :type dist:     `boolean`
    :type lobes:    `integer`
    :type blur:     `string`
    :type beta:     `string`

    :return:        the ImageMagick command
    :rtype:         `string`

    """

    # Setup keyword arguments.
    lin = kwargs.get('lin', False)
    dist = kwargs.get('dist', False)
    interp = kwargs.get('interp', False)
    lobes = kwargs.get('lobes', 0)
    blur = kwargs.get('blur', None)
    beta = kwargs.get('beta', None)

    # Create and return command string.
    cmd = 'convert {0}'
    if lin:
        cmd = ' '.join([cmd, '-colorspace RGB'])
    if method:
        cmd = ' '.join([cmd, '-filter', method])
    if lobes:
        cmd = ''.join([cmd, ' -define filter:lobes=', str(lobes)])
    if blur:
        cmd = ''.join([cmd, ' -define filter:blur=', blur])
    if beta:
        cmd = ''.join([cmd, ' -define filter:kaiser-beta=', beta])
    if interp:
        cmd = ' '.join([cmd, '-interpolative-resize {3}x{3}'])
    elif dist:
        cmd = ' '.join([cmd, '-distort Resize {3}x{3}'])
    else:
        cmd = ' '.join([cmd, '-resize {3}x{3}'])
    if lin:
        cmd = ' '.join([cmd, '-colorspace sRGB'])
    return ' '.join([cmd, '-strip {1}'])


def _metric(method, aggregator, sort):
    """Return 3-element list defining a metric, an aggregator and a sort order.

    .. note::

        This is a private function called by :func:`_add_default_metrics`.

    :param method:     image comparison metric (see :ref:`exquires-compare`)
    :param aggregator: data aggregator (see :ref:`exquires-aggregate`)
    :param sort:       best-to-worst order (`0`: ascending, `1`: descending)
    :type method:      `string`
    :type aggregator:  `string`
    :type sort:        `integer`

    :return:           metric, aggregator, and sort order
    :rtype:            `list`

    """

    return [' '.join(['exquires-compare', method, '{0} {1}']),
            ' '.join(['exquires-aggregate', aggregator, '{0}']), sort]
```

```
def _add_default_images(ini, image):
    """Add the default images to the specified :file:`.ini` file.

    .. note::

        This is a private function called by :func:`main`.

    :param ini: the :file:`.ini` file to modify
    :type ini:  :class:`configobj.ConfigObj`

    """
    # Define the list of images to be resampled.
    key = 'Images'
    ini[key] = {}
    ini.comments[key] = [
        'TEST IMAGES',
        'Images are 16-bit sRGB TIFFs with a width and height of 840 pixels.',
        'Any images that are added must conform to this standard.'
    ]
    for img in image:
        ini[key][path.splitext(path.basename(img))[0]] = path.abspath(img)


def _add_default_ratios(ini):
    """Add the default ratios to the specified :file:`.ini` file.

    .. note::

        This is a private function called by :func:`main`.

    :param ini: the :file:`.ini` file to modify
    :type ini:  :class:`configobj.ConfigObj`

    """
    # Define the list of resampling ratios to use.
    ratios = 'Ratios'
    ini[ratios] = {}
    ini.comments[ratios] = [
        '', 'RESAMPLING RATIOS',
        'The test images are downsampled to the specified sizes.',
        'Each size is obtained by dividing 840 by the ratio.'
    ]
    ini[ratios]['2'] = '420'
    ini[ratios]['3'] = '280'
    ini[ratios]['4'] = '210'
    ini[ratios]['5'] = '168'
    ini[ratios]['6'] = '140'
    ini[ratios]['7'] = '120'
    ini[ratios]['8'] = '105'


def _add_default_downsamplers(ini):
    """Add the default downsamplers to the specified :file:`.ini` file.

    .. note::

        This is a private function called by :func:`main`.

    :param ini: the :file:`.ini` file to modify
    :type ini:  :class:`configobj.ConfigObj`

    """
    # Define the list of downsamplers to use.
    downs = 'Downsamplers'
    ini[downs] = {}
    ini.comments[downs] = [
        '', 'DOWNSAMPLING COMMANDS',
        'To add a downsampler, provide the command to execute it.',
```

```
        'The command can make use of the following replacement fields:',
        '{0} = input image',
        '{1} = output image',
        '{2} = downsampling ratio',
        '{3} = downsampled size (width or height)',
        '',
        'WARNING: Be sure to use a unique name for each downsampler.'
    ]
    ini[downs]['box_srgb'] = _magick('Box')
    ini[downs]['box_linear'] = _magick('Box', lin=True)
    ini[downs]['gaussian_srgb'] = _magick('Gaussian')
    ini[downs]['gaussian_linear'] = _magick('Gaussian', lin=True)
    ini[downs]['ewa_lanczos3_srgb'] = _magick('Lanczos', dist=True)
    ini[downs]['ewa_lanczos3_linear'] = _magick('Lanczos', dist=True, lin=True)
    ini[downs]['lanczos3_srgb'] = _magick('Lanczos')
    ini[downs]['lanczos3_linear'] = _magick('Lanczos', lin=True)
    ini[downs]['nearest_srgb'] = _magick('Triangle', interp=True)
    ini[downs]['nearest_linear'] = _magick('Triangle', interp=True, lin=True)


def _std_int_lin_tensor_mtds_1(ini_ups):
    """Add the 1st part of the standard interpolatory linear tensor methods.

    .. note::

        This is a private function called by :func:`_add_default_upsamplers`.

    :param ini_ups: upsamplers for the specified :file:`.ini` file
    :type ini_ups:  `dict`

    """
    ini_ups['nearest_srgb'] = _magick('Point')
    ini_ups['nearest_linear'] = _magick('Point', lin=True)
    ini_ups['bilinear_srgb'] = _magick('Triangle')
    ini_ups['bilinear_linear'] = _magick('Triangle', lin=True)
    ini_ups['cubic_hermite_srgb'] = _magick('Hermite')
    ini_ups['cubic_hermite_linear'] = _magick('Hermite', lin=True)
    ini_ups['catmull_rom_srgb'] = _magick('Catrom')
    ini_ups['catmull_rom_linear'] = _magick('Catrom', lin=True)
    ini_ups['lanczos2_srgb'] = _magick('Lanczos2')
    ini_ups['lanczos2_linear'] = _magick('Lanczos2', lin=True)
    ini_ups['lanczos3_srgb'] = _magick('Lanczos')
    ini_ups['lanczos3_linear'] = _magick('Lanczos', lin=True)
    ini_ups['lanczos4_srgb'] = _magick('Lanczos', lobes=4)
    ini_ups['lanczos4_linear'] = _magick('Lanczos', lobes=4, lin=True)
    ini_ups['bartlett2_srgb'] = _magick('Bartlett', lobes=2)
    ini_ups['bartlett2_linear'] = _magick('Bartlett', lobes=2, lin=True)
    ini_ups['bartlett3_srgb'] = _magick('Bartlett', lobes=3)
    ini_ups['bartlett3_linear'] = _magick('Bartlett', lobes=3, lin=True)
    ini_ups['bartlett4_srgb'] = _magick('Bartlett')
    ini_ups['bartlett4_linear'] = _magick('Bartlett', lin=True)
    ini_ups['blackman2_srgb'] = _magick('Blackman', lobes=2)
    ini_ups['blackman2_linear'] = _magick('Blackman', lobes=2, lin=True)
    ini_ups['blackman3_srgb'] = _magick('Blackman', lobes=3)
    ini_ups['blackman3_linear'] = _magick('Blackman', lobes=3, lin=True)
    ini_ups['blackman4_srgb'] = _magick('Blackman')
    ini_ups['blackman4_linear'] = _magick('Blackman', lin=True)
    ini_ups['bohman2_srgb'] = _magick('Bohman', lobes=2)
    ini_ups['bohman2_linear'] = _magick('Bohman', lobes=2, lin=True)
    ini_ups['bohman3_srgb'] = _magick('Bohman', lobes=3)
    ini_ups['bohman3_linear'] = _magick('Bohman', lobes=3, lin=True)
    ini_ups['bohman4_srgb'] = _magick('Bohman')
    ini_ups['bohman4_linear'] = _magick('Bohman', lin=True)


def _std_int_lin_tensor_mtds_2(ini_ups):
    """Add the 2nd part of the standard interpolatory linear tensor methods.

    .. note::
```

```
        This is a private function called by :func:`_add_default_upsamplers`.

    :param ini_ups: upsamplers for the specified :file:`.ini` file
    :type ini_ups:  `dict`

    """
    ini_ups['cosine2_srgb'] = _magick('Cosine', lobes=2)
    ini_ups['cosine2_linear'] = _magick('Cosine', lobes=2, lin=True)
    ini_ups['cosine3_srgb'] = _magick('Cosine')
    ini_ups['cosine3_linear'] = _magick('Cosine', lin=True)
    ini_ups['cosine4_srgb'] = _magick('Cosine', lobes=4)
    ini_ups['cosine4_linear'] = _magick('Cosine', lobes=4, lin=True)
    ini_ups['hamming2_srgb'] = _magick('Hamming', lobes=2)
    ini_ups['hamming2_linear'] = _magick('Hamming', lobes=2, lin=True)
    ini_ups['hamming3_srgb'] = _magick('Hamming', lobes=3)
    ini_ups['hamming3_linear'] = _magick('Hamming', lobes=3, lin=True)
    ini_ups['hamming4_srgb'] = _magick('Hamming')
    ini_ups['hamming4_linear'] = _magick('Hamming', lin=True)
    ini_ups['parzen2_srgb'] = _magick('Parzen', lobes=2)
    ini_ups['parzen2_linear'] = _magick('Parzen', lobes=2, lin=True)
    ini_ups['parzen3_srgb'] = _magick('Parzen', lobes=3)
    ini_ups['parzen3_linear'] = _magick('Parzen', lobes=3, lin=True)
    ini_ups['parzen4_srgb'] = _magick('Parzen')
    ini_ups['parzen4_linear'] = _magick('Parzen', lin=True)
    ini_ups['welch2_srgb'] = _magick('Welsh', lobes=2)
    ini_ups['welch2_linear'] = _magick('Welsh', lobes=2, lin=True)
    ini_ups['welch3_srgb'] = _magick('Welsh')
    ini_ups['welch3_linear'] = _magick('Welsh', lin=True)
    ini_ups['welch4_srgb'] = _magick('Welsh', lobes=4)
    ini_ups['welch4_linear'] = _magick('Welsh', lobes=4, lin=True)
    ini_ups['hann2_srgb'] = _magick('Hanning', lobes=2)
    ini_ups['hann2_linear'] = _magick('Hanning', lobes=2, lin=True)
    ini_ups['hann3_srgb'] = _magick('Hanning', lobes=3)
    ini_ups['hann3_linear'] = _magick('Hanning', lobes=3, lin=True)
    ini_ups['hann4_srgb'] = _magick('Hanning')
    ini_ups['hann4_linear'] = _magick('Hanning', lin=True)


def _novel_int_lin_flt_mtds(ini_ups):
    """Add the novel interpolatory linear filtering methods.

    .. note::

        This is a private function called by :func:`_add_default_upsamplers`.

    :param ini_ups: upsamplers for the specified :file:`.ini` file
    :type ini_ups:  `dict`

    """
    ini_ups['kaiser2_srgb'] = _magick('Kaiser', lobes=2, beta='5.36')
    ini_ups['kaiser2_linear'] = _magick('Kaiser', lobes=2, beta='5.36',
                                        lin=True)
    ini_ups['kaiser3_srgb'] = _magick('Kaiser', lobes=3, beta='8.93')
    ini_ups['kaiser3_linear'] = _magick('Kaiser', lobes=3, beta='8.93',
                                        lin=True)
    ini_ups['kaiser4_srgb'] = _magick('Kaiser', beta='12.15')
    ini_ups['kaiser4_linear'] = _magick('Kaiser', beta='12.15', lin=True)
    ini_ups['kaisersharp2_srgb'] = _magick('Kaiser', lobes=2,
                                           beta='4.7123889803846899')
    ini_ups['kaisersharp2_linear'] = _magick('Kaiser', lobes=2, lin=True,
                                             beta='4.7123889803846899')
    ini_ups['kaisersharp3_srgb'] = _magick('Kaiser', lobes=3,
                                           beta='7.853981633974483')
    ini_ups['kaisersharp3_linear'] = _magick('Kaiser', lobes=3, lin=True,
                                             beta='7.853981633974483')
    ini_ups['kaisersharp4_srgb'] = _magick('Kaiser', beta='10.995574287564276')
    ini_ups['kaisersharp4_linear'] = _magick('Kaiser', lin=True,
                                             beta='10.995574287564276')
```

```
    ini_ups['kaisersoft2_srgb'] = _magick('Kaiser', lobes=2,
                                          beta='6.2831853071795865')
    ini_ups['kaisersoft2_linear'] = _magick('Kaiser', lobes=2, lin=True,
                                            beta='6.2831853071795865')
    ini_ups['kaisersoft3_srgb'] = _magick('Kaiser', lobes=3,
                                          beta='9.4247779607693797')
    ini_ups['kaisersoft3_linear'] = _magick('Kaiser', lobes=3, lin=True,
                                            beta='9.4247779607693797')
    ini_ups['kaisersoft4_srgb'] = _magick('Kaiser', beta='12.566370614359173')
    ini_ups['kaisersoft4_linear'] = _magick('Kaiser', lin=True,
                                            beta='12.566370614359173')


def _std_nonint_lin_tensor_mtds(ini_ups):
    """Add the standard non-interpolatory linear tensor methods.

    .. note::

        This is a private function called by :func:`_add_default_upsamplers`.

    :param ini_ups: upsamplers for the specified :file:`.ini` file
    :type ini_ups:  `dict`

    """
    ini_ups['quadratic_b_spline_srgb'] = _magick('Quadratic')
    ini_ups['quadratic_b_spline_linear'] = _magick('Quadratic', lin=True)
    ini_ups['cubic_b_spline_srgb'] = _magick('Cubic')
    ini_ups['cubic_b_spline_linear'] = _magick('Cubic', lin=True)
    ini_ups['mitchell_netravali_srgb'] = _magick(None)
    ini_ups['mitchell_netravali_linear'] = _magick(None, lin=True)


def _std_int_ewa_lin_flt_mtds(ini_ups):
    """Add the standard interpolatory EWA linear filtering methods.

    .. note::

        This is a private function called by :func:`_add_default_upsamplers`.

    :param ini_ups: upsamplers for the specified :file:`.ini` file
    :type ini_ups:  `dict`

    """
    ini_ups['ewa_teepee_srgb'] = _magick('Triangle', dist=True)
    ini_ups['ewa_teepee_linear'] = _magick('Triangle', dist=True, lin=True)
    ini_ups['ewa_hermite_srgb'] = _magick('Hermite', dist=True)
    ini_ups['ewa_hermite_linear'] = _magick('Hermite', dist=True, lin=True)


def _std_nonint_ewa_lin_flt_mtds(ini_ups):
    """Add the standard non-interpolatory EWA linear filtering methods.

    .. note::

        This is a private function called by :func:`_add_default_upsamplers`.

    :param ini_ups: upsamplers for the specified :file:`.ini` file
    :type ini_ups:  `dict`

    """
    ini_ups['ewa_quadratic_b_spline_srgb'] = _magick('Quadratic', dist=True)
    ini_ups['ewa_quadratic_b_spline_linear'] = _magick('Quadratic', dist=True,
                                                       lin=True)
    ini_ups['ewa_cubic_b_spline_srgb'] = _magick('Spline', dist=True)
    ini_ups['ewa_cubic_b_spline_linear'] = _magick('Spline', dist=True,
                                                   lin=True)
    ini_ups['ewa_lanczos2_srgb'] = _magick('Lanczos2', dist=True)
    ini_ups['ewa_lanczos2_linear'] = _magick('Lanczos2', dist=True, lin=True)
    ini_ups['ewa_lanczos3_srgb'] = _magick('Lanczos', dist=True)
```

```
    ini_ups['ewa_lanczos3_linear'] = _magick('Lanczos', dist=True, lin=True)
    ini_ups['ewa_lanczos4_srgb'] = _magick('Lanczos', lobes=4, dist=True)
    ini_ups['ewa_lanczos4_linear'] = _magick('Lanczos', lobes=4, dist=True,
                                             lin=True)


def _novel_nonint_ewa_lin_flt_mtds(ini_ups):
    """Add the novel non-interpolatory EWA linear filtering methods.

    .. note::

        This is a private function called by :func:`_add_default_upsamplers`.

    :param ini_ups: upsamplers for the specified :file:`.ini` file
    :type ini_ups:  `dict`

    """
    ini_ups['ewa_robidoux_srgb'] = _magick(None, dist=True)
    ini_ups['ewa_robidoux_linear'] = _magick(None, dist=True, lin=True)
    ini_ups['ewa_mitchell_netravali_srgb'] = _magick('Mitchell', dist=True)
    ini_ups['ewa_mitchell_netravali_linear'] = _magick('Mitchell', dist=True,
                                                       lin=True)
    ini_ups['ewa_robidouxsharp_srgb'] = _magick('RobidouxSharp', dist=True)
    ini_ups['ewa_robidouxsharp_linear'] = _magick('RobidouxSharp', dist=True,
                                                  lin=True)
    ini_ups['ewa_catmull_rom_srgb'] = _magick('Catrom', dist=True)
    ini_ups['ewa_catmull_rom_linear'] = _magick('Catrom', dist=True, lin=True)
    ini_ups['ewa_lanczosradius2_srgb'] = _magick('LanczosRadius', dist=True,
                                                 lobes=2)
    ini_ups['ewa_lanczosradius2_linear'] = _magick('LanczosRadius', dist=True,
                                                   lobes=2,
                                                   lin=True)
    ini_ups['ewa_lanczosradius3_srgb'] = _magick('LanczosRadius', dist=True)
    ini_ups['ewa_lanczosradius3_linear'] = _magick('LanczosRadius', dist=True,
                                                   lin=True)
    ini_ups['ewa_lanczosradius4_srgb'] = _magick('LanczosRadius', lobes=4,
                                                 dist=True)
    ini_ups['ewa_lanczosradius4_linear'] = _magick('LanczosRadius', lobes=4,
                                                   dist=True, lin=True)
    ini_ups['ewa_lanczos2sharp_srgb'] = _magick('Lanczos2', dist=True,
                                                blur='.9580278036312191')
    ini_ups['ewa_lanczos2sharp_linear'] = _magick('Lanczos2', dist=True,
                                                  blur='.9580278036312191',
                                                  lin=True)
    ini_ups['ewa_lanczos3sharp_srgb'] = _magick('Lanczos', dist=True,
                                                blur='.9891028367558475')
    ini_ups['ewa_lanczos3sharp_linear'] = _magick('Lanczos', dist=True,
                                                  blur='.9891028367558475',
                                                  lin=True)
    ini_ups['ewa_lanczos4sharp_srgb'] = _magick('Lanczos', lobes=4, dist=True,
                                                blur='.9870395083298263')
    ini_ups['ewa_lanczos4sharp_linear'] = _magick('Lanczos', lobes=4,
                                                  dist=True, lin=True,
                                                  blur='.9870395083298263')
    ini_ups['ewa_lanczos2sharpest_srgb'] = _magick('Lanczos2', dist=True,
                                                   blur='.88826421508540347')
    ini_ups['ewa_lanczos2sharpest_linear'] = _magick('Lanczos2', dist=True,
                                                     blur='.88826421508540347',
                                                     lin=True)
    ini_ups['ewa_lanczos3sharpest_srgb'] = _magick('Lanczos', dist=True,
                                                   blur='.88549061701764')
    ini_ups['ewa_lanczos3sharpest_linear'] = _magick('Lanczos', dist=True,
                                                     blur='.88549061701764',
                                                     lin=True)
    ini_ups['ewa_lanczos4sharpest_srgb'] = _magick('Lanczos', lobes=4,
                                                   dist=True,
                                                   blur='.88451002338585141')
    ini_ups['ewa_lanczos4sharpest_linear'] = _magick('Lanczos', lobes=4,
                                                     dist=True, lin=True,
```

```
                                                    blur='.884510023385851 41')


def _add_default_upsamplers(ini):
    """Add the default upsamplers to the specified :file:`.ini` file.

    .. note::

        This is a private function called by :func:`main`.

    :param ini: the :file:`.ini` file to modify
    :type ini:  :class:`configobj.ConfigObj`

    """
    # Define the list of upsamplers to use.
    ups = 'Upsamplers'
    ini[ups] = {}
    ini.comments[ups] = [
        '', 'UPSAMPLING COMMANDS',
        'To add an upsampler, provide the command to execute it.',
        'The command can make use of the following replacement fields:',
        '{0} = input image',
        '{1} = output image',
        '{2} = upsampling ratio',
        '{3} = upsampled size (always 840)'
    ]

    _std_int_lin_tensor_mtds_1(ini[ups])
    _std_int_lin_tensor_mtds_2(ini[ups])
    _novel_int_lin_flt_mtds(ini[ups])
    _std_nonint_lin_tensor_mtds(ini[ups])
    _std_int_ewa_lin_flt_mtds(ini[ups])
    _std_nonint_ewa_lin_flt_mtds(ini[ups])
    _novel_nonint_ewa_lin_flt_mtds(ini[ups])


def _add_default_metrics(ini):
    """Add the default metrics to the specified :file:`.ini` file.

    .. note::

        This is a private function called by :func:`main`.

    :param ini: the :file:`.ini` file to modify
    :type ini:  :class:`configobj.ConfigObj`

    """
    # Define the list of error metrics to use.
    metrics = 'Metrics'
    ini[metrics] = {}
    ini.comments[metrics] = [
        '', 'IMAGE DIFFERENCE METRICS AND AGGREGATORS',
        'Each metric must be associated with a data aggregation method.',
        'To add a metric, you must provide the following three items:',
        '',
        '1. Error metric command, using the following replacement fields:',
        '{0} = reference image',
        '{1} = test image',
        '',
        '2. Aggregator command, using the following replacement field:',
        '{0} = list of error data to aggregate',
        '',
        '3. Best-to-worst ordering, given as a 0 or 1:',
        '0 = ascending',
        '1 = descending'
    ]
    ini[metrics]['srgb_1'] = _metric('srgb_1', '1_1', 0)
    ini[metrics]['srgb_2'] = _metric('srgb_2', '1_2', 0)
    ini[metrics]['srgb_4'] = _metric('srgb_4', '1_4', 0)
```

```
    ini[metrics]['srgb_inf'] = _metric('srgb_inf', 'l_inf', 0)
    ini[metrics]['cmc_1'] = _metric('cmc_1', 'l_1', 0)
    ini[metrics]['cmc_2'] = _metric('cmc_2', 'l_2', 0)
    ini[metrics]['cmc_4'] = _metric('cmc_4', 'l_4', 0)
    ini[metrics]['cmc_inf'] = _metric('cmc_inf', 'l_inf', 0)
    ini[metrics]['xyz_1'] = _metric('xyz_1', 'l_1', 0)
    ini[metrics]['xyz_2'] = _metric('xyz_2', 'l_2', 0)
    ini[metrics]['xyz_4'] = _metric('xyz_4', 'l_4', 0)
    ini[metrics]['xyz_inf'] = _metric('xyz_inf', 'l_inf', 0)
    ini[metrics]['blur_1'] = _metric('blur_1', 'l_1', 0)
    ini[metrics]['blur_2'] = _metric('blur_2', 'l_2', 0)
    ini[metrics]['blur_4'] = _metric('blur_4', 'l_4', 0)
    ini[metrics]['blur_inf'] = _metric('blur_inf', 'l_inf', 0)
    ini[metrics]['mssim'] = _metric('mssim', 'l_1', 1)


def main():
    """Run :ref:`exquires-new`.

    Create a project file to use with :ref:`exquires-run` and
    :ref:`exquires-update`.

    """

    # Construct the path to the default test image.
    this_dir = path.abspath(path.dirname(__file__))
    wave = path.join(this_dir, 'wave.tif')

    # Define the command-line argument parser.
    parser = parsing.ExquiresParser(description=__doc__)
    parser.add_argument('-p', '--proj', metavar='PROJECT', type=str,
                        help='name of the project (default: project1)',
                        default='project1')
    parser.add_argument('-I', '--image', metavar='IMAGE', type=str, nargs='+',
                        help='the test images to use (default: wave.tif)',
                        default=[wave])

    # Attempt to parse the command-line arguments.
    args = parser.parse_args()

    # Create a new project file.
    ini = ConfigObj()
    ini.filename = '.'.join([args.proj, 'ini'])

    _add_default_images(ini, args.image)
    _add_default_ratios(ini)
    _add_default_downsamplers(ini)
    _add_default_upsamplers(ini)
    _add_default_metrics(ini)

    # Write the project file.
    ini.write()

if __name__ == '__main__':
    main()
```

A.7 operations.py

```
#!/usr/bin/env python
# coding: utf-8
#
#  Copyright (c) 2012, Adam Turcotte (adam.turcotte@gmail.com)
#                      Nicolas Robidoux (nicolas.robidoux@gmail.com)
#  License: BSD 2-Clause License
#
#  This file is part of the
```

```
#  EXQUIRES (EXtensible QUantitative Image RESampling) test suite
#

"""A collection of classes used to compute image difference data.

The hierarchy of classes is as follows:

    * :class:`Operations` encapsulate a list of :class:`Images`
    * :class:`Images` encapsulate a `dict` of images
      and a list of :class:`Downsamplers`
    * :class:`Downsamplers` encapsulate a `dict` of downsamplers
      and a list of :class:`Ratios`
    * :class:`Ratios` encapsulate a `dict` of ratios and a list :class:`Images`
    * :class:`Images` encapsulate a `dict` of images and a `dict` of metrics

These classes work together to downsample the master images, upsample the
downsampled images, and compare the upsampled images to the master images.
To perform the operations, call :meth:`Operations.compute`.

"""

import os
import shutil
from subprocess import call, check_output

from exquires import database, progress, tools

# pylint: disable-msg=R0903


class Operations(object):

    """A collection of Image objects to compute data with.

    This class is responsible for calling all operations defined in the
    specified project file when using :ref:`exquires-run` or
    :ref:`exquires-update`.

    :param images: images to downsample
    :type images:  list of :class:`Images`

    """

    def __init__(self, images):
        """Create a new :class:`Operations` object."""
        self.images = images
        self.len = sum(len(image) for image in self.images)

    def __len__(self):
        """Return the length of this :class:`Operations` object.

        The length of an :class:`Operations` object is the total number of
        operations (downsampling, upsampling, and comparing) to be performed.

        :return: length of this :class:`Operations` object
        :rtype:  `integer`

        """
        return self.len

    def compute(self, args, old=None):
        """Perform all operations.

        :param args:             arguments
        :param args.prog:        name of the calling program
        :param args.dbase_file:  database file
        :param args.proj:        name of the current project
        :param args.silent:      `True` if using silent mode
        :param args.met_same:    unchanged metrics
```

```
:param args.metrics:     current metrics
:param args.config_file: current configuration file
:param args.config_bak:  previous configuration file
:param old:              old configuration entries to be removed
:type args:              :class:`argparse.Namespace`
:type args.prog:         `string`
:type args.dbase_file:   `path`
:type args.proj:         `string`
:type args.silent:       `boolean`
:type args.met_same:     `dict`
:type args.metrics:      `dict`
:type args.config_file:  `path`
:type args.config_bak:   `path`
:type old:               :class:`argparse.Namespace`

"""
# Setup verbose mode.
if not args.silent:
    prg = progress.Progress(args.prog, args.proj, len(self))
    args.do_op = prg.do_op
    cleanup = prg.cleanup
    complete = prg.complete
else:
    prg = []
    args.do_op = lambda *a, **k: None
    cleanup = lambda: None
    complete = lambda: None

# Backup any existing database file.
dbase_bak = '.'.join([args.dbase_file, 'bak'])
if os.path.isfile(args.dbase_file):
    shutil.copyfile(args.dbase_file, dbase_bak)

# Open the database connection.
args.dbase = database.Database(args.dbase_file)

success = True
try:
    # Remove old database tables.
    if old:
        cleanup()
        args.dbase.drop_tables(old.images,
                               old.downsamplers, old.ratios)

    # Create the project folder if it does not exist.
    tools.create_dir(args.proj)

    # Compute for all images.
    for image in self.images:
        image.compute(args)
except StandardError as std_err:
    success = False
    error = std_err
finally:
    # Remove the project directory and close the database.
    shutil.rmtree(args.proj, True)
    args.dbase.close()

    if success:
        # Backup the project file.
        shutil.copyfile(args.config_file, args.config_bak)

        # Delete the database backup.
        if os.path.isfile(dbase_bak):
            os.remove(dbase_bak)

        # Indicate completion and restore the console.
        complete()
        del prg
```

```
            else:
                # Restore the previous database file.
                os.remove(args.dbase_file)
                if os.path.isfile(dbase_bak):
                    shutil.move(dbase_bak, args.dbase_file)

                # Restore the console
                del prg

                # Print an error message.
                print error


class Images(object):

    """This class calls operations for a particular set of images.

    :param images:       images to downsample
    :param downsamplers: downsamplers to use
    :param same:         `True` if using unchanged images
    :type images:        `dict`
    :type downsamplers:  list of :class:`Downsamplers`
    :type same:          `boolean`

    """

    def __init__(self, images, downsamplers, same=False):
        """Create a new :class:`Images` object."""
        self.images = images
        self.downsamplers = downsamplers
        self.same = same
        self.len = (len(self.images) *
                    sum(len(down) + down.ops for down in self.downsamplers))

    def __len__(self):
        """Return the length of this :class:`Images` object.

        The length of an :class:`Images` object is the number of images times
        the sum of the lengths and the number of upsampling and comparison
        operations of each :class:`Downsamplers` object.

        :return: length of this :class:`Images` object
        :rtype:  `integer`

        """
        return self.len

    def compute(self, args):
        """Perform all operations for this set of images.

        :param args:             arguments
        :param args.dbase_file:  database file
        :param args.dbase:       connected database
        :param args.proj:        name of the current project
        :param args.silent:      `True` if using silent mode
        :param args.met_same:    unchanged metrics
        :param args.metrics:     current metrics
        :param args.do_op:       updates the displayed progress
        :type args:              :class:`argparse.Namespace`
        :type args.dbase_file:   `path`
        :type args.dbase:        :class:`database.Database`
        :type args.proj:         `string`
        :type args.silent:       `boolean`
        :type args.met_same:     `dict`
        :type args.metrics:      `dict`
        :type args.do_op:        `function`

        """
        # Compute for all images.
```

```
        for args.image in self.images:
            # Make a copy of the test image.
            if len(self):
                args.image_dir = tools.create_dir(args.proj, args.image)
                args.master = os.path.join(args.image_dir, 'master.tif')
                shutil.copyfile(self.images[args.image], args.master)

            # Compute for all downsamplers.
            for downsampler in self.downsamplers:
                downsampler.compute(args, self.same)

            # Remove the directory for this image.
            if len(self):
                shutil.rmtree(args.image_dir, True)


class Downsamplers(object):

    """This class calls operations for a particular set of downsamplers.

    :param downsamplers: downsamplers to use
    :param ratios:       ratios to downsample by
    :param same:         `True` if using unchanged downsamplers
    :type downsamplers:  `dict`
    :type ratios:        list of :class:`Ratios`
    :type same:          `boolean`

    """

    def __init__(self, downsamplers, ratios, same=False):
        """Create a new :class:`Downsamplers` object."""
        self.downsamplers = downsamplers
        self.ratios = ratios
        self.same = same
        self.ops = len(self.downsamplers) * sum(rat.ops for rat in self.ratios)
        self.len = (len(self.downsamplers) *
                    sum(len(rat) for rat in self.ratios)) if self.ops else 0

    def __len__(self):
        """Return the length of this :class:`Downsamplers` object.

        The length of a :class:`Downsamplers` object is the number of
        downsamplers times the sum of the lengths of each :class:`Ratios`
        object.

        :return: length of this :class:`Downsamplers` object
        :rtype:  `integer`

        """
        return self.len

    def compute(self, args, same):
        """Perform all operations for this set of downsamplers.

        :param args:             arguments
        :param args.dbase_file: database file
        :param args.dbase:       connected database
        :param args.proj:        name of the current project
        :param args.silent:      `True` if using silent mode
        :param args.met_same:    unchanged metrics
        :param args.metrics:     current metrics
        :param args.do_op:       updates the displayed progress
        :param args.image:       name of the image
        :param args.image_dir:   directory to store results for this image
        :param args.master:      master image to downsample
        :param same:             `True` if possibly accessing an existing table
        :type args:              :class:`argparse.Namespace`
        :type args.dbase_file:   `path`
        :type args.dbase:        :class:`database.Database`
```

```
        :type args.proj:         'string'
        :type args.silent:       'boolean'
        :type args.met_same:     'dict'
        :type args.metrics:      'dict'
        :type args.do_op:        'function'
        :type args.image:        'string'
        :type args.image_dir:    'path'
        :type args.master:       'path'
        :type same:              'boolean'

        """
        is_same = self.same and same

        # Compute for all downsamplers.
        for args.downsampler in self.downsamplers:
            # Create a directory for this downsampler if necessary.
            if len(self):
                args.downsampler_dir = tools.create_dir(args.image_dir,
                                                        args.downsampler)

            # Compute for all ratios.
            for ratio in self.ratios:
                ratio.compute(args, self.downsamplers, is_same)

            # Remove the directory for this downsampler.
            if len(self):
                shutil.rmtree(args.downsampler_dir, True)


class Ratios(object):

    """This class calls operations for a particular set of ratios.

    :param ratios:     ratios to downsample by
    :param upsamplers: upsamplers to use
    :param same:       'True' if using unchanged ratios
    :type ratios:      'dict'
    :type upsamplers:  list of :class:'Upsamplers'
    :type same:        'boolean'

    """

    def __init__(self, ratios, upsamplers, same=False):
        """Create a new :class:'Ratios' object."""
        self.ratios = ratios
        self.upsamplers = upsamplers
        self.same = same
        self.ops = len(self.ratios) * sum(len(ups) for ups in self.upsamplers)
        self.len = len(self.ratios) if self.ops else 0

    def __len__(self):
        """Return the length of this :class:'Ratios' object.

        The length of a :class:'Ratios' object is the number of ratios.

        :return: length of this :class:'Ratios' object
        :rtype:  'integer'

        """
        return self.len

    def compute(self, args, downsamplers, same):
        """Perform all operations for this set of ratios.

        :param args:                arguments
        :param args.dbase_file:     database file
        :param args.dbase:          connected database
        :param args.proj:           name of the current project
        :param args.silent:         'True' if using silent mode
```

```
:param args.met_same:         unchanged metrics
:param args.metrics:          current metrics
:param args.do_op:            updates the displayed progress
:param args.image:            name of the image
:param args.image_dir:        directory to store results for this image
:param args.master:           master image to downsample
:param args.downsampler:      name of the downsampler
:param args.downsampler_dir: directory to store dowsampled images
:param downsamplers:          downsamplers to use
:param same:                  'True' if accessing an existing table
:type args:                   :class:'argparse.Namespace'
:type args.dbase_file:        'path'
:type args.dbase:             :class:'database.Database'
:type args.proj:              'string'
:type args.silent:            'boolean'
:type args.met_same:          'dict'
:type args.metrics:           'dict'
:type args.do_op:             'function'
:type args.image:             'string'
:type args.image_dir:         'path'
:type args.master:            'path'
:type args.downsampler:       'string'
:type args.downsampler_dir:   'path'
:type downsamplers:           'dict'
:type same:                   'boolean'

"""
is_same = self.same and same

# Compute for all ratios.
for args.ratio in self.ratios:
    if len(self):
        args.small = os.path.join(args.downsampler_dir,
                                  '.'.join([args.ratio, 'tif']))

        # Create a directory for this ratio.
        ratio_dir = tools.create_dir(args.downsampler_dir, args.ratio)

        # Downsample master.tif by ratio using downsampler.
        #  {0} input image path (master)
        #  {1} output image path (small)
        #  {2} downsampling ratio
        #  {3} downsampled size (width or height)
        args.do_op(args)
        call(
            downsamplers[args.downsampler].format(
                args.master, args.small,
                args.ratio, self.ratios[args.ratio]
            ).split()
        )

    if is_same:
        # Access the existing database table.
        args.table = '_'.join([args.image,
                               args.downsampler, args.ratio])
        args.table_bak = args.dbase.backup_table(args.table,
                                                 args.metrics)
    else:
        # Create a new database table.
        args.table = args.dbase.add_table(
            args.image, args.downsampler, args.ratio, args.metrics)

    # Compute for all upsamplers.
    for upsampler in self.upsamplers:
        upsampler.compute(args, is_same)

    # Remove the directory for this ratio.
    if len(self):
        shutil.rmtree(ratio_dir, True)
```

```
            # Delete the backup table.
            if is_same:
                args.dbase.drop_backup(args.table_bak)


class Upsamplers(object):

    """This class upsamples an image and compares with its master image.

    :param upsamplers: upsamplers to use
    :param metrics:    metrics to compare with
    :param same:       `True` if using unchanged upsamplers
    :type upsamplers:  `dict`
    :type metrics:     `dict`
    :type same:        `boolean`

    """

    def __init__(self, upsamplers, metrics, same=False):
        """Create a new :class:`Upsamplers` object."""
        self.upsamplers = upsamplers
        self.metrics = metrics
        self.same = same
        ops = len(self.metrics)
        self.len = len(self.upsamplers) * (ops + 1) if ops else 0

    def __len__(self):
        """Return the length of this :class:`Upsamplers` object.

        The length of an :class:`Upsamplers` object is the number of upsampling
        and comparison operations to perform.

        :return: length of this :class:`Upsamplers` object
        :rtype:  `integer`

        """
        return self.len

    def compute(self, args, same):
        """Perform all operations for this set of ratios.

        :param args:                arguments
        :param args.dbase_file:     database file
        :param args.dbase:          connected database
        :param args.proj:           name of the current project
        :param args.silent:         `True` if using silent mode
        :param args.met_same:       unchanged metrics
        :param args.metrics:        current metrics
        :param args.do_op:          updates the displayed progress
        :param args.image:          name of the image
        :param args.image_dir:      directory to store results for this image
        :param args.master:         master image to downsample
        :param args.downsampler:    name of the downsampler
        :param args.downsampler_dir: directory to store dowsampled images
        :param args.ratio:          resampling ratio
        :param args.small:          downsampled image
        :param args.table:          name of the table to insert the row into
        :param args.table_bak:      name of the backup table (if it exists)
        :param same:                `True` if accessing an existing table
        :type args:                 :class:`argparse.Namespace`
        :type args.dbase_file:      `path`
        :type args.dbase:           :class:`database.Database`
        :type args.proj:            `string`
        :type args.silent:          `boolean`
        :type args.met_same:        `dict`
        :type args.metrics:         `dict`
        :type args.do_op:           `function`
        :type args.image:           `string`
```

```
        :type args.image_dir:         'path'
        :type args.master:            'path'
        :type args.downsampler:       'string'
        :type args.downsampler_dir:   'path'
        :type args.ratio:             'string'
        :type args.small:             'path'
        :type args.table:             'string'
        :type args.table_bak:         'string'
        :type same:                   'boolean'

        """
        is_same = self.same and same and args.met_same

        # Compute for all upsamplers.
        for upsampler in self.upsamplers:
            row = {}
            if is_same:
                # Access the existing table row.
                row = args.dbase.get_error_data(args.table_bak, upsampler,
                                                ','.join(args.met_same))

            if len(self):
                if not is_same:
                    # Start creating a new table row.
                    row['upsampler'] = upsampler

                # Construct the path to the upsampled image.
                large = os.path.join(
                    os.path.dirname(args.small), args.ratio,
                    '.'.join([upsampler, 'tif'])
                )

                # Upsample ratio.tif back to 840 using upsampler.
                #  {0} input image path (small)
                #  {1} output image path (large)
                #  {2} upsampling ratio
                #  {3} upsampled size (always 840)
                args.do_op(args, upsampler)
                call(self.upsamplers[upsampler].format(
                        args.small, large, args.ratio, 840).split())

                # Compute for all metrics.
                for metric in self.metrics:
                    # Compare master.tif to upsampler.tif.
                    #  {0} reference image path (master)
                    #  {1} test image path (large)
                    args.do_op(args, upsampler, metric)
                    row[metric] = float(
                        check_output(
                            self.metrics[metric][0].format(args.master,
                                                           large).split()
                        )
                    )

                # Remove the upsampled image.
                os.remove(large)

            # Add the new row to the table.
            if row:
                args.dbase.insert(args.table, row)
```

A.8 parsing.py

```
#!/usr/bin/env python
# coding: utf-8
#
```

```
#  Copyright (c) 2012, Adam Turcotte (adam.turcotte@gmail.com)
#                      Nicolas Robidoux (nicolas.robidoux@gmail.com)
#  License: BSD 2-Clause License
#
#  This file is part of the
#  EXQUIRES (EXtensible QUantitative Image RESampling) test suite
#

"""Classes and methods used for parsing arguments and formatting help text."""

import argparse
import os
import fnmatch
import re
import sys

from configobj import ConfigObj

from exquires import tools
from exquires import __version__ as VERSION

# pylint: disable-msg=R0903


def _remove_duplicates(input_list):
    """Remove duplicate entries from a list.

    .. note::

        This is a private function called by :meth:`ListAction.__call__`
        and :meth:`RatioAction.__call__`.

    :param input_list: list to remove duplicate entries from
    :type input_list:  `list of values`

    :return:           list with duplicate entries removed
    :rtype:            `list of values`

    """
    unique = set()
    return [x for x in input_list if x not in unique and not unique.add(x)]


def _format_doc(docstring):
    """Parse the module docstring and re-format all `reST` markup.

    .. note::

        This is a private function called when creating a new
        :class:`ExquiresParser` object.

    :param docstring: docstring to format
    :type docstring:  `string`

    :return:          formatted docstring
    :rtype:           `string`

    """
    # Deal with directives and LaTeX math symbols.
    dir1 = re.sub(r':file:`', r'\033[4m', docstring)
    dir2 = re.sub(r':\S+:`', r'\033[1m', dir1)
    dir3 = re.sub('`', r'\033[0m', dir2)
    dir4 = re.sub(r'\\infty', 'infinity', re.sub(r'\\ell', 'L', dir3))

    # Deal with list items.
    item1 = re.sub(r'    \* ', u'    \u2022 ', dir4)

    # Deal with bold formatting.
    bold1 = re.sub(r' \*{2}', r' \033[1m', item1)
```

```
    bold2 = re.sub(r'^\*{2}', r'^\033[1m', bold1)
    bold3 = re.sub(r'\*{2} ', r'\033[0m ', bold2)
    bold4 = re.sub(r'\*{2},', r'\033[0m,', bold3)
    bold5 = re.sub(r'\*{2}.', r'\033[0m.', bold4)
    return re.sub(r'\*{2}$', r'\033[0m$', bold5)


class ExquiresParser(argparse.ArgumentParser):

    """Generic **EXQUIRES** parser.

    :param description: docstring from the calling program
    :type description:  `string`

    """

    def __init__(self, description):
        """Create a new ExquiresParser object."""
        super(ExquiresParser, self).__init__(
            version=VERSION, description=_format_doc(description),
            formatter_class=lambda prog: ExquiresHelp(prog,
                                                      max_help_position=36)
        )

    def parse_args(self, args=None, namespace=None):
        """Parse command-line arguments.

        :param args:      the command-line arguments
        :param namespace: the namespace
        :type args:       `string`
        :type namespace:  :class:`argparse.Namespace`

        :return:          the parsed arguments
        :rtype:           :class:`argparse.Namespace`

        """
        # Get the raw command-line arguments
        if args is None:
            args = sys.argv[1:]

        # Attempt to parse the command-line arguments.
        try:
            args = super(ExquiresParser, self).parse_args(args, namespace)
        except argparse.ArgumentTypeError, error:
            self.error(str(error))

        # Return the parsed arguments.
        return args


class OperationsParser(ExquiresParser):

    """Parser used by :ref:`exquires-run` and :ref:`exquires-update`.

    :param description: docstring from the calling program
    :param update:      `True` if called by :ref:`exquires-update`
    :type description:  `string`
    :type update:       `boolean`

    """

    def __init__(self, description, update=False):
        """Create a new OperationsParser object."""
        super(OperationsParser, self).__init__(description)
        self.add_argument('-s', '--silent', action='store_true',
                          help='do not display progress information')
        self.add_argument('-p', '--proj', metavar='PROJECT',
                          type=str, default='project1',
                          help='name of the project (default: project1)')
```

```
        self.update = update

    def parse_args(self, args=None, namespace=None):
        """Parse the received arguments.

        This method parses the arguments received by  :ref:`exquires-run` or
        :ref`exquires-update`.

        :param args:      the command-line arguments
        :param namespace: the namespace
        :type args:       `string`
        :type namespace:  :class:`argparse.Namespace`

        :return:          the parsed arguments
        :rtype:           :class:`argparse.Namespace`

        """
        # Get the raw command-line arguments
        if args is None:
            args = sys.argv[1:]

        # Attempt to parse the command-line arguments.
        args = super(OperationsParser, self).parse_args(args, namespace)

        # Construct the path to the configuration and database files.
        args.dbase_file = '.'.join([args.proj, 'db'])
        args.config_file = '.'.join([args.proj, 'ini'])
        args.config_bak = '.'.join([args.config_file, 'bak'])
        args.prog = self.prog

        # Report an error if the configuration file does not exist.
        if not os.path.isfile(args.config_file):
            self.error(' '.join(['unrecognized project:', args.proj]))

        if self.update:
            # Determine if the database can be updated.
            if not (os.path.isfile(args.config_bak) and
                    os.path.isfile(args.dbase_file)):
                self.error(' '.join([args.proj, 'has not been run']))
        else:
            # Create a new database file, backing up any that already exists.
            if os.path.isfile(args.dbase_file):
                os.rename(args.dbase_file, '.'.join([args.proj, 'db', 'bak']))

        # Return the parsed arguments.
        return args


class StatsParser(ExquiresParser):

    """Parser used by :ref:`exquires-report` and :ref:`exquires-correlate`.

    :param description: docstring from the calling program
    :param correlate:   `True` if using :ref:`exquires-correlate`
    :type description:  `string`
    :type correlate:    `boolean`

    """

    def __init__(self, description, correlate=False):
        """Create a new StatsParser object."""
        super(StatsParser, self).__init__(description)
        self.correlate = correlate

        # Output options.
        self.add_argument('-l', '--latex', action='store_true',
                          help='print a LaTeX formatted table')

        if not correlate:
```

```
        group = self.add_mutually_exclusive_group()
        group.add_argument('-r', '--rank', action='store_true',
                           help='print Spearman (fractional) ranks')
        group.add_argument('-m', '--merge', action='store_true',
                           help='print merged Spearman ranks')

    self.add_argument('-p', '--proj', metavar='PROJECT', type=str,
                      action=ProjectAction,
                      help='name of the project (default: project1)')
    self.add_argument('-f', '--file', metavar='FILE',
                      type=argparse.FileType('w'), default=sys.stdout,
                      help='output to file (default: sys.stdout)')
    self.add_argument('-d', '--digits', metavar='DIGITS',
                      type=int, choices=range(1, 16), default=4,
                      help='total number of digits (default: 4)')

    if correlate:
        # Anchor option (sorting for exquires-correlate).
        self.add_argument('-a', '--anchor', metavar='ANCHOR', type=str,
                          action=AnchorAction, default=None,
                          help='sort using this anchor (default: none)')
    else:
        # Sort option.
        self.add_argument('-s', '--sort', metavar='METRIC', type=str,
                          action=SortAction, default=None,
                          help='sort using this metric (default: first)')

    # Upsampler selection.
    self.add_argument('-U', '--up', metavar='METHOD',
                      type=str, nargs='+', action=ListAction,
                      help='upsamplers to consider (default: all)')

    # Determine if using exquires-report or exquires-correlate.
    if correlate:
        group = self.add_mutually_exclusive_group()
    else:
        group = self

    # Aggregation/correlation options.
    group.add_argument('-I', '--image', metavar='IMAGE',
                       type=str, nargs='+', action=ListAction,
                       help='images to consider (default: all)')
    group.add_argument('-D', '--down', metavar='METHOD',
                       type=str, nargs='+', action=ListAction,
                       help='downsamplers to consider (default: all)')
    group.add_argument('-R', '--ratio', metavar='RATIO',
                       type=str, nargs='+', action=RatioAction,
                       help='ratios to consider (default: all)')
    group.add_argument('-M', '--metric', metavar='METRIC',
                       type=str, nargs='+', action=ListAction,
                       help='metrics to consider (default: all)')

def parse_args(self, args=sys.argv[1:], namespace=None):
    """Parse the received arguments.

    This method parses the arguments received by :ref:`exquires-report` or
    :ref:`exquires-correlate`.

    :param args:      the command-line arguments
    :param namespace: the namespace
    :type args:       `string`
    :type namespace:  :class:`argparse.Namespace`

    :return:          the parsed arguments
    :rtype:           :class:`argparse.Namespace`

    """
    # Deal with the -h/--help  and -v/--version options.
    help_or_version = False
```

```
        for arg in args:
            if arg in ('-h', '--help', '-v', '--version'):
                help_or_version = True
                break

        if not help_or_version:
            # Deal with the -p/--proj option.
            proj = False
            for i, arg in enumerate(args, 1):
                if arg in ('-p', '--proj'):
                    args.insert(0, args.pop(i))
                    args.insert(0, args.pop(i))
                    proj = True
                    break
            if not proj:
                args.insert(0, 'project1')
                args.insert(0, '--proj')

        # Setup the arguments to be parsed last.
        if self.correlate:
            # Make -a/--anchor the rightmost option.
            flags = ('-a', '--anchor')
        else:
            # Make -s/--sort the rightmost option.
            flags = ('-s', '--sort')
        for i, arg in enumerate(args):
            if arg in flags:
                args.append(args.pop(i))
                args.append(args.pop(i))
                break

        # Attempt to parse the command-line arguments.
        args = super(StatsParser, self).parse_args(args, namespace)

        # Default to sorting by the leftmost column.
        if not args.sort:
            args.sort = args.metric[0]

        # Deal with the sort/metric options.
        if not (self.correlate or args.merge) and args.sort not in args.metric:
            args.metric.insert(0, args.sort)
            args.show_sort = False

        # Construct the path to the database file.
        args.dbase_file = '.'.join([args.proj, 'db'])

        # Need to prune the dict first so it matches the argument list.
        args.metrics_d = tools.prune_metrics(args.metric, args.metrics_d)

        # Return the parsed arguments.
        return args


class ExquiresHelp(argparse.RawDescriptionHelpFormatter):

    """Formatter for generating usage messages and argument help strings.

    This class is designed to display options in a cleaner format than the
    standard argparse help strings.

    """

    def _format_action_invocation(self, action):
        """Format the string describing the invocation of the specified action.

        :param action: the parsing action
        :type action:  :class:`argparse.Action`

        :return:       the formatted action invocation
```

```
        :rtype:         `string`

        """
        if not action.option_strings:
            metavar, = self._metavar_formatter(action, action.dest)(1)
            return metavar
        else:
            parts = []
            if action.nargs == 0:
                # If the Optional doesn't take a value, the format is:
                #    -s, --long
                parts.extend(action.option_strings)
            else:
                # If the Optional takes a value, the format is:
                #    -s, --long ARGS
                default = action.dest.upper()
                args_string = self._format_args(action, default)
                option_strings = action.option_strings[:]
                last_option_string = option_strings.pop()
                for option_string in option_strings:
                    parts.append(option_string)
                parts.append('%s %s' % (last_option_string, args_string))

            return ', '.join(parts)

    def _fill_text(self, text, width, indent):
        """Fill action text with whitespace.

        :param text:   the text to display
        :param width:  line width
        :param indent: indentation printed before the text
        :type text:    `string`
        :type width:   `integer`
        :type indent:  `string`

        :return:       the formatted text
        :rtype:        `string`

        """
        return ''.join([indent + line for line in text.splitlines(True)])


class ProjectAction(argparse.Action):

    """Parser action to read a project file based on the specified name."""

    def __call__(self, parser, args, value, option_string=None):
        """Parse the :option:`-p`/:option:`--project` option.

        :param parser:        the parser calling this action
        :param args:          arguments
        :param values:        values
        :param option_string: command-line option string
        :type parser:         :class:`ExquiresParser`
        :type args:           :class:`argparse.Namespace`
        :type values:         `list of values`
        :type option_string:  `string`

        :raises:              :class:`argparse.ArgumentError`

        """
        # Construct the path to the configuation and database files.
        proj_file = '.'.join([value, 'ini', 'bak'])
        db_file = '.'.join([value, 'db'])
        setattr(args, 'db_file', db_file)

        # Exit with an error if one of these files is missing.
        if not (os.path.isfile(proj_file) and os.path.isfile(db_file)):
            msg = ' '.join(['do \'exquires-run -p', value, '\' first'])
```

```
            raise argparse.ArgumentTypeError(msg)

        # Read the configuration file last used to update the database.
        config = ConfigObj(proj_file)
        setattr(args, self.dest, value)
        args.image = config['Images'].keys()
        args.down = config['Downsamplers'].keys()
        args.ratio = config['Ratios'].keys()
        args.up = config['Upsamplers'].keys()
        args.metrics_d = config['Metrics']
        args.metrics = config['Metrics'].keys()
        args.metric = config['Metrics'].keys()
        args.sort = None
        args.show_sort = True
        args.merge = False
        args.rank = False

        # Set the default correlation key in case an anchor is specified.
        args.key = 'metric'


class ListAction(argparse.Action):

    """Parser action to handle wildcards for options that support them.

    When specifying aggregation options with exquires-report, this class
    expands any wildcards passed in arguments for the following options:

    * Images
    * Downsamplers
    * Upsamplers
    * Metrics

    """

    def __call__(self, parser, args, values, option_string=None):
        """Parse any option that supports lists with wildcard characters.

        :param parser:        the parser calling this action
        :param args:          arguments
        :param values:        values
        :param option_string: command-line option string
        :type parser:         :class:`ExquiresParser`
        :type args:           :class:`argparse.Namespace`
        :type values:         `list of values`
        :type option_string:  `string`

        :raises:              :class:`argparse.ArgumentError`

        """
        value_list = getattr(args, self.dest)
        matches = []
        for value in values:
            results = fnmatch.filter(value_list, value)
            if not results:
                tup = value, ', '.join([repr(val) for val in value_list])
                msg = 'invalid choice: %r (choose from %s)' % tup
                raise argparse.ArgumentError(self, msg)
            matches.extend(results)

        # Set the argument and possibly set the correlation key.
        setattr(args, self.dest, _remove_duplicates(matches))
        if self.dest is not 'up':
            args.key = self.dest


class RatioAction(argparse.Action):

    """Parser action to deal with ratio ranges."""
```

```
    def __call__(self, parser, args, values, option_string=None):
        """Parse the :option:`-r`/:option:`--ratio` option.

        :param parser:        the parser calling this action
        :param args:          arguments
        :param values:        values
        :param option_string: command-line option string
        :type parser:         :class:`ExquiresParser`
        :type args:           :class:`argparse.Namespace`
        :type values:         `list of values`
        :type option_string:  `string`

        :raises:              :class:`argparse.ArgumentError`

        """
        matches = []
        for value in values:
            # Detect range.
            nums = value.split('-')
            if len(nums) == 1:
                if nums[0] not in args.ratio:
                    tup = value, ', '.join([repr(val) for val in args.ratio])
                    msg = 'invalid choice: %r (choose from %s)' % tup
                    raise argparse.ArgumentError(self, msg)
                matches.append(int(nums[0]))
            elif len(nums) == 2:
                value_range = range(int(nums[0]), int(nums[1]) + 1)
                for num in value_range:
                    if str(num) not in args.ratio:
                        tup = value, ', '.join([
                            repr(val) for val in args.ratio
                        ])
                        msg = 'invalid choice: %r (choose from %s)' % tup
                        raise argparse.ArgumentError(self, msg)
                matches.extend(value_range)
            else:
                msg = 'format error in {}'.format(value)
                raise argparse.ArgumentError(self, msg)

        # Set the argument and correlation key.
        setattr(args, self.dest, _remove_duplicates(matches))
        args.key = self.dest


class AnchorAction(argparse.Action):

    """Parser action to sort the correlation matrix."""

    def __call__(self, parser, args, value, option_string=None):
        """Parse the :option:`-a`/:option:`--anchor` option.

        :param parser:        the parser calling this action
        :param args:          arguments
        :param values:        values
        :param option_string: command-line option string
        :type parser:         :class:`ExquiresParser`
        :type args:           :class:`argparse.Namespace`
        :type values:         `list of values`
        :type option_string:  `string`

        :raises:              :class:`argparse.ArgumentError`

        """
        group = getattr(args, args.key)
        if value not in group:
            tup = value, ', '.join([repr(val) for val in group])
            msg = 'invalid choice: %r (choose from %s)' % tup
            raise argparse.ArgumentError(self, msg)
```

```
        setattr(args, self.dest, value)


class SortAction(argparse.Action):

    """Parser action to sort the data by the appropriate metric."""

    def __call__(self, parser, args, value, option_string=None):
        """Parse the :option:`-s`/:option:`--sort` option.

        :param parser:        the parser calling this action
        :param args:          arguments
        :param values:        values
        :param option_string: command-line option string
        :type parser:         :class:`ExquiresParser`
        :type args:           :class:`argparse.Namespace`
        :type values:         `list of values`
        :type option_string:  `string`

        :raises:              :class:`argparse.ArgumentError`

        """
        if value not in args.metrics:
            tup = value, ', '.join([repr(val) for val in args.metrics])
            msg = 'invalid choice: %r (choose from %s)' % tup
            raise argparse.ArgumentError(self, msg)
        setattr(args, self.dest, value)
```

A.9 progress.py

```
#!/usr/bin/env python
# coding: utf-8
#
#  Copyright (c) 2012, Adam Turcotte (adam.turcotte@gmail.com)
#                      Nicolas Robidoux (nicolas.robidoux@gmail.com)
#  License: BSD 2-Clause License
#
#  This file is part of the
#  EXQUIRES (EXtensible QUantitative Image RESampling) test suite
#

"""Display progress info for :ref:`exquires-run` and :ref:`exquires-update`.

When the :option:`-s`/:option:`--silent` option is not selected in
:ref:`exquires-run` or :ref:`exquires-update`, the Progress class is used to
display the appropriate information.

"""

import curses
import time


# pylint: disable-msg=E1101
class Progress(object):

    """This class contains methods for displaying progress in exquires.

    When :ref:`exquires-run` and :ref:`exquires-update` are used without silent
    mode enabled, this class is responsible for displaying information about
    the downsampling, upsampling, and comparison steps and the total progress.

    :param program:   name of the program that is running
    :param proj:      name of the project being used
    :param total_ops: total number of operations
    :type program:    `string`
```

```
    :type proj:        'string'
    :type total_ops:   'integer'

    """

    def __init__(self, program, proj, total_ops):
        """Create a new Progress object."""
        self.scr = curses.initscr()
        curses.cbreak()
        curses.noecho()
        curses.curs_set(0)
        self.program = program
        self.proj = proj
        self.total_ops = total_ops
        self.num_ops = 0

    def __del__(self):
        """Destruct this Progress object and restore the console."""
        self.scr.keypad(0)
        curses.echo()
        curses.nocbreak()
        curses.endwin()

    def __table_top(self, line, label, content):
        """Draw the top row of the progress table.

        This method draws the first row of the progress table, which
        displays the project name. Three lines are used to draw this section
        of the table.

        .. note::

            This is a private method called by
            :meth:'cleanup', :meth:'complete', and :meth:'do_op'.

        .. warning::

            To display the updated progress table, the screen must be
            refreshed by calling :meth:'self.scr.refresh'.

        :param line:    line number to start drawing at
        :param label:   label for this table entry
        :param content: content for this table entry
        :type line:     'integer'
        :type label:    'string'
        :type content:  'string'

        """
        # Top line.
        self.scr.addch(line, 1, curses.ACS_ULCORNER)
        for column in range(2, 14):
            self.scr.addch(line, column, curses.ACS_HLINE)
        self.scr.addch(line, 14, curses.ACS_TTEE)
        for column in range(15, 56):
            self.scr.addch(line, column, curses.ACS_HLINE)
        self.scr.addch(line, 56, curses.ACS_URCORNER)

        # Content line.
        self.scr.addch(line + 1, 1, curses.ACS_VLINE)
        self.scr.addstr(line + 1, 2, label)
        self.scr.addch(line + 1, 14, curses.ACS_VLINE)
        self.scr.addstr(line + 1, 16, content)
        self.scr.addch(line + 1, 56, curses.ACS_VLINE)

        # Bottom line.
        self.scr.addch(line + 2, 1, curses.ACS_LTEE)
        for column in range(2, 14):
            self.scr.addch(line + 2, column, curses.ACS_HLINE)
        self.scr.addch(line + 2, 14, curses.ACS_PLUS)
```

```
        for column in range(15, 56):
            self.scr.addch(line + 2, column, curses.ACS_HLINE)
        self.scr.addch(line + 2, 56, curses.ACS_RTEE)

    def __table_middle(self, line, label, content):
        """Draw one of the middle rows of the progress table.

        This method draws one of the middle rows of the progress table.
        Two lines are used to draw this section of the table.

        .. note::

            This is a private method called by
            :meth:`cleanup`, :meth:`complete`, and :meth:`do_op`.

        .. warning::

            To display the updated progress table, the screen must be
            refreshed by calling :meth:`self.scr.refresh`.

        :param line:    line number to start drawing at
        :param label:   label for this table entry
        :param content: content for this table entry
        :type line:     `integer`
        :type label:    `string`
        :type content:  `string`

        """
        # Content line.
        self.scr.addch(line, 1, curses.ACS_VLINE)
        self.scr.addstr(line, 2, label)
        self.scr.addch(line, 14, curses.ACS_VLINE)
        self.scr.addstr(line, 16, content)
        self.scr.addch(line, 56, curses.ACS_VLINE)

        # Bottom line.
        self.scr.addch(line + 1, 1, curses.ACS_LTEE)
        for column in range(2, 14):
            self.scr.addch(line + 1, column, curses.ACS_HLINE)
        self.scr.addch(line + 1, 14, curses.ACS_PLUS)
        for column in range(15, 56):
            self.scr.addch(line + 1, column, curses.ACS_HLINE)
        self.scr.addch(line + 1, 56, curses.ACS_RTEE)

    def __table_bottom(self, line, label, content):
        """Draw the bottom row of the progress table.

        This method draws one of the middle rows of the progress table.
        Two lines are used to draw this section of the table.

        .. note::

            This is a private method called by
            :meth:`cleanup`, :meth:`complete`, and :meth:`do_op`.

        .. warning::

            To display the updated progress table, the screen must be
            refreshed by calling :meth:`self.scr.refresh`.

        :param line:    line number to start drawing at
        :param label:   label for this table entry
        :param content: content for this table entry
        :type line:     `integer`
        :type label:    `string`
        :type content:  `string`

        """
        # Content line.
```

```
        self.scr.addch(line, 1, curses.ACS_VLINE)
        self.scr.addstr(line, 2, label)
        self.scr.addch(line, 14, curses.ACS_VLINE)
        self.scr.addstr(line, 16, content)
        self.scr.addch(line, 56, curses.ACS_VLINE)

        # Bottom line.
        self.scr.addch(line + 1, 1, curses.ACS_LLCORNER)
        for column in range(2, 14):
            self.scr.addch(line + 1, column, curses.ACS_HLINE)
        self.scr.addch(line + 1, 14, curses.ACS_BTEE)
        for column in range(15, 56):
            self.scr.addch(line + 1, column, curses.ACS_HLINE)
        self.scr.addch(line + 1, 56, curses.ACS_LRCORNER)

    def cleanup(self):
        """Indicate that files are being deleted."""
        self.scr.clear()
        self.scr.addstr(1, 1, self.program, curses.A_BOLD)
        self.__table_top(3, 'PROJECT', self.proj)
        self.__table_middle(6, 'PROGRESS', '0%')
        self.__table_bottom(8, 'STATUS', 'DELETING OLD FILES')
        self.scr.refresh()

    def do_op(self, args, upsampler=None, metric=None):
        """Update the progress indicator.

        * If no upsampler is specified, `operation=downsampling`
        * If an upsampler is specified, but no metric, `operation=upsampling`
        * If an upsampler and metric are specified, `operation=comparing`

        :param args:              arguments
        :param args.image:        image being processed
        :param args.downsampler:  downsampler being used
        :param args.ratio:        resampling ratio being used
        :param upsampler:         upsampler being used
        :param metric:            metric being used
        :type args:               :class:`argparse.Namespace`
        :type args.image:         `string`
        :type args.downsampler:   `string`
        :type args.ratio:         `string`
        :type upsampler:          `string`
        :type metric:             `string`

        """
        percent = int(self.num_ops * 100 / self.total_ops)
        self.num_ops += 1
        self.scr.clear()
        self.scr.addstr(1, 1, self.program, curses.A_BOLD)
        self.__table_top(3, 'PROJECT', self.proj)
        self.__table_middle(6, 'PROGRESS', '{}%'.format(percent))

        if metric:
            self.__table_middle(8, 'STATUS', 'COMPARING')
        elif upsampler:
            self.__table_middle(8, 'STATUS', 'UPSAMPLING')
        else:
            self.__table_middle(8, 'STATUS', 'DOWNSAMPLING')

        self.__table_middle(10, 'IMAGE', args.image)
        self.__table_middle(12, 'DOWNSAMPLER', args.downsampler)

        if metric:
            self.__table_middle(14, 'RATIO', args.ratio)
            self.__table_middle(16, 'UPSAMPLER', upsampler)
            self.__table_bottom(18, 'METRIC', metric)
        elif upsampler:
            self.__table_middle(14, 'RATIO', args.ratio)
            self.__table_bottom(16, 'UPSAMPLER', upsampler)
```

```
        else:
            self.__table_bottom(14, 'RATIO', args.ratio)

        self.scr.refresh()

    def complete(self):
        """Complete the progress indicator.

        Call this method to indicate success once all operations have been
        performed.

        .. note::

            The completion screen is displayed for a half second.

        .. warning::

            To restore the terminal after completion, destruct the
            :class:`Progress` object by calling `del prg`
            (where `prg` is the object to destruct).

        """
        self.scr.clear()
        self.scr.addstr(1, 1, self.program, curses.A_BOLD)
        self.__table_top(3, 'PROJECT', self.proj)
        self.__table_middle(6, 'PROGRESS', '100%')
        self.__table_bottom(8, 'STATUS', 'COMPLETE')
        self.scr.refresh()
        time.sleep(0.5)
```

A.10 report.py

```
#!/usr/bin/env python
# coding: utf-8
#
#  Copyright (c) 2012, Adam Turcotte (adam.turcotte@gmail.com)
#                      Nicolas Robidoux (nicolas.robidoux@gmail.com)
#  License: BSD 2-Clause License
#
#  This file is part of the
#  EXQUIRES (EXtensible QUantitative Image RESampling) test suite
#

"""Print a formatted table of aggregate image difference data.

Each database table in the current project contains data for a single image,
downsampler, and ratio. Each row represents an upsampler and each column
represents a difference metric. By default, the data across all rows and
columns of all tables is aggregated. Use the appropriate option flags to
aggregate across a subset of the database.

  **Features:**

    * :option:`-R`/:option`--ratio` supports hyphenated ranges
      (for example, '1-3 5' gives '1 2 3 5')
    * :option:`-U`/:option:`--up`, :option:`-I`/:option:`--image`,
      :option:`-D`/:option:`--down` and :option:`-M`/:option:`--metric`
      support wildcard characters

"""

from operator import itemgetter

from exquires import database, parsing, stats
```

```
def _print_table(args):
    """Print a table of aggregate image comparison data.

    Since the database contains error data for several images, downsamplers,
    ratios, upsamplers, and metrics, it is convenient to be able to specify
    which of these to consider. This method aggregates the data for each
    relevant column in the appropriate tables.

    .. note::

        This is a private function called by :func:`main`.

    :param args:             arguments
    :param args.dbase_file: database file
    :param args.image:       selected image names
    :param args.down:        selected downsampler names
    :param args.ratio:       selected ratios
    :param args.up:          selected upsampler names
    :param args.metric:      selected metric names
    :param args.metrics_d:   all metric names
    :param args.file:        output file
    :param args.digits:      number of digits to print
    :param args.latex:       `True` if printing a LaTeX-formatted table
    :param args.rank:        `True` if printing Spearman (fractional) ranks
    :param args.merge:       `True` if printing merged Spearman ranks
    :param args.sort:        metric to sort by
    :param args.show_sort:   `True` if the sort column should be displayed
    :type args:              :class:`argparse.Namespace`
    :type args.dbase_file:   `path`
    :type args.image:        `list of strings`
    :type args.down:         `list of strings`
    :type args.ratio:        `list of strings`
    :type args.up:           `list of strings`
    :type args.metric:       `list of strings`
    :type args.metrics_d:    `dict`
    :type args.file:         `path`
    :type args.digits:       `integer`
    :type args.latex:        `boolean`
    :type args.rank:         `boolean`
    :type args.merge:        `boolean`
    :type args.sort:         `string`
    :type args.show_sort:    `boolean`

    """
    # Create a list of the sorting options for each metric.
    metrics_desc = []
    for metric in args.metric:
        metrics_desc.append(int(args.metrics_d[metric][2]))

    # Determine the sort index.
    reverse_index = args.metric.index(args.sort)
    sort_index = reverse_index + 1

    # See if the config file has been poorly edited by the user.
    if not (len(args.image) or len(args.down) or
            len(args.ratio) or len(args.up) or len(args.metric)):
        return

    # Open the database connection.
    dbase = database.Database(args.dbase_file)

    # Get a list of table names to aggregate across.
    tables = dbase.get_tables(args)

    # Get the table (list of lists) of aggregate image difference data.
    printdata = stats.get_aggregate_table(dbase, args.up,
                                          args.metrics_d, tables)

    # Close the database connection.
```

```
    dbase.close()

    if args.rank:
        # Modify the table so it contains Spearman ranks instead of data.
        printdata = stats.get_ranks(printdata, metrics_desc, sort_index)
    elif args.merge:
        # Modify the table so it contains merged ranks instead of data.
        printdata = stats.get_merged_ranks(printdata, metrics_desc, 1)
    else:
        # Sort by the specified index in the appropriate order.
        printdata.sort(key=itemgetter(sort_index),
                       reverse=metrics_desc[reverse_index])

    # Add the table headers.
    if args.merge:
        header = ['upsampler', 'rank']
    else:
        header = ['upsampler']
        for metric in args.metric:
            header.append(metric)

        # Remove the sort column if necessary.
        if not args.show_sort:
            header.pop(1)
            for row in printdata:
                row.pop(sort_index)

    # Pass the printdata to the appropriate table printer.
    if args.latex:
        stats.print_latex(printdata, args, header)
    else:
        stats.print_normal(printdata, args, header)


def main():
    """Run :ref:`exquires-report`.

    Parse the command-line arguments and print the aggregate data table.

    """
    _print_table(parsing.StatsParser(__doc__).parse_args())

if __name__ == '__main__':
    main()
```

A.11 run.py

```
#!/usr/bin/env python
# coding: utf-8
#
#  Copyright (c) 2012, Adam Turcotte (adam.turcotte@gmail.com)
#                      Nicolas Robidoux (nicolas.robidoux@gmail.com)
#  License: BSD 2-Clause License
#
#  This file is part of the
#  EXQUIRES (EXtensible QUantitative Image RESampling) test suite
#

"""Compute error data for the entries in the specified project file.

The project file is read to determine which images, downsamplers, ratios,
upsamplers, and metrics to use. If a database file already exists for this
project, it will be backed up and a new one will be created.

Each image will be downsampled by each of the ratios using each of the
downsamplers. The downsampled images will then be upsampled back to their
```

```
original size (840x840) using each of the upsamplers. The upsampled images will
be compared to the original images using each of the metrics and the results
will be stored in the database file.

If you make changes to the project file and wish to only compute data for these
changes rather than recomputing everything, use :ref:'exquires-update'.

To view aggregated error data, use :ref:'exquires-report'.

"""

from configobj import ConfigObj

from exquires import operations, parsing


def _run(args):
    """Create a new project database and populate it with computed data.

    .. note::

        This is a private function called by :func:'main'.

    :param args:              arguments
    :param args.config_file: current configuration file
    :type args:               :class:'argparse.Namespace'
    :type args.config_file:   'path'

    """
    # Read the configuration file.
    config = ConfigObj(args.config_file)
    args.metrics = config['Metrics']

    # Perform operations.
    operations.Operations(
            [operations.Images(config['Images'],
            [operations.Downsamplers(config['Downsamplers'],
            [operations.Ratios(config['Ratios'],
            [operations.Upsamplers(config['Upsamplers'], args.metrics)])])])]
    ).compute(args)


def main():
    """Run :ref:'exquires-run'.

    Create a database for the specified project file.

    .. warning::

        If a database already exists for this project, it will be overwritten.

    """
    _run(parsing.OperationsParser(__doc__).parse_args())

if __name__ == '__main__':
    main()
```

A.12 stats.py

```
#!/usr/bin/env python
# coding: utf-8
#
#  Copyright (c) 2012, Adam Turcotte (adam.turcotte@gmail.com)
#                      Nicolas Robidoux (nicolas.robidoux@gmail.com)
#  License: BSD 2-Clause License
#
```

```
#  This file is part of the
#  EXQUIRES (EXtensible QUantitative Image RESampling) test suite
#

"""A collection of methods for producing statistical output."""

from operator import itemgetter
from subprocess import check_output

import numpy


def _format_cell(cell, digits):
    """Return a formatted version of this cell of the data table.

    .. note::

        This is a private function called by :func:`print_normal`
        and :func:`print_latex`.

    :param cell:   cell to format
    :param digits: maximum number of digits to display
    :type cell:    `string`
    :type digits:  `integer`

    :return:       the formatted cell
    :rtype:        `string`

    """
    try:
        value = str(float(cell))
        if value[0] is '0':
            return value[1:digits + 2]
        elif value[0] is '-':
            if value[1] is '0':
                return ''.join(['-', value[2:digits + 3]])
            return value[:digits + 2]
        return value[:digits + 1]
    except ValueError:
        # Cell is not a float.
        return cell


def print_normal(printdata, args, header, matrix=False):
    """Print the processed data table with normal formatting.

    :param printdata:   table of data to print
    :param args:        arguments
    :param args.file:   path to write the aggregated error table
    :param args.digits: maximum number of digits to display
    :param header:      table headings
    :param matrix:      `True` if printing a correlation matrix
    :type printdata:    `list of lists`
    :type args:         :class:`argparse.Namespace`
    :type args.file:    `path`
    :type args.digits:  `integer`
    :type header:       `list of strings`
    :type matrix:       `boolean`

    """
    # Print the header.
    if matrix:
        index = 0
        pad = [max((len(head) for head in header))]
        print >> args.file, ''.ljust(pad[0]),
    else:
        index = 1
        pad = [max(len(header[0]), max(len(str(row[0])) for row in printdata))]
        print >> args.file, header[0].ljust(pad[0]),
```

```
    pad[1:] = [max(args.digits + 2, len(head)) for head in header[index:]]
    for i, heading in enumerate(header[index:], 1):
        print >> args.file, heading.rjust(pad[i] + 1),
    print >> args.file

    # Print the remaining rows.
    for j, row in enumerate(printdata):
        # Print the cell for the left column.
        if matrix:
            print >> args.file, header[j].ljust(pad[0]),
        else:
            print >> args.file, str(row[0]).ljust(pad[0]),

        # Print the cells for the remaining columns.
        for i, cell in enumerate(row[index:], 1):
            print >> args.file, _format_cell(
                    cell, args.digits).rjust(pad[i] + 1),
        print >> args.file


def print_latex(printdata, args, header, matrix=False):
    """Print the processed data table with LaTeX formatting.

    :param printdata:   table of data to print
    :param args:        arguments
    :param args.file:   path to write the aggregated error table
    :param args.digits: maximum number of digits to display
    :param header:      table headings
    :param matrix:      `True` if printing a correlation matrix
    :type printdata:    `list of lists`
    :type args:         :class:`argparse.Namespace`
    :type args.file:    `path`
    :type args.digits:  `integer`
    :type header:       `list of strings`
    :type matrix:       `boolean`

    """
    # No padding is necessary since this is a LaTeX table.
    print >> args.file, '\\begin{table}[t]'
    print >> args.file, '\\centering'
    print >> args.file, '\\begin{tabular}{|l||',
    for dummy in range(len(printdata[0]) - 1):
        print >> args.file, 'r|',
    print >> args.file, '}'
    print >> args.file, '\\hline'

    # Print the header.
    if matrix:
        index = 0
    else:
        index = 1
        print >> args.file, header[0],
    for heading in header[index:]:
        print >> args.file, ' & {}'.format(heading),
    print >> args.file, '\\\\'
    print >> args.file, '\\hline'

    # Print the remaining rows.
    for j, row in enumerate(printdata):
        # Print the cell for the left column.
        if matrix:
            print >> args.file, header[j],
        else:
            print >> args.file, row[0],

        # Print the cells for the remaining columns.
        for cell in row[index:]:
            print >> args.file, ' & {}'.format(
                    _format_cell(cell, args.digits)
```

```
            ),
        print >> args.file, '\\\\'

    print >> args.file, '\\hline'
    print >> args.file, '\\end{{tabular}}'
    print >> args.file, '\\caption{{Insert a caption}}'
    print >> args.file, '\\label{{tab:table1}}'
    print >> args.file, '\\end{{table}}'


def get_ranks(printdata, metrics_desc, sort_index):
    """Return a table of Spearman (Fractional) ranks based on a data table.

    :param printdata:    table of data to print
    :param metrics_desc: list of 0s and 1s (where 1 is 'descending')
    :param sort_index:   index of the column to sort by
    :type printdata:     'list of lists'
    :type metrics_desc:  'list of integers'
    :type sort_index:    'integer'

    :return:             table of ranks
    :rtype:              'list of lists'

    """
    data = [x[:] for x in printdata]
    for j in range(1, len(printdata[0])):
        data.sort(key=itemgetter(j), reverse=metrics_desc[j - 1])
        i = 0
        end = len(printdata) - 1
        while i <= end:
            if i == end or data[i][j] != data[i + 1][j]:
                data[i][j] = i + 1
                i += 1
            else:  # We have at least one tie.
                matches = 1
                for k in range(i + 2, len(printdata)):
                    if data[i][j] != data[k][j]:
                        break
                    matches += 1
                rank = i + 1 + matches * 0.5
                for k in range(i, i + 1 + matches):
                    data[k][j] = rank
                i += matches + 1

    data.sort(key=itemgetter(sort_index))
    return data


def get_merged_ranks(printdata, metrics_desc, sort_index):
    """Return a table of merged Spearman ranks based on a data table.

    :param printdata:    table of data to print
    :param metrics_desc: list of 0s and 1s (where 1 is 'descending')
    :param sort_index:   index of the column to sort by
    :type printdata:     'list of lists'
    :type metrics_desc:  'list of integers'
    :type sort_index:    'integer'

    :return:             table of merged ranks
    :rtype:              'list of lists'

    """
    # Get the Spearman (Fractional) ranks.
    data = get_ranks(printdata, metrics_desc, sort_index)

    # Combine the ranks into a single column.
    for row in data:
        row[1:] = [numpy.average(row[1:])]
```

```
    # Convert the averages back into ranks.
    return get_ranks(data, [0], sort_index)


def get_aggregate_table(dbase, upsamplers, metrics_d, tables):
    """Return a table of aggregate image difference data.

    :param dbase:       connected database
    :param upsamplers: upsamplers (rows) of the table
    :param metrics_d:  metrics (columns) of the table in dictionary form
    :param tables:      names of database tables to aggregate across
    :type dbase:        :class:`database.Database`
    :type upsamplers:   `list of strings`
    :type metrics_d:    `dict`
    :type tables:       `list of strings`

    :return:            table of aggregate image difference data
    :rtype:             `list of lists`

    """
    metrics = metrics_d.keys()
    metrics_str = ','.join(metrics)
    aggregate_table = []
    for upsampler in upsamplers:
        datarow = [upsampler]

        # Create a new dictionary.
        metric_lists = {}
        for metric in metrics:
            metric_lists[metric] = []
        for table in tables:
            row = dbase.get_error_data(table, upsampler, metrics_str)
            for metric in metrics:
                metric_lists[metric].append(row[metric])
        for metric in metrics:
            # Aggregate the error data using the appropriate method.
            metric_list = ' '.join(str(x) for x in metric_lists[metric])
            met = metrics_d[metric][1].format(metric_list).split()
            datarow.append(float(check_output(met)))
        aggregate_table.append(datarow)

    # Return the table of aggregate image difference data.
    return aggregate_table
```

A.13 tools.py

```
#!/usr/bin/env python
# coding: utf-8
#
#  Copyright (c) 2012, Adam Turcotte (adam.turcotte@gmail.com)
#                      Nicolas Robidoux (nicolas.robidoux@gmail.com)
#  License: BSD 2-Clause License
#
#  This file is part of the
#  EXQUIRES (EXtensible QUantitative Image RESampling) test suite
#

"""A collection of convenience methods."""

from collections import OrderedDict
import os


def prune_metrics(keys, metrics_d):
    """Prune a dictionary of metrics using a list of keys.
```

```
    :param keys:      keys to retain
    :param metrics_d: metrics to prune
    :type keys:       `list of strings`
    :type metrics_d:  `dict`

    :return:          pruned metrics
    :rtype:           `dict`

    """
    result = OrderedDict()
    for key in keys:
        result[key] = metrics_d[key]
    return result


def create_dir(base_dir, relative_dir=''):
    """Create a directory if it doesn't already exist and return it.

    :param base_dir:     base directory within which to create the directory
    :param relative_dir: directory to create inside the base directory
    :type base_dir:      `path`
    :type relative_dir:  `path`

    :return:             the created directory
    :rtype:              `path`

    """
    directory = os.path.join(base_dir, relative_dir)
    if not os.path.exists(directory):
        os.makedirs(directory)
    return directory
```

A.14 update.py

```
#!/usr/bin/env python
# coding: utf-8
#
#  Copyright (c) 2012, Adam Turcotte (adam.turcotte@gmail.com)
#                      Nicolas Robidoux (nicolas.robidoux@gmail.com)
#  License: BSD 2-Clause License
#
#  This file is part of the
#  EXQUIRES (EXtensible QUantitative Image RESampling) test suite
#

"""Compute new error data for changes to the user-specified project file.

The project file is inspected to determine which changes have been made. Items
that have been removed will result in entries being removed from the database.
Items that have been changed or added will result in new data being computed
and added to the database file. If no changes have been made to the project
file, the database will not be updated.

If you wish to recompute all data based on your project file rather than simply
updating it with the changes, use :ref:`exquires-run`.

To view aggregated error data, use :ref:`exquires-report`.

"""

import argparse

from configobj import ConfigObj

import exquires.operations as operations
import exquires.parsing as parsing
```

```
def _subtract(dict1, dict2):
    """Subtract dictionary `dict2` from `dict1` and return the difference.

    This function creates a new `dict`, then iterates over `dict1` and adds
    all entries that are not found in `dict2`.

    .. note::

        This is a private function called by :func:`_get_namespaces`.

    :param dict1: dictionary to subtract from
    :param dict2: dictionary to subtract
    :type dict1:  `dict`
    :type dict2:  `dict`

    :return:      dict1 - dict2
    :rtype:       `dict`

    """
    result = {}
    for key in dict1.keys():
        if not key in dict2 or dict1[key] != dict2[key]:
            result[key] = dict1[key]
    return result


def _get_namespaces(config_file, config_bak):
    """Return all necessary configuration namespaces.

    This function returns four namespaces that specify which images,
    downsamplers, ratios, upsamplers, and metrics to use when creating
    or updating a project database:

    * `current` -- all entries in current project file
    * `new` -- entries only in currenty project file
    * `old` -- entries only in previous project file
    * `same` -- entries common to both project files

    .. note::

        This is a private function called by :func:`_update`.

    :param config_file: current configuration file
    :param config_bak:  previous configuration file
    :type config_file:  `path`
    :type config_bak:   `path`

    :return:            the current, new, old, and same namespaces
    :rtype:             :class:`argparse.Namespace`

    """

    # Read the current configuration file.
    current = argparse.Namespace()
    config_current = ConfigObj(config_file)
    current.images = config_current['Images']
    current.ratios = config_current['Ratios']
    current.downsamplers = config_current['Downsamplers']
    current.upsamplers = config_current['Upsamplers']
    current.metrics = config_current['Metrics']

    # Read the configuration file last used to update the database.
    previous = argparse.Namespace()
    config_previous = ConfigObj(config_bak)
    previous.images = config_previous['Images']
    previous.ratios = config_previous['Ratios']
    previous.downsamplers = config_previous['Downsamplers']
```

```
    previous.upsamplers = config_previous['Upsamplers']
    previous.metrics = config_previous['Metrics']

    # Construct dictionaries from the current and previous configurations.
    new = argparse.Namespace()
    new.images = _subtract(current.images, previous.images)
    new.ratios = _subtract(current.ratios, previous.ratios)
    new.downsamplers = _subtract(current.downsamplers, previous.downsamplers)
    new.upsamplers = _subtract(current.upsamplers, previous.upsamplers)
    new.metrics = _subtract(current.metrics, previous.metrics)
    old = argparse.Namespace()
    old.images = _subtract(previous.images, current.images)
    old.ratios = _subtract(previous.ratios, current.ratios)
    old.downsamplers = _subtract(previous.downsamplers, current.downsamplers)
    same = argparse.Namespace()
    same.images = _subtract(current.images, new.images)
    same.ratios = _subtract(current.ratios, new.ratios)
    same.downsamplers = _subtract(current.downsamplers, new.downsamplers)
    same.upsamplers = _subtract(current.upsamplers, new.upsamplers)
    same.metrics = _subtract(current.metrics, new.metrics)

    # Return the namespaces.
    return current, new, old, same


def _update(args):
    """Update the database.

    .. note::

        This is a private function called by :func:`main`.

    :param args:              arguments
    :param args.config_file: current project file
    :param args.config_bak:  previous project file
    :type args:               :class:`argparse.Namespace`
    :type args.config_file:   `path`
    :type args.config_bak:    `path`

    """
    # Get the various namespaces for this project update.
    current, new, old, same = _get_namespaces(args.config_file,
                                              args.config_bak)
    args.met_same = same.metrics
    args.metrics = current.metrics

    # Define operations.
    same.up_obj = operations.Upsamplers(same.upsamplers, new.metrics, True)
    new.up_obj = operations.Upsamplers(new.upsamplers, current.metrics)
    current.up_obj = operations.Upsamplers(current.upsamplers, current.metrics)
    same.rat_obj = operations.Ratios(same.ratios,
                                     [same.up_obj, new.up_obj], True)
    new.rat_obj = operations.Ratios(new.ratios, [current.up_obj])
    current.rat_obj = operations.Ratios(current.ratios, [current.up_obj])
    same.down_obj = operations.Downsamplers(same.downsamplers,
                                            [same.rat_obj, new.rat_obj], True)
    new.down_obj = operations.Downsamplers(new.downsamplers, [current.rat_obj])
    current.down_obj = operations.Downsamplers(current.downsamplers,
                                               [current.rat_obj])
    same.img_obj = operations.Images(same.images,
                                     [same.down_obj, new.down_obj], True)
    new.img_obj = operations.Images(new.images, [current.down_obj])
    operations.Operations([same.img_obj, new.img_obj]).compute(args, old)


def main():
    """Run :ref:`exquires-update`.

    Update the project database based on changes to the project file.
```

```
    .. note::

        If the update fails, the previous database will be restored.

    """
    _update(parsing.OperationsParser(__doc__, True).parse_args())

if __name__ == '__main__':
    main()
```

B EXQUIRES examples: source code

The following code was written by Adam Turcotte and Nicolas Robidoux. An up-to-date version of this code is in the examples folder of the EXQUIRES GitHub repository [148].

B.1 nohalo.cpp

```
/* Wrapper for VIPS Nohalo subdivision followed by LBB
 * (locally bounded bicubic) interpolation resampler
 *
 * This program is a wrapper around the Nohalo implementation found in VIPS.
 * Unfortunately, VIPS does not conform to the pixel alignment convention used in
 * the EXQUIRES test suite. In order to alleviate this problem, the leftmost
 * column of the image is duplicated and the topmost row of the resulting image is
 * duplicated before performing the resampling step.
 *
 * Note: this program only supports aspect ratio preserving upsampling.
 *
 * Code written by Adam Turcotte with contributions by Nicolas Robidoux
 *
 * Published December 9, 2012
 */

#include <vips/vips>
#include <cstdlib>
#include <iostream>

using namespace vips;
using std::cout;
using std::endl;


int
main (int argc, char **argv)
{
    // Check for correct number of command-line arguments
    if (argc != 5) {
        cout << "usage: nohalo image_in image_out enlargement_factor "
                "(0: sRGB | 1: linear)" << endl;
        return (1);
    }

    try {
        // Define sRGB profile path and method name
        char profile[] = "/usr/local/lib/python2.7/dist-packages"
                         "/exquires/sRGB_IEC61966-2-1_black_scaled.icc";
        char method[] = "nohalo";
```

```
        // Read input image
        VImage image_in (argv[1]);

        // Get ratio, size, and determine if we're using linear light
        double ratio = atof (argv[3]);
        bool linear = atoi (argv[4]);
        double dx = -0.5 * (ratio + 1);
        int size = (int) (ratio * image_in.Xsize() + 0.5);

        // Import from sRGB to XYZ if linear option selected
        if (linear)
            image_in = image_in.icc_import (profile, 1);

        // Create the temporarily padded image
        VImage col = image_in.extract_area (0, 0, 1, image_in.Ysize ());
        VImage added_col = col.lrjoin (image_in);
        VImage row = added_col.extract_area (0, 0, added_col.Xsize (), 1);
        VImage padded = row.tbjoin (added_col);

        // Prepare the output image
        VImage image_out = padded.affinei (method, ratio, 0, 0,ratio,
                                           dx, dx, 0, 0, size, size);

        // Export from XYZ to sRGB if linear option selected
        if (linear)
            image_out = image_out.icc_export_depth (16, profile, 1);

        // Write output image
        image_out.write (argv[2]);
    }
    catch (VError &e) {
        e.perror (argv[0]);
    }

    return (0);
}
```

C Refactored SSIM: source code

The following code was written by Nicolas Robidoux and Adam Turcotte. It conforms to the API (application programming interface) specified by the original SSIM programmer, Zhou Wang.

C.1 ssim_index_refactored.m

```
function [mssim, ssim_map] = ssim_index_refactored(img1, img2, K, L, window)

%========================================================================
%
% Refactored SSIM Index, Version 2.0
% Modifications Copyright(c) 2011 Adam Turcotte and Nicolas Robidoux
%
% All Rights Reserved.
%
% Permission to use, copy, or modify this software and its documentation
% for educational and research purposes only and without fee is hereby
% granted, provided that this copyright notice and the original authors'
% names appear on all copies and supporting documentation. This program
% shall not be used, rewritten, or adapted as the basis of a commercial
% software or hardware product without first obtaining permission of the
% authors. The authors make no representations about the suitability of
% this software for any purpose. It is provided "as is" without express
% or implied warranty.
%
% See below for additional copyright notices.
%
% ------------------------------------------------------------------------
%
% A. Turcotte is an M.Sc. candidate at the Department of Mathematics
% and Computer Science, Laurentian University, Sudbury, ON, Canada.
%
% N. Robidoux is an image processing and applied mathematics consultant.
%
% Last Modified: 2011-10-02
%
% ------------------------------------------------------------------------
%
% This Matlab code implements a refactored computation of SSIM that
% requires one blur less (4 instead of 5), as well as fewer binary and
% unary operations. In addition, this version reduces memory usage with
% in-place functions. As a result, it runs faster and supports larger
% input images.
%
% Warning: To improve the argument parsing, the last two arguments of Z.
```

```
%          Wang's version have been swapped: 'L' is now specified before
%          'window' (instead of after).
%
% Note: The above improvements only apply when the standard blur 'window'
%       (11x11 Gaussian blur with sigma=1.5) is used. If you supply your
%       own window the computation falls back to Z. Wang's version.
%
%=========================================================================

%=========================================================================
%SSIM Index, Version 1.0
%Copyright(c) 2003 Zhou Wang
%All Rights Reserved.
%
%The author is with Howard Hughes Medical Institute, and Laboratory
%for Computational Vision at Center for Neural Science and Courant
%Institute of Mathematical Sciences, New York University.
%
%------------------------------------------------------------------------
%Permission to use, copy, or modify this software and its documentation
%for educational and research purposes only and without fee is hereby
%granted, provided that this copyright notice and the original authors'
%names appear on all copies and supporting documentation. This program
%shall not be used, rewritten, or adapted as the basis of a commercial
%software or hardware product without first obtaining permission of the
%authors. The authors make no representations about the suitability of
%this software for any purpose. It is provided "as is" without express
%or implied warranty.
%------------------------------------------------------------------------
%
%This is an implementation of the algorithm for calculating the
%Structural SIMilarity (SSIM) index between two images. Please refer
%to the following paper:
%
%Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image
%quality assessment: From error measurement to structural similarity"
%IEEE Transactios on Image Processing, vol. 13, no. 1, Jan. 2004.
%
%Kindly report any suggestions or corrections to zhouwang@ieee.org
%
%------------------------------------------------------------------------
%
%Input : (1) x: the first image being compared
%        (2) y: the second image being compared
%        (3) K: constants in the SSIM index formula (see the above
%            reference). defualt value: K = [0.01 0.03]
%        (4) L: dynamic range of the images. default: L = 255
%        (5) window: local window for statistics (see the above
%            reference). default widnow is Gaussian given by
%            window = fspspecial('gaussian', 11, 1.5);
%
%Output: (1) mssim: the mean SSIM index value between 2 images.
%            If one of the images being compared is regarded as
%            perfect quality, then mssim can be considered as the
%            quality measure of the other image.
%            If x = y, then mssim = 1.
%        (2) ssim_map: the SSIM index map of the test image. The map
%            has a smaller size than the input images. The actual size:
%            size(x) - size(window) + 1.
%
%Default Usage:
%   Given 2 test images x and y, whose dynamic range is 0-255
%
%   [mssim ssim_map] = ssim_index(x, y);
%
%Advanced Usage:
%   User defined parameters. For example
%
%   K = [0.05 0.05];
```

```
%   window = ones(8);
%   L = 100;
%   [mssim ssim_map] = ssim_index(x, y, K, L, window);
%
%See the results:
%
%   mssim                           %Gives the mssim value
%   imshow(max(0, ssim_map).^4)     %Shows the SSIM index map
%
%========================================================================

if (nargin >= 2 && nargin <= 4)

   if (size(img1) ~= size(img2))
      mssim = -Inf;
      ssim_map = -Inf;
      return;
   end

   [M N] = size(img1);
   if ((M < 11) || (N < 11))
      mssim = -Inf;
      ssim_map = -Inf;
      return;
   end

   if (nargin == 2)
      K(1) = 0.01;          %
      K(2) = 0.03;          % default settings
      L = 255;              %
   elseif (nargin == 3)
      L = 255;
      if (length(K) == 2)
         if (K(1) < 0 || K(2) < 0)
            mssim = -Inf;
            ssim_map = -Inf;
            return;
         end
      else
         mssim = -Inf;
         ssim_map = -Inf;
         return;
      end
   elseif (nargin == 4)
      if (length(K) == 2)
         if (K(1) < 0 || K(2) < 0)
            mssim = -Inf;
            ssim_map = -Inf;
            return;
         end
      else
         mssim = -Inf;
         ssim_map = -Inf;
         return;
      end
   end

   C1 = (K(1)*L)^2;
   C2 = (K(2)*L)^2;
   if (C1<=0 || C2<=0)
      mssim = -Inf;
      ssim_map = -Inf;
      return;
   end
   c2 = C1 + C2;

   x = double(img1);
   y = double(img2);
```

```
    window = fspecial('gaussian', 11, 1.5);
    window2 = 2*window;

    if (C1 > 0 & C2 > 0)

       ssim_map = filter2(window, x, 'valid');
       t        = filter2(window, y, 'valid');
       u        = t - ssim_map;
       t        = 2 * ( t .* ssim_map ) + C1;
       ssim_map = filter2(window2, x .* y, 'valid');
       ssim_map = ( ssim_map + c2 - t ) .* t;
       u        = u.^2 + t;
       t        = filter2(window, x.^2 + y.^2, 'valid');
       ssim_map = ssim_map ./ ( (t + c2 - u ) .* u );

    else

       numerator2      = filter2(window, x, 'valid');
       numerator1      = filter2(window, y, 'valid');
       denominator1    = numerator1 - numerator2;
       numerator1      = 2 * ( numerator1 .* numerator2 ) + C1;
       numerator2      = filter2(window2, x .* y, 'valid');
       numerator2      = numerator2 + c2 - numerator1;
       denominator1    = denominator1.^2 + numerator1;
       denominator2    = filter2(window, x.^2 + y.^2, 'valid');
       denominator2    = denominator2 + c2 - denominator1;
       ssim_map        = ones(size(numerator2));
       index           = (denominator1 > 0 & denominator2 > 0);
       ssim_map(index) = (numerator1(index).*numerator2(index)) ...
                         ./(denominator1(index).*denominator2(index));
       index           = (denominator1 > 0) & (denominator2 <= 0);
       ssim_map(index) = numerator1(index)./denominator1(index);

    end

    mssim = mean2(ssim_map);
    return;

elseif (nargin == 5)

    if (size(x) ~= size(y))
       mssim = -Inf;
       ssim_map = -Inf;
       return;
    end

    [M N] = size(x);
    if ((M < 11) || (N < 11))
       mssim = -Inf;
       ssim_map = -Inf;
       return;
    end

    [H W] = size(window);
    if ((H*W) < 4 || (H > M) || (W > N))
       ssim_index = -Inf;
       ssim_map = -Inf;
       return;
    end
    if (length(K) == 2)
       if (K(1) < 0 || K(2) < 0)
          ssim_index = -Inf;
          ssim_map = -Inf;
          return;
       end
    else
       ssim_index = -Inf;
       ssim_map = -Inf;
       return;
```

```
    end

    C1 = (K(1)*L)^2;
    C2 = (K(2)*L)^2;
    window = window/sum(sum(window));
    x = double(x);
    y = double(y);

    mu1   = filter2(window, x, 'valid');
    mu2   = filter2(window, y, 'valid');
    mu1_sq = mu1.*mu1;
    mu2_sq = mu2.*mu2;
    mu1_mu2 = mu1.*mu2;
    sigma1_sq = filter2(window, x.*x, 'valid') - mu1_sq;
    sigma2_sq = filter2(window, y.*y, 'valid') - mu2_sq;
    sigma12 = filter2(window, x.*y, 'valid') - mu1_mu2;

    if (C1 > 0 & C2 > 0)
       ssim_map = ((2*mu1_mu2 + C1).*(2*sigma12 + C2)) ...
                  ./((mu1_sq + mu2_sq + C1).*(sigma1_sq + sigma2_sq + C2));
    else
       numerator1 = 2*mu1_mu2 + C1;
       numerator2 = 2*sigma12 + C2;
       denominator1 = mu1_sq + mu2_sq + C1;
       denominator2 = sigma1_sq + sigma2_sq + C2;
       ssim_map = ones(size(mu1));
       index = (denominator1.*denominator2 > 0);
       ssim_map(index) = (numerator1(index).*numerator2(index)) ...
                         ./(denominator1(index).*denominator2(index));
       index = (denominator1 ~= 0) & (denominator2 == 0);
       ssim_map(index) = numerator1(index)./denominator1(index);
    end
    mssim = mean2(ssim_map);
    return;
else
    ssim_index = -Inf;
    ssim_map = -Inf;
    return;

end
```

D EXQUIRES user manual

An up-to-date online version of this manual is found at the EXQUIRES home [149].

The EXQUIRES (EXtensible QUantitative Image RESampling) Test Suite

Release 0.9.9.3

Adam Turcotte and Nicolas Robidoux

December 20, 2012

CONTENTS

CHAPTER

ONE

ABOUT THE EXQUIRES TEST SUITE

Website http://exquires.ca

PyPI http://pypi.python.org/pypi/exquires

GitHub http://github.com/aturcotte/exquires

License BSD 2-Clause License

Authors Adam Turcotte and Nicolas Robidoux

1.1 What is EXQUIRES?

The **EXQUIRES** test suite (hereby referred to as **EXQUIRES**) is an open source framework for assessing the accuracy of image upsampling methods. **EXQUIRES** can also be used to compare image difference metrics, or to measure the impact of various factors, including test image selection and properties, downsampler choice, resizing ratio, etc.

An upsampler's performance is based on its ability to reconstruct test images from various reduced versions. The downsampler used to reduce the images has an influence on the re-enlargements, so any number of downsampling methods can be used. The difference between the re-enlargements and the orginal images is determined by using image comparison metrics. When viewing the comparison data, it is possible to aggregate across any combination of test images, downsamplers, and resampling ratios.

EXQUIRES is fully extensible: External applications can be used alongside its own to compute downsampled and upsampled images as well as image difference metrics. The following components of **EXQUIRES** are configurable:

- Test Images
- Resampling Ratios
- Downsampling Methods
- Upsampling Methods
- Difference Metrics

1.2 Technical Notes

EXQUIRES is written in Python (requiring version 2.7 or higher) and makes use of several modules, including the following:

- argparse – command-line argument parsing
- *configobj – reading and writing* `.ini` *files*
- curses – displaying progress information
- fnmatch – handling wildcard characters
- inspect – listing a class' methods
- numpy – applying operations to lists of numbers
- re – handling arguments with hypenated ranges
- sqlite3 – database for storing image comparison data
- subprocess – calling external applications

- vipsCC – Python interface for VIPS

The following image processing applications are also used:

- ImageMagick – resampling images
- VIPS – computing image difference metrics

CHAPTER

TWO

EXQUIRES DOCUMENTATION

2.1 Online Documentation

The documentation for the latest release version of **EXQUIRES** can be viewed online at http://exquires.ca.

2.2 Building the Documentation

The **EXQUIRES** documentation/website is built using Sphinx.

Before building the documentation, you must first perform the following tasks:

- Install Sphinx (`$ sudo apt-get install python-sphinx`)
- Install **EXQUIRES**

To produce the HTML documentation (same as the online documentation):

- From the `docs` directory, run: `$ make html`
- This will produce HTML documentation in the `_build/html/` directory
- Open `_build/html/index.html` with your browser

To produce the PDF manual:

- From the `docs` directory, run: `$ make latexpdf`
- This will produce LaTeX files in the `_build/latex` directory and run them through pdflatex

- Open `exquires.pdf` to view the manual.

CHAPTER

THREE

INSTALLING EXQUIRES

3.1 Basic Installation Instructions

EXQUIRES can be installed from PyPI using pip:

```
pip install -U exquires
```

Alternatively, download the source distribution from PyPI, unarchive, and run:

```
python setup.py install
```

3.2 Detailed Installation Instructions for Debian

The following instructions are for Debian/Ubuntu/Mint Linux. For other platforms, the setup is generally the same, with the exception of installing system dependencies.

3.2.1 Requirements

EXQUIRES requires ImageMagick 6.8.0-2 or newer, VIPS 7.24 or newer, Python 2.7, and the Python packages ConfigObj and NumPy.

3.2.2 Installing ImageMagick from source

- Install dependencies on Debian/Ubuntu/Mint:

```
$ sudo apt-get install imagemagick libmagick++-dev subversion
```

- Download and extract the ImageMagick source:

```
$ wget http://www.imagemagick.org/download/ImageMagick.tar.gz
$ tar xvfz ImageMagick.tar.gz
```

- Configure, compile and install ImageMagick:

```
$ cd ImageMagick-6.8.X-X
$ CFLAGS="-march=native -O2" CXXFLAGS="-march=native -O2" ./configure --enable-hdri
$ make
$ sudo make install
```

- Configure the dynamic linker run-time bindings:

```
$ sudo ldconfig /usr/local/lib
```

- (Optional) Ensure that the correct version is now installed:

```
$ identify -version
$ pkg-config --modversion ImageMagick
```

- Updating ImageMagick development version:

```
$ cd ImageMagick-6.8.X-X
$ sudo make uninstall
$ make clean
$ svn update
$ CFLAGS="-march=native -O2" CXXFLAGS="-march=native -O2" ./configure --enable-hdri
$ make
$ sudo make install
$ sudo ldconfig /usr/local/lib
```

3.2.3 Installing EXQUIRES

- Install remaining dependencies:

```
$ sudo apt-get install python-pip python-configobj python-dev python-numpy python-vipscc libvips-tools
```

- Install **EXQUIRES** from PyPI using pip:

```
$ sudo pip install -U exquires
```

3.2.4 Installing latest EXQUIRES development branch

- The latest development version can be installed from the GitHub repository:

```
sudo pip install -e git+http://github.com/aturcotte/exquires.git#egg=exquires
```

CHAPTER

FOUR

USING EXQUIRES

4.1 Usage Overview

- Obtain suitable 16 bit 840x840 test images
- Use *exquires-new* (page 23) to create a new project file
- Modify the project file to suit your needs
- Use *exquires-run* (page 24) to compute the image difference data
- Use *exquires-update* (page 25) to compute only the new data after editing the project file
- Use *exquires-report* (page 25) to produce tables of aggregated data
- Use *exquires-correlate* (page 26) to produce Spearman's rank cross-correlation matrices

4.2 Usage Instructions

EXQUIRES comes with several *Programs* (page 23), each including a `-h`/`--help` option to display usage information and a `-v`/`--version` option to display the version number.

These five main programs can be used to create and maintain a project, which can be specified with the `-p`/`--proj` option:

- *exquires-new* (page 23)
- *exquires-run* (page 24)

- *exquires-update* (page 25)
- *exquires-report* (page 25)
- *exquires-correlate* (page 26)

These two programs are responsible for computing image differences and aggregating the results:

- *exquires-compare* (page 27)
- *exquires-aggregate* (page 29)

The following sections will explain how to make use of these programs to compute data and view aggregated results and cross-correlation matrices.

4.2.1 Obtaining suitable test images

EXQUIRES is designed to use sRGB TIFF images with 16 bits per sample (48 bits per pixel) and a width and height of 840 pixels. One image (`wave.tif`) is included as a default selection.

A separate distribution of test images converted from RAW is available at http://www.imagemagick.org/download/image-bank/16bit840x840images/. The examples in this section make use of several images from this collection.

The easiest way to obtain a copy of the image bank is as follows:

```
$ wget -r -nH --cut-dirs=3 ftp://ftp.imagemagick.org/pub/ImageMagick/image-bank/16bit840x840images/
```

4.2.2 Creating a new project file

A project file is a `.ini` file that tells **EXQUIRES** which of the following to use:

- Images
- Resampling Ratios
- Downsamplers
- Upsamplers
- Difference Metrics

The basic syntax to create a new project using *exquires-new* (page 23) is:

```
$ exquires-new
```

which will create the project file `project1.ini` and include the image wave.tif along with a default collection of ratios, downsamplers, upsamplers, and metrics.

In order to specify a project name and a set of test images, type one of the following:

```
$ exquires-new -p my_project -I my_images
$ exquires-new --proj my_project --image my_images
```

where `my_project` is a name to identify your project and `my_images` is a list (wildcards are supported) of images with the following properties:

File Format TIFF

Colour Space sRGB

Bit Depth 16 bits/sample (48 bits/pixel)

Size 840x840 pixels

To demonstrate, we will create a new project **example_proj** using the 16bit840x840images collection:

```
$ exquires-new -p example_proj -I /path/to/16bit840x840images/images/*
```

4.2.3 Customizing the project file

Once a project file has been generated, you can manually edit it to suit your needs. For our example project **example_proj**, we have a project file `example_proj.ini` and we will look at each section in detail.

Images

This section lists the paths to the test images that will be used. We will keep this example project small by removing all but two of the 16bit840x840images, `apartments.tif` and `cabins.tif`.

```
# TEST IMAGES
# Images are 16 bit sRGB TIFFs with a width and height of 840 pixels.
# Any images that are added must conform to this standard.
[Images]
apartments = /path/to/16bit840x840images/images/apartments.tif
cabins = /path/to/16bit840x840images/images/cabins.tif
```

Notice that **EXQUIRES** has also assigned default names for these images, which you can also modify.

Ratios

This section lists the resampling ratios and specifies the width and height of the downsampled image for each ratio. Here are the default ratios:

```
# RESAMPLING RATIOS
# The test images are downsampled to the specified sizes.
# Each size is obtained by dividing 840 by the ratio.
[Ratios]
2 = 420
3 = 280
4 = 210
5 = 168
6 = 140
7 = 120
8 = 105
```

Downsamplers

This section lists the downsampling methods that will be used to reduce each of the test images. We have edited our example project to include a small subset of the defaults.

```
# DOWNSAMPLING COMMANDS
# To add a downsampler, provide the command to execute it.
# The command can make use of the following replacement fields:
#     {0} = input image
#     {1} = output image
#     {2} = downsampling ratio
#     {3} = downsampled size (width or height)
# WARNING: Be sure to use a unique name for each downsampler.
[Downsamplers]
box_srgb = magick {0} -filter Box -resize {3}x{3} -strip {1}
box_linear = magick {0} -colorspace RGB -filter Box -resize {3}x{3} -colorspace sRGB -strip {1}
nearest_srgb = magick {0} -filter Point -resize {3}x{3} -strip {1}
nearest_linear = magick {0} -colorspace RGB -filter Point -resize {3}x{3} -colorspace sRGB -strip {1}
```

Note that the ImageMagick commands in this example make use of numbered replacement fields to denote the command-line arguments. If you wish to add your own downsampling method, you must use **{0}** and **{1}** to specify the input and output images, and either **{2}** or **{3}** (or both) to specify the size of the reduced image.

Also note that the methods suffixed with **_srgb** do not perform any colour space conversion within the resize operations, meaning that the sRGB images are downsampled using linear averaging even though sRGB is a non-linear colour space. The methods suffixed with **_linear** convert the input image to linear RGB with sRGB primaries before downsampling, then convert the result back to sRGB, using the ImageMagick command **-colorspace**. Such suffixes are useful because

they allow one to separately aggregate the results of only downsampling or upsampling using the two main "tracks" without having to list the methods individually. In the same spirit if, for example, you were to program downsamplers or upsamplers that convert into and out of sRGB using ICC profiles, we would suggest that you use something like the **_icc** suffix; if you were to go through the XYZ colourspace, we would suggest **_xyz**.

Upsamplers

This section lists the upsampling methods that will be used to re-enlarge each of the downsampled images, and makes use of the same replacement fields as the Downsamplers section.

Since the purpose of ***EXQUIRES*** *is to assess the accuracy of upsampling methods, you may wish to add your own method to see how it ranks alongside pre-existing* methods. For example, we can compare the Nohalo method with several Lanczos variations.

```
# UPSAMPLING COMMANDS
# To add an upsampler, provide the command to execute it.
# The command can make use of the following replacement fields:
#     {0} - input image
#     {1} - output image
#     {2} - upsampling ratio
#     {3} - upsampled size (always 840)
[Upsamplers]
lanczos2_srgb = magick {0} -filter Lanczos2 -resize {3}x{3} -strip {1}
lanczos2_linear = magick {0} -colorspace RGB -filter Lanczos2 -resize {3}x{3} -colorspace sRGB -strip {1}
lanczos3_srgb = magick {0} -filter Lanczos -resize {3}x{3} -strip {1}
lanczos3_linear = magick {0} -colorspace RGB -filter Lanczos -resize {3}x{3} -colorspace sRGB -strip {1}
nohalo_srgb = nohalo {0} {1} {2} 0
nohalo_linear = nohalo {0} {1} {2} 1
```

The **nohalo** program is found in `nohalo.cpp`, which uses VIPS to resample the image (using a trick to produce a result that conforms to the proper pixel alignment convention). For more information on this method, see *example*.

Metrics

This section lists the image comparison metrics that will be used to assess the accuracy of the re-enlarged images. Each metric is associated with an aggregator and a best-to-worst ordering, as seen in the default settings.

```
# IMAGE DIFFERENCE METRICS AND AGGREGATORS
# Each metric must be associated with a data aggregation method.
# To add a metric, you must provide the following three items:
#     1. Error metric command, using the following replacement fields:
#         {0} - reference image
```

```
#          {1} - test image
#      2. Aggregator command, using the following replacement field:
#          {0} - list of error data to aggregate
#      3. Best to worst ordering, given as a 0 or 1:
#          0 - ascending
#          1 - descending
[Metrics]
srgb_1 = exquires compare srgb_1 {0} {1}, exquires aggregate l_1 {0}, 0
srgb_2 = exquires compare srgb_2 {0} {1}, exquires aggregate l_2 {0}, 0
srgb_4 = exquires compare srgb_4 {0} {1}, exquires aggregate l_4 {0}, 0
srgb_inf = exquires-compare srgb_inf {0} {1}, exquires-aggregate l_inf {0}, 0
cmc_1 = exquires-compare cmc_1 {0} {1}, exquires-aggregate l_1 {0}, 0
cmc_2 = exquires-compare cmc_2 {0} {1}, exquires-aggregate l_2 {0}, 0
cmc_4 = exquires-compare cmc_4 {0} {1}, exquires-aggregate l_4 {0}, 0
cmc_inf = exquires-compare cmc_inf {0} {1}, exquires-aggregate l_inf {0}, 0
xyz_1 = exquires-compare xyz_1 {0} {1}, exquires-aggregate l_1 {0}, 0
xyz_2 = exquires-compare xyz_2 {0} {1}, exquires-aggregate l_2 {0}, 0
xyz_4 = exquires compare xyz_4 {0} {1}, exquires aggregate l_4 {0}, 0
xyz_inf = exquires compare xyz_inf {0} {1}, exquires aggregate l_inf {0}, 0
blur_1 = exquires compare blur_1 {0} {1}, exquires aggregate l_1 {0}, 0
blur_2 = exquires compare blur_2 {0} {1}, exquires aggregate l_2 {0}, 0
blur_4 = exquires compare blur_4 {0} {1}, exquires aggregate l_4 {0}, 0
blur_inf = exquires compare blur_inf {0} {1}, exquires aggregate l_inf {0}, 0
mssim = exquires compare mssim {0} {1}, exquires aggregate l_1 {0}, 1
```

Note that these default metric definitions make use of *exquires-compare* (page 27) and *exquires-aggregate* (page 29). Also note that most of the metrics return an error measure, meaning that a lower result is better. MSSIM, on the other hand, is a similarity index, meaning that a higher result is better.

For more information on the default metrics, see `compare` (page 32).

For more information on the aggregation methods, see `aggregate` (page 31).

4.2.4 Computing the image comparison data

The basic syntax to run a project using *exquires-run* (page 24) is:

```
$ exquires-run
```

which will read the project file `project1.ini`, downsample the images by each ratio using each downsampler, re-enlarge the downsampled images using each upsampler, and compute the difference using each metric.

You can specify the project name using one of the following:

```
$ exquires-run -p my_project
$ exquires-run --proj my_project
```

where `my_project` is a name to identify your project.

By default, *exquires-run* (page 24) displays progress information. You can disable this output using one of the following:

```
$ exquires-run -s
$ exquires-run --silent
```

Warning: With large project files, this program can take an *extremely* long time to run. For slower machines, it is recommended to start with a small set of test images. You can add additional images later and call *exquires-update* (page 25) to compute the new data.

4.2.5 Updating the image comparison data

If you make changes to the project file after calling *exquires-run* (page 24), running it again will compute all data, including data for unchanged entries in the project file. To compute only the new data rather than recomputing the entire data set, use *exquires-update* (page 25), which supports the same options as *exquires-run* (page 24).

See *Computing the image comparison data* (page 16) for more information.

4.2.6 Generating a table of aggregate image comparison table

Once the image difference data has been computed, you can generate various aggregations of the data and either display it in the terminal or write it to a file.

The basic syntax to print aggregated data using *exquires-report* (page 25) is:

```
$ exquires-report
```

which will read a backup of the project file `project1.ini` that was created the last time *exquires-run* (page 24) or *exquires-update* (page 25) was called, select the appropriate values from the database, aggregate the data, and print the results in tabular format to standard output.

As with the other programs, you can specify the project name using one of the following:

```
$ exquires-report -p my_project
$ exquires-report --proj my_project
```

where `my_project` is a name to identify your project.

Normally, *exquires-report* (page 25) prints the data as a plaintext table. You may wish to include the results in a LaTeX document instead, which can be done using one of the following:

```
$ exquires-report -l
$ exquires-report --latex
```

Likewise, *exquires-report* (page 25) normally shows the aggregated data when it prints the table. You can instead show the Spearman (fractional) ranks for each upsampling method by using one of the following:

```
$ exquires-report -r
$ exquires-report --rank
```

Furthermore, you can instead merge the Spearman (fractional) ranks across all specified metrics by using one of the following:

```
$ exquires-report -m
$ exquires-report --merge
```

Whether you display aggregated data or ranks, by default the upsamplers in the printed table will be sorted from best-to-worst according to the first metric specified. If you wish to sort according to a different metric (including those that are not selected to be displayed), use one of the following:

```
$ exquires-report -s my_metric
$ exquires-report --sort my_metric
```

where `my_metric` is one of the metrics defined in the project file.

By default, *exquires-report* (page 25) prints the aggregated data to standard output. You can write the aggregated data to a file by using one of the following:

```
$ exquires-report -f my_file
$ exquires-report --file my_file
```

where `my_file` is the file you wish to write the data to.

When producing tables, *exquires-report* (page 25) will display 4 digits by default. You can select any number of digits between 1 and 16. For example, you can change the number of digits to to 6 using one of the following:

```
$ exquires-report -d 6
$ exquires-report --digits 6
```

There are three components that determine which database tables to aggregate across: images, ratios, and downsamplers. By default, the image comparison data is aggregated across all images, ratios, and downsampler. If you wish to aggregate over a subset of the database, use the following options.

You can specify the images to aggregate across by using one of the following:

```
$ exquires-report -I my_images
$ exquires-report --image my_images
```

where `my_images` is a list of images defined in the project file.

Note: The arguments passed to the *-I/--image* option support wildcard characters.

You can specify the downsamplers to aggregate across by using one of the following:

```
$ exquires-report -D my_downsamplers
$ exquires-report --down my_downsamplers
```

where `my_downsamplers` is a list of downsamplers defined in the project file.

Note: The arguments passed to the *-D/--down* option support wildcard characters.

You can specify the ratios to aggregate across by using one of the following:

```
$ exquires-report -R my_ratios
$ exquires-report --ratio my_ratios
```

where `my_ratios` is a list of images defined in the project file.

Note: The arguments passed to the *-R/--ratio* option support hyphenated ranges.

For example, to aggregate over the ratios 2 through 4 and 6, type:

```
$ exquires-report -R 2-4 6
```

Regardless of which images, downsamplers, and ratios the data is aggregated across, the default behaviour is to display data for each upsampler and metric, with each row representing an upsampler and each column representing a metric. If you wish to display only certain rows and columns, use the following options.

You can specify the metrics (columns) to display by using one of the following:

```
$ exquires-report -M my_metrics
$ exquires-report --metric my_metrics
```

where `my_metrics` is a list of metrics defined in the project file.

Note: The arguments passed to the *-M/--metric* option support wildcard characters.

For example, to only display data for the metrics prefixed with **xyz_**, type:

```
$ exquires-report -M xyz_*
```

You can specify the upsamplers (rows) to display by using one of the following:

```
$ exquires-report -U my_upsamplers
$ exquires-report --up my_upsamplers
```

where `my_upsamplers` is a list of upsamplers defined in the project file.

Note: The arguments passed to the *-U/--up* option support wildcard characters.

For example, to only display data for the upsamplers suffixed with **_srgb**, type:

```
$ exquires-report -U *_srgb
```

4.2.7 Generating a Spearman's rank cross-correlation matrix

In addition to using *exquires-report* (page 25) with the *-r/--rank* or *-m/--merge* options, which produce tables of Spearman (fractional) ranks, you can use *exquires-correlate* (page 26) to compute Spearman's rank cross-correlation matrices for several different groups.

The basic syntax to print a cross-correlation matrix using *exquires-correlate* (page 26) is:

```
$ exquires-correlate
```

which will read a backup of the project file `project1.ini` that was created the last time *exquires-run* (page 24) or *exquires-update* (page 25) was called, select the appropriate values from the database, aggregate the data, and print the cross-correlation matrix for all comparison metrics to standard output.

You can select which upsamplers to consider when computing the matrix by using the *-U/--up* option.

By default, the *-M/--metric* option is selected. You can select one of the following cross-correlation groups:

- *-I/--image*
- *-D/--down*
- *-R/--ratio*
- *-M/--metric*

As with the other programs, you can specify the project name using one of the following:

```
$ exquires-correlate -p my_project
$ exquires-correlate --proj my_project
```

Normally, *exquires-correlate* (page 26) prints the cross-correlation matrix as a plaintext table. You may wish to include the results in a LaTeX document instead, which can be done using one of the following:

```
$ exquires-correlate -l
$ exquires-correlate --latex
```

By default, *exquires-correlate* (page 26) prints the cross-correlation matrix to standard output. You can write the matrix to a file by using one of the following:

```
$ exquires-correlate -f my_file
$ exquires-correlate --file my_file
```

where `my_file` is the file you wish to write the data to.

When producing a matrix, *exquires-correlate* (page 26) will display 4 digits by default. You can select any number of digits between 1 and 16. For example, you can change the number of digits to to 6 using one of the following:

```
$ exquires-correlate -d 6
$ exquires-correlate --digits 6
```

By default, the order of the rows and columns of the correlation matrix corresponds to the order they were passed to *exquires-correlate* (page 26). It is often useful to sort the coefficients from best to worst based on a specific anchor row/column. You can specify the anchor using one of the following:

```
$ exquires-correlate -a my_anchor
$ exquires-correlate --anchor my_anchor
```

where `my_anchor` is the anchor you wish to use.

You can specify the upsamplers (rows) to consider in the computation by using one of the following:

```
$ exquires-correlate -U my_upsamplers
$ exquires-correlate --up my_upsamplers
```

where `my_upsamplers` is a list of upsamplers defined in the project file.

Note: The arguments passed to the *-U/--up* option support wildcard characters.

For example, to only consider data for the upsamplers suffixed with **_srgb**, type:

```
$ exquires-correlate -U *_srgb
```

4.2.8 Manually comparing images

The *exquires-run* (page 24) and *exquires-update* (page 25) programs compute data to be inserted into the database by calling *exquires-compare* (page 27) (see *The compare Module* (page 32)).

You can call *exquires-compare* (page 27) directly on any pair of images with the same dimensions by using:

```
$ exquires-compare my_metric my_image1 my_image2
```

where `my_image1` and `my_image2` are the images to compare and `my_metric` is one of the metrics described in *The compare Module* (page 32).

By default, *exquires-compare* (page 27) expects images with 16 bits per sample: each value is between 0 and 65535. You can change the maximum value from 65535 to anything you like. For example, to support images with 8 bits per sample (values between 0 and 255), type one of the following:

```
$ exquires-compare my_metric my_image1 my_image2 -m 255
$ exquires-compare my_metric my_image1 my_image2 --maxval 255
```

4.2.9 Manually aggregating data

The *exquires-report* (page 25) program aggregates the image comparison data before printing it to standard output or writing it to a file by calling *exquires-aggregate* (page 29).

You can call *exquires-aggregate* (page 29) directly on any list of numbers by using:

```
$ exquires-aggregate my_method my_numbers
```

where `my_numbers` is a list of numbers separated by spaces and `my_method` is one of the aggregation methods described in *The aggregate Module* (page 31).

For example, to return the average of a list of numbers, type:

```
$ exquires-aggregate l_1 1.2 2.4 3.6 4.8
3.000000000000000
```

and to find the maximum, type:

```
$ exquires-aggregate l_inf 1.2 2.4 3.6 4.8
4.800000000000000
```

CHAPTER

FIVE

PROGRAMS

5.1 exquires-new

Syntax:

```
exquires-new [-h] [-v] [-p PROJECT] [-I IMAGE [IMAGE ...]]
```

Description:

Generate a new project file to use with *exquires-run* (page 24).

The project file is used to specify the following components of the suite:

- Images (sRGB TIFF | 16 bits/sample (48/pixel) | 840x840 pixels)
- Downsamplers
- Resampling Ratios
- Upsamplers
- Difference Metrics

For the specified project name and list of images, a default project file will be created with the name `PROJECT.ini`, where `PROJECT` is a name specified using the *-p*:option:*–proj* option. If a name is not specified, the default name is `project1`.

Use the *-I*:option:*–image* option to provide a list of images to include in the project file. If no images are specified, a default image (`wave.tif`) is included in the project file.

Manually edit this file to customize your project.

Optional Arguments:

SHORT FLAG	LONG FLAG	ARGUMENTS	DESCRIPTION
`-h`	`--help`		show this help message and exit
`-v`	`--version`		show program's version number and exit
`-p`	`--proj`	*PROJECT*	name of the project (default: *project1*)
`-I`	`--image`	*IMAGE [IMAGE ...]*	the test images to use (default: `wave.tif`)

For additional usage instructions, see *Obtaining suitable test images* (page 12), *Creating a new project file* (page 12), and *Customizing the project file* (page 13).

For technical information, see `new` (page 44).

5.2 exquires-run

Syntax:

```
exquires-run [-h] [-v] [-s] [-p PROJECT]
```

Description:

Compute error data for the entries in the specified project file.

The project file is read to determine which images, downsamplers, ratios, upsamplers, and metrics to use. If a database file already exists for this project, it will be backed up and a new one will be created.

Each image will be downsampled by each of the ratios using each of the downsamplers. The downsampled images will then be upsampled back to their original size (840x840) using each of the upsamplers. The upsampled images will be compared to the original images using each of the metrics and the results will be stored in the database file.

If you make changes to the project file and wish to only compute data for these changes rather than recomputing everything, use *exquires-update* (page 25).

To view aggregated error data, use *exquires-report* (page 25).

Optional Arguments:

SHORT FLAG	LONG FLAG	ARGUMENTS	DESCRIPTION
`-h`	`--help`		show this help message and exit
`-v`	`--version`		show program's version number and exit
`-s`	`--silent`		do not display progress information
`-p`	`--proj`	*PROJECT*	name of the project (default: *project1*)

For additional usage instructions, see *Computing the image comparison data* (page 16).

For technical information, see `run` (page 63).

5.3 exquires-update

Syntax:

```
exquires-update [-h] [-v] [-s] [-p PROJECT]
```

Description:

Compute new error data for changes to the user-specified project file.

The project file is inspected to determine which changes have been made. Items that have been removed will result in entries being removed from the database. Items that have been changed or added will result in new data being computed and added to the database file. If no changes have been made to the project file, the database will not be updated.

If you wish to recompute all data based on your project file rather than simply updating it with the changes, use *exquires-run* (page 24).

To view aggregated error data, use *exquires-report* (page 25).

Optional Arguments:

SHORT FLAG	LONG FLAG	ARGUMENTS	DESCRIPTION
-h	*--help*		show this help message and exit
-v	*--version*		show program's version number and exit
-s	*--silent*		do not display progress information
-p	*--proj*	*PROJECT*	name of the project (default: *project1*)

For additional usage instructions, see *Updating the image comparison data* (page 17).

For technical information, see `update` (page 67).

5.4 exquires-report

Syntax:

```
exquires-report [-h] [-v] [-l] [-r | -m] [-p PROJECT] [-f FILE]
                [-d DIGITS] [-s METRIC] [-U METHOD [METHOD ...]]
                [-I IMAGE [IMAGE ...]] [-D METHOD [METHOD ...]]
                [-R RATIO [RATIO ...]] [-M METRIC [METRIC ...]]
```

Description:

Print a formatted table of aggregate image difference data.

Each database table in the current project contains data for a single image, downsampler, and ratio. Each row represents an upsampler and each column represents a difference metric. By default, the data across all rows and columns of all tables is aggregated. Use the appropriate option flags to aggregate across a subset of the database.

Optional Arguments:

SHORT FLAG	LONG FLAG	ARGUMENTS	DESCRIPTION
`-h`	`--help`		show this help message and exit
`-v`	`--version`		show program's version number and exit
`-l`	`--latex`		print a LaTeX formatted table
`-r`	`--rank`		print Spearman (fractional) ranks
`-m`	`--merge`		print merged Spearman ranks
`-p`	`--proj`	*PROJECT*	name of the project (default: *project1*)
`-f`	`--file`	*FILE*	output to file (default: *sys.stdout*)
`-d`	`--digits`	*DIGITS*	total number of digits (default: *4*)
`-s`	`--sort`	*METRIC*	sort using this metric (default: *first*)
`-U`	`--up`	*METHOD [METHOD ...]*	upsamplers to consider (default: *all*)
`-I`	`--image`	*IMAGE [IMAGE ...]*	images to consider (default: *all*)
`-D`	`--down`	*METHOD [METHOD ...]*	downsamplers to consider (default: *all*)
`-R`	`--ratio`	*RATIO [RATIO ...]*	ratios to consider (default: *all*)
`-M`	`--metric`	*METRIC [METRIC ...]*	metrics to consider (default: *all*)

Features:

- `-R`/`--ratio` supports hyphenated ranges (ex. '2-4 6' gives '2 3 4 6')
- `-U`/`--up`, `-I`/`--image`, `-D`/`--down` and `-M`/`--metric` support wildcards

For additional usage instructions, see *Generating a table of aggregate image comparison table* (page 17).

For technical information, see `report` (page 62).

5.5 exquires-correlate

Syntax:

```
exquires-correlate [-h] [-v] [-l] [-p PROJECT] [-f FILE] [-d DIGITS]
                   [-U METHOD [METHOD ...]] [-I IMAGE [IMAGE ...] | -D
                   METHOD [METHOD ...] | -R RATIO [RATIO ...] | -M
                   METRIC [METRIC ...]]
```

Description:

Produce a Spearman's rank cross-correlation matrix for the specified group.

By default, the *-M/--metric* option is selected. You can select one of the following cross-correlation groups:

- *-I/--image*
- *-D/--down*
- *-R/--ratio*
- *-M/--metric*

You can also select which upsamplers to consider when computing the matrix by using the *-U/--up* option.

Optional Arguments:

SHORT FLAG	LONG FLAG	ARGUMENTS	DESCRIPTION
-h	*--help*		show this help message and exit
-v	*--version*		show program's version number and exit
-l	*--latex*		print a LaTeX formatted table
-p	*--proj*	*PROJECT*	name of the project (default: *project1*)
-f	*--file*	*FILE*	output to file (default: *sys.stdout*)
-d	*--digits*	*DIGITS*	total number of digits (default: *4*)
-a	*--anchor*	*ANCHOR*	sort using this anchor (default: *none*)
-U	*--up*	*METHOD [METHOD ...]*	upsamplers to consider (default: *all*)
-I	*--image*	*IMAGE [IMAGE ...]*	images to consider (default: *all*)
-D	*--down*	*METHOD [METHOD ...]*	downsamplers to consider (default: *all*)
-R	*--ratio*	*RATIO [RATIO ...]*	ratios to consider (default: *all*)
-M	*--metric*	*METRIC [METRIC ...]*	metrics to consider (default: *all*)

For additional usage instructions, see *Generating a Spearman's rank cross-correlation matrix* (page 20).

For technical information, see `correlate` (page 39).

5.6 exquires-compare

Syntax:

```
exquires-compare [-h] [-v] [-m MAX_LEVEL] METRIC IMAGE_1 IMAGE_2
```

Description:

Print the result of calling a difference metric on two image files.

Difference Metrics:

NAME	DESCRIPTION
`srgb_1`	ℓ_1 norm in sRGB colour space
`srgb_2`	ℓ_2 norm in sRGB colour space
`srgb_4`	ℓ_4 norm in sRGB colour space
`srgb_inf`	ℓ_∞ norm in sRGB colour space
`cmc_1`	ℓ_1 norm in CMC(1:1) colour space
`cmc_2`	ℓ_2 norm in CMC(1:1) colour space
`cmc_4`	ℓ_4 norm in CMC(1:1) colour space
`cmc_inf`	ℓ_∞ norm in CMC(1:1) colour space
`xyz_1`	ℓ_1 norm in XYZ colour space
`xyz_2`	ℓ_2 norm in XYZ colour space
`xyz_4`	ℓ_4 norm in XYZ colour space
`xyz_inf`	ℓ_∞ norm in XYZ colour space
`blur_1`	MSSIM-inspired ℓ_1 norm
`blur_2`	MSSIM-inspired ℓ_2 norm
`blur_4`	MSSIM-inspired ℓ_4 norm
`blur_inf`	MSSIM-inspired ℓ_∞ norm
`mssim`	Mean Structural Similarity Index (MSSIM)

Positional Arguments:

ARGUMENT	DESCRIPTION
METRIC	the difference metric to use
IMAGE_1	the first image to compare
IMAGE_2	the second image to compare

Optional Arguments:

SHORT FLAG	LONG FLAG	ARGUMENTS	DESCRIPTION
`-h`	`--help`		show this help message and exit
`-v`	`--version`		show program's version number and exit
`-m`	`--maxval`	*MAX_LEVEL*	the maximum pixel value (default: *65535*)

For additional usage instructions, see *Manually comparing images* (page 22).

For technical information, see `compare` (page 32).

5.7 exquires-aggregate

Syntax:

```
exquires-aggregate [-h] [-v] METHOD NUM [NUM ...]
```

Description:

Aggregate a list of values using the selected method.

Aggregators:

NAME	DESCRIPTION
l_1	return the average
l_2	average the squares and return the square root
l_4	average the quads and return the fourth root
l_inf	return the maximum

Positional Arguments:

ARGUMENT	DESCRIPTION
METHOD	the type of aggregation to use
NUM	number to include in aggregation

Optional Arguments:

SHORT FLAG	LONG FLAG	DESCRIPTION
-h	*--help*	show the help message and exit
-v	*--version*	show the program's version number and exit

For additional usage instructions, see *Manually comparing images* (page 22).

For technical information, see `compare` (page 32).

CHAPTER

SIX

MODULES & CLASSES

6.1 The `aggregate` Module

Aggregate a list of numbers using the specified method.

Aggregators:

NAME	DESCRIPTION
l_1	return the average
l_2	average the squares and return the square root
l_4	average the quads and return the fourth root
l_inf	return the maximum

`aggregate.`**`main`**`()`

Run *exquires-aggregate* (page 29).

6.1.1 The `Aggregate` Class

class `aggregate.`**`Aggregate`**`(`*values*`)`

Bases: `object`

This class provide various ways of aggregating error data.

Parameters **values** (*list of numbers*) – numbers to aggregate

`l_1``()`

Return the average.

Returns the average

Return type *float*

`l_2()`
Average the squares and return the square root.

Returns the square root of the average of the squares

Return type *float*

`l_4()`
Average the quads and return the fourth root.

Returns the fourth root of the average of the quads

Return type *float*

`l_inf()`
Return the maximum.

Returns the maximum

Return type *float*

6.2 The `compare` Module

Print the result of calling a difference metric on two image files.

Difference Metrics:

NAME	DESCRIPTION
srgb_1	ℓ_1 norm in sRGB colour space
srgb_2	ℓ_2 norm in sRGB colour space
srgb_4	ℓ_4 norm in sRGB colour space
srgb_inf	ℓ_∞ norm in sRGB colour space
cmc_1	ℓ_1 norm using the CMC(1:1) colour difference
cmc_2	ℓ_2 norm using the CMC(1:1) colour difference
cmc_4	ℓ_4 norm using the CMC(1:1) colour difference
cmc_inf	ℓ_∞ norm using the CMC(1:1) colour difference
xyz_1	ℓ_1 norm in XYZ colour space
xyz_2	ℓ_2 norm in XYZ colour space
xyz_4	ℓ_4 norm in XYZ colour space
xyz_inf	ℓ_∞ norm in XYZ colour space
blur_1	MSSIM-inspired ℓ_1 norm
blur_2	MSSIM-inspired ℓ_2 norm
blur_4	MSSIM-inspired ℓ_4 norm
blur_inf	MSSIM-inspired ℓ_∞ norm
mssim	Mean Structural Similarity Index (MSSIM)

compare.**_get_blurlist**()

Private method to return a Gaussian blur mask.

Note: This is a private function called by `blur_1()` (page 34), `blur_2()` (page 34), `blur_4()` (page 35), `blur_inf()` (page 35), and `mssim()` (page 36).

---

compare.**main**()

Run *exquires-compare* (page 27).

6.2.1 The `Metrics` Class

class compare.**Metrics**(*image1*, *image2*, *maxval=65535*)

Bases: `object`

This class contains error metrics to be used on sRGB images.

The ℓ_1, ℓ_2, ℓ_4, and ℓ_∞ metrics are normalized by L, the largest possible pixel value of the input images (the lowest is assumed to be 0). The range of output for these metrics is [0, 100].

The MSSIM metric produces output in the range [-1, 1], but it is unlikely that a negative value will be produced, as the image features must differ greatly. For instance, a pure white image compared with a pure black image produces a result slightly greater than 0.

The CMC and XYZ errors can be slightly outside the range [0, 100], but this will not occur for most image pairs.

Note: By default, a `Metrics` (page 33) object is configured to operate on 16-bit images.

Parameters

- **image1** (*path*) – first image to compare (reference image)
- **image2** (*path*) – second image to compare (test image)
- **L** (*integer*) – highest possible pixel value (default=65535)

blur_1()

Compute MSSIM-inspired ℓ_1 error.

This method performs the same greyscale conversion, Gaussian blur, and cropping as MSSIM, but returns the ℓ_1 error of the cropped image.

See (6.2) for details on how the blurred images are compared.

Note: The images are converted to grayscale before applying Gaussian blur. The grayscale conversion is equivalent to taking the Y channel in YIQ colour space.

Returns MSSIM-inspired ℓ_1 error

Return type *float*

blur_2()

Compute MSSIM-inspired ℓ_2 error.

This method performs the same greyscale conversion, Gaussian blur, and cropping as MSSIM, but returns the ℓ_2 error of the cropped image.

See (6.3) for details on how the blurred images are compared.

Note: The images are converted to grayscale before applying Gaussian blur. The grayscale conversion is equivalent to taking the Y channel in YIQ colour space.

Returns MSSIM-inspired ℓ_2 error

Return type *float*

blur_4()
Compute MSSIM-inspired ℓ_4 error.

This method performs the same greyscale conversion, Gaussian blur, and cropping as MSSIM, but returns the ℓ_4 error of the cropped image.

See (6.4) for details on how the blurred images are compared.

Note: The images are converted to grayscale before applying Gaussian blur. The grayscale conversion is equivalent to taking the Y channel in YIQ colour space.

Returns MSSIM-inspired ℓ_4 error

Return type *float*

blur_inf()
Compute MSSIM-inspired ℓ_∞ error.

This method performs the same greyscale conversion, Gaussian blur, and cropping as MSSIM, but returns the ℓ_∞ error of the cropped image.

See (6.5) for details on how the blurred images are compared.

Note: The images are converted to grayscale before applying Gaussian blur. The grayscale conversion is equivalent to taking the Y channel in YIQ colour space.

Returns MSSIM-inspired ℓ_∞ error

Return type *float*

cmc_1()
Compute ℓ_1 error in Uniform Colour Space (UCS).

This method imports the images into Lab colour space, then calculates delta-E CMC(1:1) and returns the average.

See (6.2) for details on how the standard ℓ_1 norm is computed.

Returns ℓ_1 error in Uniform Colour Space (UCS)

Return type *float*

cmc_2 ()

Compute ℓ_2 error in Uniform Colour Space (UCS).

This method imports the images into Lab colour space, then calculates delta-E CMC(1:1) and returns the ℓ_2 norm.

See (6.3) for details on how the standard ℓ_2 norm is computed.

Returns ℓ_2 error in Uniform Colour Space (UCS)

Return type *float*

cmc_4 ()

Compute ℓ_4 error in Uniform Colour Space (UCS).

This method imports the images into Lab colour space, then calculates delta-E CMC(1:1) and returns the ℓ_4 norm.

See (6.4) for details on how the standard ℓ_4 norm is computed.

Returns ℓ_4 error in Uniform Colour Space (UCS)

Return type *float*

cmc_inf ()

Compute ℓ_∞ error in Uniform Colour Space (UCS).

This method imports the images into Lab colour space, then calculates delta-E CMC(1:1) and returns the ℓ_∞ norm.

See (6.5) for details on how the standard ℓ_∞ norm is computed.

Returns ℓ_∞ error in Uniform Colour Space (UCS)

Return type *float*

mssim ()

Compute the Mean Structural Similarity Index (MSSIM).

The equation for SSIM is

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{6.1}$$

where μ_x and μ_y are the sample means, σ_x and σ_y are the standard deviations, and σ_{xy} is the correlation coefficient between images x and y.

Once the SSIM map is computed, the border is trimmed by 5 pixels and the mean is returned.

This version is slightly more efficient than the method proposed by Wang et. al. because it reduces the number of Gaussian blurs from 5 to 4.

Note: The images are converted to grayscale before applying Gaussian blur. The grayscale conversion is equivalent to taking the Y channel in YIQ colour space.

Returns mean SSIM

Return type *float*

srgb_1 ()

Compute ℓ_1 error in sRGB colour space.

The equation for the ℓ_1 error, aka Average Absolute Error (AAE), is

$$\ell_1(x, y) = \frac{1}{N} \sum_{i=1}^{N} |x_i - y_i| \tag{6.2}$$

where x and y are the images to compare, each consisting of N pixels.

Returns ℓ_1 error

Return type *float*

srgb_2 ()

Compute ℓ_2 error in sRGB colour space.

The equation for the ℓ_2 error, aka Root Mean Squared Error (RMSE), is

$$\ell_2(x, y) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (x_i - y_i)^2} \tag{6.3}$$

where x and y are the images to compare, each consisting of N pixels.

Returns ℓ_2 error

Return type *float*

srgb_4 ()

Compute ℓ_4 error in sRGB colour space.

The equation for the ℓ_1 error is

$$\ell_4(x, y) = \sqrt[4]{\frac{1}{N} \sum_{i=1}^{N} (x_i - y_i)^4} \tag{6.4}$$

where x and y are the images to compare, each consisting of N pixels.

Returns ℓ_4 error

Return type *float*

srgb_inf ()
Compute ℓ_∞ error in sRGB colour space.

The equation for the ℓ_∞ error, aka Maximum Absolute Error (MAE), is

$$\ell_\infty(x, y) = \max_{1 \le i \le N} |x_i - y_i| \tag{6.5}$$

where x and y are the images to compare, each consisting of N pixels.

Returns ℓ_∞ error

Return type *float*

xyz_1 ()
Compute ℓ_1 error in XYZ Colour Space.

This method imports the images into XYZ colour space, then calculates the ℓ_1 error.

See (6.2) for details on how the standard ℓ_1 norm is computed.

Returns ℓ_1 error in XYZ Colour Space

Return type *float*

xyz_2 ()
Compute ℓ_2 error in XYZ Colour Space.

This method imports the images into XYZ colour space, then calculates the ℓ_2 error.

See (6.3) for details on how the standard ℓ_2 norm is computed.

Returns ℓ_2 error in XYZ Colour Space

Return type *float*

xyz_4 ()
Compute ℓ_4 error in XYZ Colour Space.

This method imports the images into XYZ colour space, then calculates the ℓ_4 error.

See (6.4) for details on how the standard ℓ_4 norm is computed.

Returns ℓ_4 error in XYZ Colour Space

Return type *float*

xyz_inf()
Compute ℓ_∞ error in XYZ Colour Space.

This method imports the images into XYZ colour space, then calculates the ℓ_∞ error.

See (6.5) for details on how the standard ℓ_∞ norm is computed.

Returns ℓ_∞ error in XYZ Colour Space

Return type *float*

6.3 The `correlate` Module

Produce a Spearman's rank cross-correlation matrix for the specified group.

By default, the `-M/--metric` option is selected. You can select one of the following cross-correlation groups:

- `-I/--image`
- `-D/--down`
- `-R/--ratio`
- `-M/--metric`

You can also select which upsamplers to consider when computing the matrix by using the `-U/--up` option.

`correlate.`**_get_group_and_ranks** (*args*)
Return the correlation group and ranks.

Note: This is a private function called by `_print_matrix()` (page 40).

Parameters

- **args** (`argparse.Namespace`) – arguments
- **args.dbase_file** (*path*) – database file
- **args.image** (*list of strings*) – selected image names
- **args.down** (*list of strings*) – selected downsampler names
- **args.ratio** (*list of strings*) – selected ratios
- **args.up** (*list of strings*) – selected upsampler names

- **args.metric** (*list of strings*) – selected metric names
- **args.metrics_d** (*dict*) – all metric names
- **args.file** (*path*) – output file
- **args.digits** (*integer*) – number of digits to print
- **args.latex** (*boolean*) – *True* if printing a LaTeX-formatted table
- **args.key** (*string*) – key for the correlation group

Returns the group and ranks

Return type *string, list of lists*

`correlate._print_matrix` (*args*)

Print a cross-correlation matrix from aggregate image comparison data.

Note: This is a private function called by `main()` (page 41).

Parameters

- **args** (`argparse.Namespace`) – arguments
- **args.dbase_file** (*path*) – database file
- **args.image** (*list of strings*) – selected image names
- **args.down** (*list of strings*) – selected downsampler names
- **args.ratio** (*list of strings*) – selected ratios
- **args.up** (*list of strings*) – selected upsampler names
- **args.metric** (*list of strings*) – selected metric names
- **args.metrics_d** (*dict*) – all metric names
- **args.file** (*path*) – output file
- **args.digits** (*integer*) – number of digits to print
- **args.latex** (*boolean*) – *True* if printing a LaTeX-formatted table
- **args.key** (*string*) – key for the correlation group
- **args.anchor** (*string*) – row/column to order the matrix by

correlate.**main**()
Run *exquires-correlate* (page 26).

Parse the command-line arguments and print the cross-correlation matrix.

6.4 The database Module

Provides an interface to the sqlite3 image error database.

6.4.1 The Database Class

class database.**Database**(*dbasefile*)
This class provides an interface to the sqlite3 image error database.

The database stores error data computed by *exquires-run* (page 24) and *exquires-update* (page 25). This data is retrieved and used to compute the output given by *exquires-report* (page 25) and *exquires-correlate* (page 26).

Parameters **dbasefile** (*path*) – database file to connect to

_Database__create_table(*name*, *metrics*)
Private method used to create a new database table.

Note: This is a private method called by add_table() (page 41) and backup_table() (page 42).

Parameters

- **name** (*string*) – name of the table to create
- **metrics** (*list of strings*) – error metrics to compute (the table columns)

add_table(*image*, *downsampler*, *ratio*, *metrics*)
Add a new table to the database.

Each table is defined in the following way:

1. image, downsampler, and ratio define the table name
2. metrics define the columns of the table

To keep track of each table in terms of the image, downsampler, and ratio that defines it, an entry is created in the TABLEDATA table.

Parameters

- **image** (*string*) – name of the image
- **downsampler** (*string*) – name of the downsampler
- **ratio** (*string*) – resampling ratio
- **metrics** (*list of strings*) – names of the metrics

Returns the table name

Return type *string*

backup_table (*name, metrics*)
Backup an existing image error table.

Parameters

- **name** (*string*) – name of the table to backup
- **metrics** (*list of strings*) – error metrics (columns) to backup

Returns name of the backup table

Return type *string*

close ()
Close the connection to the database.

delete (*table, upsampler*)
Delete a row from the table.

Parameters

- **table** (*string*) – name of the table to delete from
- **upsampler** (*string*) – upsampler (row) to remove from the table

drop_backup (*name*)
Drop a backup table once it is no longer needed.

Parameters **name** (*string*) – name of the backup table to drop

drop_tables (*images, downsamplers, ratios*)
Drop database tables.

All tables defined by any of the images, downsamplers, or ratios are dropped. The TABLEDATA table is updated to reflect these changes.

Parameters

- **images** (*list of strings*) – names of the images
- **downsamplers** (*list of strings*) – names of the downsamplers
- **ratios** (*list of strings*) – resampling ratios

get_error_data (*table*, *upsampler*, *metrics_str*)
Return a filtered row of error data.

For the upsampler row in the table, return a dictionary containing only the error data for the specified metrics.

Parameters

- **table** (*string*) – name of the table to query
- **upsampler** (*string*) – name of the upsampler (row) to acquire data from
- **metrics_str** (*string*) – metrics to return error data for (comma-separated)

Returns the filtered row of error data

Return type *dict*

get_tables (*args*)
Return table names for these images, downsamplers, and ratios.

Parameters

- **args** (`argparse.Namespace`) – arguments
- **args.image** (*list of strings*) – names of images
- **args.down** (*list of strings*) – names of downsamplers
- **args.ratio** (*list of strings*) – resampling ratios

Returns names of the tables

Return type *list of strings*

insert (*table*, *row*)
Insert a single row into the table, or update if it exists.

Parameters

- **table** (*string*) – name of the table
- **row** (*dict*) – row data to insert

sql_do (*sql*, *params=()*)
Perform an operation on the database and commit the changes.

Parameters

- **sql** (*string*) – SQL statement to execute and commit
- **params** (*list of values*) – values to fill wildcards in the SQL statement

sql_fetchall (*sql*, *params=()*)
Fetch all rows for the specified SQL query.

Parameters

- **sql** (*string*) – SQL query to execute
- **params** (*list of values*) – values to fill wildcards in the SQL statement

Returns rows specified by the SQL query

Return type *list of dicts*

6.5 The new Module

Generate a new project file to use with *exquires-run* (page 24).

The project file is used to specify the following components of the suite:

- Images (sRGB TIFF | 16 bits/sample (48/pixel) | 840x840 pixels)
- Downsamplers
- Resampling Ratios
- Upsamplers
- Difference Metrics

For the specified project name and list of images, a default project file will be created with the name `PROJECT.ini`, where `PROJECT` is a name specified using the *-p*:option:*–proj* option. If a name is not specified, the default name is `project1`.

Use the *-I*:option:*–image* option to provide a list of images to include in the project file. If no images are specified, a default image (`wave.tif`) is included in the project file.

Manually edit this file to customize your project.

new.**_add_default_downsamplers**(*ini*)
Add the default downsamplers to the specified `.ini` file.

Note: This is a private function called by `main()` (page 48).

Parameters **ini** (`configobj.ConfigObj`) – the `.ini` file to modify

new.**_add_default_images**(*ini*, *image*)
Add the default images to the specified `.ini` file.

Note: This is a private function called by `main()` (page 48).

Parameters **ini** (`configobj.ConfigObj`) – the `.ini` file to modify

new.**_add_default_metrics**(*ini*)
Add the default metrics to the specified `.ini` file.

Note: This is a private function called by `main()` (page 48).

Parameters **ini** (`configobj.ConfigObj`) – the `.ini` file to modify

new.**_add_default_ratios**(*ini*)
Add the default ratios to the specified `.ini` file.

Note: This is a private function called by `main()` (page 48).

Parameters **ini** (`configobj.ConfigObj`) – the `.ini` file to modify

new.**_add_default_upsamplers**(*ini*)
Add the default upsamplers to the specified `.ini` file.

Note: This is a private function called by `main()` (page 48).

Parameters **ini** (`configobj.ConfigObj`) – the `.ini` file to modify

new.**_magick** (*method*, ***kwargs*)

Return an ImageMagick resize command as a string.

Blur and Kaiser beta values are passed as strings to avoid truncation.

Note: This is a private function called by `_add_default_downsamplers()` (page 44), `_std_int_lin_tensor_mtds_1()` (page 47), `_std_int_lin_tensor_mtds_2()` (page 47), `_novel_int_lin_flt_mtds()` (page 47), `_std_nonint_lin_tensor_mtds()` (page 48), `_std_int_ewa_lin_flt_mtds()` (page 47), `_std_nonint_ewa_lin_flt_mtds()` (page 48), and `_novel_nonint_ewa_lin_flt_mtds()` (page 47).

Parameters

- **method** (*string*) – method to use with *-resize* or *-distort Resize*
- **lin** (*boolean*) – *True* if using a linear method
- **dist** (*boolean*) – *True* if using a *-distort Resize* method
- **lobes** (*integer*) – number of lobes
- **blur** (*string*) – blur value
- **beta** (*string*) – beta value for Kaiser method

Returns the ImageMagick command

Return type *string*

new.**_metric** (*method*, *aggregator*, *sort*)

Return 3-element list defining a metric, an aggregator and a sort order.

Note: This is a private function called by `_add_default_metrics()` (page 45).

Parameters

- **method** (*string*) – image comparison metric (see *exquires-compare* (page 27))
- **aggregator** (*string*) – data aggregator (see *exquires-aggregate* (page 29))
- **sort** (*integer*) – best-to-worst order (*0*: ascending, *1*: descending)

Returns metric, aggregator, and sort order

Return type *list*

new.**_novel_int_lin_flt_mtds** (*ini_ups*)
Add the novel interpolatory linear filtering methods.

Note: This is a private function called by `_add_default_upsamplers()` (page 45).

Parameters **ini_ups** (*dict*) – upsamplers for the specified `.ini` file

new.**_novel_nonint_ewa_lin_flt_mtds** (*ini_ups*)
Add the novel non-interpolatory EWA linear filtering methods.

Note: This is a private function called by `_add_default_upsamplers()` (page 45).

Parameters **ini_ups** (*dict*) – upsamplers for the specified `.ini` file

new.**_std_int_ewa_lin_flt_mtds** (*ini_ups*)
Add the standard interpolatory EWA linear filtering methods.

Note: This is a private function called by `_add_default_upsamplers()` (page 45).

Parameters **ini_ups** (*dict*) – upsamplers for the specified `.ini` file

new.**_std_int_lin_tensor_mtds_1** (*ini_ups*)
Add the 1st part of the standard interpolatory linear tensor methods.

Note: This is a private function called by `_add_default_upsamplers()` (page 45).

Parameters **ini_ups** (*dict*) – upsamplers for the specified `.ini` file

new.**_std_int_lin_tensor_mtds_2** (*ini_ups*)
Add the 2nd part of the standard interpolatory linear tensor methods.

Note: This is a private function called by `_add_default_upsamplers()` (page 45).

Parameters **ini_ups** (*dict*) – upsamplers for the specified `.ini` file

new.**_std_nonint_ewa_lin_flt_mtds** (*ini_ups*)
Add the standard non-interpolatory EWA linear filtering methods.

Note: This is a private function called by `_add_default_upsamplers()` (page 45).

Parameters **ini_ups** (*dict*) – upsamplers for the specified `.ini` file

new.**_std_nonint_lin_tensor_mtds** (*ini_ups*)
Add the standard non-interpolatory linear tensor methods.

Note: This is a private function called by `_add_default_upsamplers()` (page 45).

Parameters **ini_ups** (*dict*) – upsamplers for the specified `.ini` file

new.**main**()
Run *exquires-new* (page 23).

Create a project file to use with *exquires-run* (page 24) and *exquires-update* (page 25).

6.6 The `operations` Module

A collection of classes used to compute image difference data.

The hierarchy of classes is as follows:

- `Operations` (page 49) encapsulate a list of `Images` (page 49)
- `Images` (page 49) encapsulate a *dict* of images and a list of `Downsamplers` (page 50)
- `Downsamplers` (page 50) encapsulate a *dict* of downsamplers and a list of `Ratios` (page 51)
- `Ratios` (page 51) encapsulate a *dict* of ratios and a list `Images` (page 49)
- `Images` (page 49) encapsulate a *dict* of images and a *dict* of metrics

These classes work together to downsample the master images, upsample the downsampled images, and compare the upsampled images to the master images. To perform the operations, call `Operations.compute()` (page 49).

6.6.1 The `Operations` Class

class `operations.`**`Operations`**(*images*)

Bases: `object`

A collection of Image objects to compute data with.

This class is responsible for calling all operations defined in the specified project file when using *exquires-run* (page 24) or *exquires-update* (page 25).

Parameters **images** (list of `Images` (page 49)) – images to downsample

`compute`(*args*, *old=None*)

Perform all operations.

Parameters

- **args** (`argparse.Namespace`) – arguments
- **args.prog** (*string*) – name of the calling program
- **args.dbase_file** (*path*) – database file
- **args.proj** (*string*) – name of the current project
- **args.silent** (*boolean*) – *True* if using silent mode
- **args.met_same** (*dict*) – unchanged metrics
- **args.metrics** (*dict*) – current metrics
- **args.config_file** (*path*) – current configuration file
- **args.config_bak** (*path*) – previous configuration file
- **old** (`argparse.Namespace`) – old configuration entries to be removed

6.6.2 The `Images` Class

class `operations.`**`Images`**(*images*, *downsamplers*, *same=False*)

Bases: `object`

This class calls operations for a particular set of images.

Parameters

- **images** (*dict*) – images to downsample
- **downsamplers** (list of `Downsamplers` (page 50)) – downsamplers to use
- **same** (*boolean*) – *True* if using unchanged images

compute (*args*)
Perform all operations for this set of images.

Parameters

- **args** (`argparse.Namespace`) – arguments
- **args.dbase_file** (*path*) – database file
- **args.dbase** (`database.Database` (page 41)) – connected database
- **args.proj** (*string*) – name of the current project
- **args.silent** (*boolean*) – *True* if using silent mode
- **args.met_same** (*dict*) – unchanged metrics
- **args.metrics** (*dict*) – current metrics
- **args.do_op** (*function*) – updates the displayed progress

6.6.3 The `Downsamplers` Class

class `operations.`**Downsamplers** (*downsamplers, ratios, same=False*)
Bases: `object`

This class calls operations for a particular set of downsamplers.

Parameters

- **downsamplers** (*dict*) – downsamplers to use
- **ratios** (list of `Ratios` (page 51)) – ratios to downsample by
- **same** (*boolean*) – *True* if using unchanged downsamplers

compute (*args, same*)
Perform all operations for this set of downsamplers.

Parameters

- **args** (`argparse.Namespace`) – arguments
- **args.dbase_file** (*path*) – database file
- **args.dbase** (`database.Database` (page 41)) – connected database
- **args.proj** (*string*) – name of the current project
- **args.silent** (*boolean*) – *True* if using silent mode
- **args.met_same** (*dict*) – unchanged metrics
- **args.metrics** (*dict*) – current metrics
- **args.do_op** (*function*) – updates the displayed progress
- **args.image** (*string*) – name of the image
- **args.image_dir** (*path*) – directory to store results for this image
- **args.master** (*path*) – master image to downsample
- **same** (*boolean*) – *True* if possibly accessing an existing table

6.6.4 The `Ratios` Class

class `operations.`**`Ratios`**(*ratios*, *upsamplers*, *same=False*)

Bases: `object`

This class calls operations for a particular set of ratios.

Parameters

- **ratios** (*dict*) – ratios to downsample by
- **upsamplers** (list of `Upsamplers` (page 52)) – upsamplers to use
- **same** (*boolean*) – *True* if using unchanged ratios

`compute`(*args*, *downsamplers*, *same*)

Perform all operations for this set of ratios.

Parameters

- **args** (`argparse.Namespace`) – arguments
- **args.dbase_file** (*path*) – database file

- **args.dbase** (`database.Database` (page 41)) – connected database
- **args.proj** (*string*) – name of the current project
- **args.silent** (*boolean*) – *True* if using silent mode
- **args.met_same** (*dict*) – unchanged metrics
- **args.metrics** (*dict*) – current metrics
- **args.do_op** (*function*) – updates the displayed progress
- **args.image** (*string*) – name of the image
- **args.image_dir** (*path*) – directory to store results for this image
- **args.master** (*path*) – master image to downsample
- **args.downsampler** (*string*) – name of the downsampler
- **args.downsampler_dir** (*path*) – directory to store dowsampled images
- **downsamplers** (*dict*) – downsamplers to use
- **same** (*boolean*) – *True* if accessing an existing table

6.6.5 The `Upsamplers` Class

class `operations.`**`Upsamplers`**(*upsamplers*, *metrics*, *same=False*)

Bases: `object`

This class upsamples an image and compares with its master image.

Parameters

- **upsamplers** (*dict*) – upsamplers to use
- **metrics** (*dict*) – metrics to compare with
- **same** (*boolean*) – *True* if using unchanged upsamplers

`compute`(*args*, *same*)

Perform all operations for this set of ratios.

Parameters

- **args** (`argparse.Namespace`) – arguments

- **args.dbase_file** (*path*) – database file
- **args.dbase** (`database.Database` (page 41)) – connected database
- **args.proj** (*string*) – name of the current project
- **args.silent** (*boolean*) – *True* if using silent mode
- **args.met_same** (*dict*) – unchanged metrics
- **args.metrics** (*dict*) – current metrics
- **args.do_op** (*function*) – updates the displayed progress
- **args.image** (*string*) – name of the image
- **args.image_dir** (*path*) – directory to store results for this image
- **args.master** (*path*) – master image to downsample
- **args.downsampler** (*string*) – name of the downsampler
- **args.downsampler_dir** (*path*) – directory to store dowsampled images
- **args.ratio** (*string*) – resampling ratio
- **args.small** (*path*) – downsampled image
- **args.table** (*string*) – name of the table to insert the row into
- **args.table_bak** (*string*) – name of the backup table (if it exists)
- **same** (*boolean*) – *True* if accessing an existing table

6.7 The `parsing` Module

Classes and methods used for parsing arguments and formatting help text.

`parsing._format_doc`(*docstring*)

Parse the module docstring and re-format all *reST* markup.

Note: This is a private function called when creating a new `ExquiresParser` (page 54) object.

Parameters **docstring** (*string*) – docstring to format

Returns formatted docstring

Return type *string*

parsing.**_remove_duplicates**(*input_list*)
Remove duplicate entries from a list.

Note: This is a private function called by `ListAction.__call__()` (page 57) and `RatioAction.__call__()` (page 58).

Parameters **input_list** (*list of values*) – list to remove duplicate entries from

Returns list with duplicate entries removed

Return type *list of values*

6.7.1 The `ExquiresParser` Class

class parsing.**ExquiresParser**(*description*)
Bases: `argparse.ArgumentParser`

Generic **EXQUIRES** parser.

Parameters **description** (*string*) – docstring from the calling program

parse_args(*args=None*, *namespace=None*)
Parse command-line arguments.

Parameters

- **args** (*string*) – the command-line arguments
- **namespace** (`argparse.Namespace`) – the namespace

Returns the parsed arguments

Return type `argparse.Namespace`

6.7.2 The `OperationsParser` Class

class parsing.**OperationsParser**(*description*, *update=False*)
Bases: `parsing.ExquiresParser` (page 54)

Parser used by *exquires-run* (page 24) and *exquires-update* (page 25).

Parameters

- **description** (*string*) – docstring from the calling program
- **update** (*boolean*) – *True* if called by *exquires-update* (page 25)

parse_args (*args=None*, *namespace=None*)
Parse the received arguments.

This method parses the arguments received by *exquires-run* (page 24) or :ref'exquires-update'.

Parameters

- **args** (*string*) – the command-line arguments
- **namespace** (`argparse.Namespace`) – the namespace

Returns the parsed arguments

Return type `argparse.Namespace`

6.7.3 The `StatsParser` Class

class `parsing.`**StatsParser** (*description*, *correlate=False*)
Bases: `parsing.ExquiresParser` (page 54)

Parser used by *exquires-report* (page 25) and *exquires-correlate* (page 26).

Parameters

- **description** (*string*) – docstring from the calling program
- **correlate** (*boolean*) – *True* if using *exquires-correlate* (page 26)

parse_args (*args=['-b', 'html', '-d', '_build/doctrees', '.', '_build/html']*, *namespace=None*)
Parse the received arguments.

This method parses the arguments received by *exquires-report* (page 25) or *exquires-correlate* (page 26).

Parameters

- **args** (*string*) – the command-line arguments
- **namespace** (`argparse.Namespace`) – the namespace

Returns the parsed arguments

Return type argparse.Namespace

6.7.4 The ExquiresHelp Class

class parsing.**ExquiresHelp**(*prog, indent_increment=2, max_help_position=24, width=None*)
Bases: argparse.RawDescriptionHelpFormatter

Formatter for generating usage messages and argument help strings.

This class is designed to display options in a cleaner format than the standard argparse help strings.

_fill_text(*text, width, indent*)
Fill action text with whitespace.

Parameters

- **text** (*string*) – the text to display
- **width** (*integer*) – line width
- **indent** (*string*) – indentation printed before the text

Returns the formatted text

Return type *string*

_format_action_invocation(*action*)
Format the string describing the invocation of the specified action.

Parameters **action** (argparse.Action) – the parsing action

Returns the formatted action invocation

Return type *string*

6.7.5 The ProjectAction Class

class parsing.**ProjectAction**(*option_strings, dest, nargs=None, const=None, default=None, type=None, choices=None, required=False, help=None, metavar=None*)
Bases: argparse.Action

Parser action to read a project file based on the specified name.

__call__(*parser, args, value, option_string=None*)
Parse the *-p/--project* option.

Parameters

- **parser** (`ExquiresParser` (page 54)) – the parser calling this action
- **args** (`argparse.Namespace`) – arguments
- **values** (*list of values*) – values
- **option_string** (*string*) – command-line option string

Raises `argparse.ArgumentError`

6.7.6 The `ListAction` Class

class `parsing.`**`ListAction`**(*option_strings, dest, nargs=None, const=None, default=None, type=None, choices=None, required=False, help=None, metavar=None*)

Bases: `argparse.Action`

Parser action to handle wildcards for options that support them.

When specifying aggregation options with exquires-report, this class expands any wildcards passed in arguments for the following options:

•Images

•Downsamplers

•Upsamplers

•Metrics

`__call__`(*parser, args, values, option_string=None*)

Parse any option that supports lists with wildcard characters.

Parameters

- **parser** (`ExquiresParser` (page 54)) – the parser calling this action
- **args** (`argparse.Namespace`) – arguments
- **values** (*list of values*) – values
- **option_string** (*string*) – command-line option string

Raises `argparse.ArgumentError`

6.7.7 The `RatioAction` Class

class `parsing.`**`RatioAction`**(*option_strings*, *dest*, *nargs=None*, *const=None*, *default=None*, *type=None*, *choices=None*, *required=False*, *help=None*, *metavar=None*)

Bases: `argparse.Action`

Parser action to deal with ratio ranges.

`__call__`(*parser*, *args*, *values*, *option_string=None*)

Parse the `-r/--ratio` option.

Parameters

- **parser** (`ExquiresParser` (page 54)) – the parser calling this action
- **args** (`argparse.Namespace`) – arguments
- **values** (*list of values*) – values
- **option_string** (*string*) – command-line option string

Raises `argparse.ArgumentError`

6.7.8 The `AnchorAction` Class

class `parsing.`**`AnchorAction`**(*option_strings*, *dest*, *nargs=None*, *const=None*, *default=None*, *type=None*, *choices=None*, *required=False*, *help=None*, *metavar=None*)

Bases: `argparse.Action`

Parser action to sort the correlation matrix.

`__call__`(*parser*, *args*, *value*, *option_string=None*)

Parse the `-a/--anchor` option.

Parameters

- **parser** (`ExquiresParser` (page 54)) – the parser calling this action
- **args** (`argparse.Namespace`) – arguments
- **values** (*list of values*) – values
- **option_string** (*string*) – command-line option string

Raises `argparse.ArgumentError`

6.7.9 The `SortAction` Class

class `parsing.`**`SortAction`**(*option_strings, dest, nargs=None, const=None, default=None, type=None, choices=None, required=False, help=None, metavar=None*)

Bases: `argparse.Action`

Parser action to sort the data by the appropriate metric.

`__call__`(*parser, args, value, option_string=None*)

Parse the *-s/--sort* option.

Parameters

- **parser** (`ExquiresParser` (page 54)) – the parser calling this action
- **args** (`argparse.Namespace`) – arguments
- **values** (*list of values*) – values
- **option_string** (*string*) – command-line option string

Raises `argparse.ArgumentError`

6.8 The `progress` Module

Display progress info for *exquires-run* (page 24) and *exquires-update* (page 25).

When the *-s/--silent* option is not selected in *exquires-run* (page 24) or *exquires-update* (page 25), the Progress class is used to display the appropriate information.

6.8.1 The `Progress` Class

class `progress.`**`Progress`**(*program, proj, total_ops*)

Bases: `object`

This class contains methods for displaying progress in exquires.

When *exquires-run* (page 24) and *exquires-update* (page 25) are used without silent mode enabled, this class is responsible for displaying information about the downsampling, upsampling, and comparison steps and the total progress.

Parameters

- **program** (*string*) – name of the program that is running

- **proj** (*string*) – name of the project being used
- **total_ops** (*integer*) – total number of operations

_Progress__table_bottom (*line*, *label*, *content*)
Draw the bottom row of the progress table.

This method draws one of the middle rows of the progress table. Two lines are used to draw this section of the table.

Note: This is a private method called by `cleanup()` (page 61), `complete()` (page 61), and `do_op()` (page 61).

Warning: To display the updated progress table, the screen must be refreshed by calling `self.scr.refresh()`.

Parameters

- **line** (*integer*) – line number to start drawing at
- **label** (*string*) – label for this table entry
- **content** (*string*) – content for this table entry

_Progress__table_middle (*line*, *label*, *content*)
Draw one of the middle rows of the progress table.

This method draws one of the middle rows of the progress table. Two lines are used to draw this section of the table.

Note: This is a private method called by `cleanup()` (page 61), `complete()` (page 61), and `do_op()` (page 61).

Warning: To display the updated progress table, the screen must be refreshed by calling `self.scr.refresh()`.

Parameters

- **line** (*integer*) – line number to start drawing at
- **label** (*string*) – label for this table entry
- **content** (*string*) – content for this table entry

_Progress__table_top (*line*, *label*, *content*)
Draw the top row of the progress table.

This method draws the first row of the progress table, which displays the project name. Three lines are used to draw this section of the table.

Note: This is a private method called by `cleanup()` (page 61), `complete()` (page 61), and `do_op()` (page 61).

Warning: To display the updated progress table, the screen must be refreshed by calling `self.scr.refresh()`.

Parameters

- **line** (*integer*) – line number to start drawing at
- **label** (*string*) – label for this table entry
- **content** (*string*) – content for this table entry

cleanup()
Indicate that files are being deleted.

complete()
Complete the progress indicator.

Call this method to indicate success once all operations have been performed.

Note: The completion screen is displayed for a half second.

Warning: To restore the terminal after completion, destruct the `Progress` (page 59) object by calling *del prg* (where *prg* is the object to destruct).

do_op(*args*, *upsampler=None*, *metric=None*)
Update the progress indicator.

•If no upsampler is specified, *operation=downsampling*

•If an upsampler is specified, but no metric, *operation=upsampling*

•If an upsampler and metric are specified, *operation=comparing*

Parameters

- **args** (`argparse.Namespace`) – arguments
- **args.image** (*string*) – image being processed
- **args.downsampler** (*string*) – downsampler being used
- **args.ratio** (*string*) – resampling ratio being used
- **upsampler** (*string*) – upsampler being used

- **metric** (*string*) – metric being used

6.9 The `report` Module

Print a formatted table of aggregate image difference data.

Each database table in the current project contains data for a single image, downsampler, and ratio. Each row represents an upsampler and each column represents a difference metric. By default, the data across all rows and columns of all tables is aggregated. Use the appropriate option flags to aggregate across a subset of the database.

Features:

- *-R/*:option'–ratio' supports hyphenated ranges (for example, '1-3 5' gives '1 2 3 5')
- *-U/--up*, *-I/--image*, *-D/--down* and *-M/--metric* support wildcard characters

`report._print_table` (*args*)

Print a table of aggregate image comparison data.

Since the database contains error data for several images, downsamplers, ratios, upsamplers, and metrics, it is convenient to be able to specify which of these to consider. This method aggregates the data for each relevant column in the appropriate tables.

Note: This is a private function called by `main()` (page 63).

Parameters

- **args** (`argparse.Namespace`) – arguments
- **args.dbase_file** (*path*) – database file
- **args.image** (*list of strings*) – selected image names
- **args.down** (*list of strings*) – selected downsampler names
- **args.ratio** (*list of strings*) – selected ratios
- **args.up** (*list of strings*) – selected upsampler names
- **args.metric** (*list of strings*) – selected metric names
- **args.metrics_d** (*dict*) – all metric names
- **args.file** (*path*) – output file

- **args.digits** (*integer*) – number of digits to print
- **args.latex** (*boolean*) – *True* if printing a LaTeX-formatted table
- **args.rank** (*boolean*) – *True* if printing Spearman (fractional) ranks
- **args.merge** (*boolean*) – *True* if printing merged Spearman ranks
- **args.sort** (*string*) – metric to sort by
- **args.show_sort** (*boolean*) – *True* if the sort column should be displayed

`report.main()`
Run *exquires-report* (page 25).

Parse the command-line arguments and print the aggregate data table.

6.10 The `run` Module

Compute error data for the entries in the specified project file.

The project file is read to determine which images, downsamplers, ratios, upsamplers, and metrics to use. If a database file already exists for this project, it will be backed up and a new one will be created.

Each image will be downsampled by each of the ratios using each of the downsamplers. The downsampled images will then be upsampled back to their original size (840x840) using each of the upsamplers. The upsampled images will be compared to the original images using each of the metrics and the results will be stored in the database file.

If you make changes to the project file and wish to only compute data for these changes rather than recomputing everything, use *exquires-update* (page 25).

To view aggregated error data, use *exquires-report* (page 25).

`run._run(args)`
Create a new project database and populate it with computed data.

Note: This is a private function called by `main()` (page 63).

Parameters

- **args** (`argparse.Namespace`) – arguments
- **args.config_file** (*path*) – current configuration file

run.**main**()
Run *exquires-run* (page 24).

Create a database for the specified project file.

Warning: If a database already exists for this project, it will be overwritten.

6.11 The `stats` Module

A collection of methods for producing statistical output.

stats.**_format_cell**(*cell*, *digits*)
Return a formatted version of this cell of the data table.

Note: This is a private function called by `print_normal()` (page 65) and `print_latex()` (page 65).

Parameters

- **cell** (*string*) – cell to format
- **digits** (*integer*) – maximum number of digits to display

Returns the formatted cell

Return type *string*

stats.**get_aggregate_table**(*dbase*, *upsamplers*, *metrics_d*, *tables*)
Return a table of aggregate image difference data.

Parameters

- **dbase** (`database.Database` (page 41)) – connected database
- **upsamplers** (*list of strings*) – upsamplers (rows) of the table
- **metrics_d** (*dict*) – metrics (columns) of the table in dictionary form
- **tables** (*list of strings*) – names of database tables to aggregate across

Returns table of aggregate image difference data

Return type *list of lists*

stats.**get_merged_ranks** (*printdata, metrics_desc, sort_index*)
Return a table of merged Spearman ranks based on a data table.

Parameters

- **printdata** (*list of lists*) – table of data to print
- **metrics_desc** (*list of integers*) – list of 0s and 1s (where 1 is 'descending')
- **sort_index** (*integer*) – index of the column to sort by

Returns table of merged ranks

Return type *list of lists*

stats.**get_ranks** (*printdata, metrics_desc, sort_index*)
Return a table of Spearman (Fractional) ranks based on a data table.

Parameters

- **printdata** (*list of lists*) – table of data to print
- **metrics_desc** (*list of integers*) – list of 0s and 1s (where 1 is 'descending')
- **sort_index** (*integer*) – index of the column to sort by

Returns table of ranks

Return type *list of lists*

stats.**print_latex** (*printdata, args, header, matrix=False*)
Print the processed data table with LaTeX formatting.

Parameters

- **printdata** (*list of lists*) – table of data to print
- **args** (`argparse.Namespace`) – arguments
- **args.file** (*path*) – path to write the aggregated error table
- **args.digits** (*integer*) – maximum number of digits to display
- **header** (*list of strings*) – table headings
- **matrix** (*boolean*) – *True* if printing a correlation matrix

stats.**print_normal** (*printdata, args, header, matrix=False*)
Print the processed data table with normal formatting.

Parameters

- **printdata** (*list of lists*) – table of data to print
- **args** (`argparse.Namespace`) – arguments
- **args.file** (*path*) – path to write the aggregated error table
- **args.digits** (*integer*) – maximum number of digits to display
- **header** (*list of strings*) – table headings
- **matrix** (*boolean*) – *True* if printing a correlation matrix

6.12 The `tools` Module

A collection of convenience methods.

`tools.`**`create_dir`**(*base_dir*, *relative_dir=''*)

Create a directory if it doesn't already exist and return it.

Parameters

- **base_dir** (*path*) – base directory within which to create the directory
- **relative_dir** (*path*) – directory to create inside the base directory

Returns the created directory

Return type *path*

`tools.`**`prune_metrics`**(*keys*, *metrics_d*)

Prune a dictionary of metrics using a list of keys.

Parameters

- **keys** (*list of strings*) – keys to retain
- **metrics_d** (*dict*) – metrics to prune

Returns pruned metrics

Return type *dict*

6.13 The `update` Module

Compute new error data for changes to the user-specified project file.

The project file is inspected to determine which changes have been made. Items that have been removed will result in entries being removed from the database. Items that have been changed or added will result in new data being computed and added to the database file. If no changes have been made to the project file, the database will not be updated.

If you wish to recompute all data based on your project file rather than simply updating it with the changes, use *exquires-run* (page 24).

To view aggregated error data, use *exquires-report* (page 25).

`update.`**`_get_namespaces`** (*config_file*, *config_bak*)

Return all necessary configuration namespaces.

This function returns four namespaces that specify which images, downsamplers, ratios, upsamplers, and metrics to use when creating or updating a project database:

- *current* – all entries in current project file
- *new* – entries only in currenty project file
- *old* – entries only in previous project file
- *same* – entries common to both project files

Note: This is a private function called by `_update()` (page 68).

Parameters

- **config_file** (*path*) – current configuration file
- **config_bak** (*path*) – previous configuration file

Returns the current, new, old, and same namespaces

Return type `argparse.Namespace`

`update.`**`_subtract`** (*dict1*, *dict2*)

Subtract dictionary *dict2* from *dict1* and return the difference.

This function creates a new *dict*, then iterates over *dict1* and adds all entries that are not found in *dict2*.

Note: This is a private function called by `_get_namespaces()` (page 67).

Parameters

- **dict1** (*dict*) – dictionary to subtract from
- **dict2** (*dict*) – dictionary to subtract

Returns dict1 - dict2

Return type *dict*

update.**_update**(*args*)

Update the database.

Note: This is a private function called by `main()` (page 68).

Parameters

- **args** (`argparse.Namespace`) – arguments
- **args.config_file** (*path*) – current project file
- **args.config_bak** (*path*) – previous project file

update.**main**()

Run *exquires-update* (page 25).

Update the project database based on changes to the project file.

Note: If the update fails, the previous database will be restored.

CHAPTER

SEVEN

LICENSE INFORMATION

EXQUIRES is released under the BSD 2-Clause License as outlined below:

CHAPTER

EIGHT

CHANGELOG

8.1 Version 0.9.9.3

- Fixed a bug in exquires-correlate
- Fixed docstring formatting for help messages

8.2 Version 0.9.9.2

- Replaced EANBQH example with Nohalo-LBB (Locally Bounded Bicubic)

8.3 Version 0.9.9.1

- Switched from ImageMagick 7 alpha to ImageMagick 6.8.0-2 (or newer)
- Improved EANBQH example

8.4 Version 0.9.9

- Added -a/–anchor option to exquires-correlate
- Added superior error handling

- Improved ratio parsing
- Reformatted documentation

8.5 Version 0.9.8.3

- Modified matrix formatting for exquires-correlate

8.6 Version 0.9.8.2

- Important bugfix for exquires-update

8.7 Version 0.9.8.1

- Added -l/--linear option to eanbqh.py

8.8 Version 0.9.8

- Major code revision (10/10 pylint score)
- Added Spearman's rank cross-correlation

8.9 Version 0.9.7

- Initial public release

CHAPTER

NINE

TODO

The following features are planned for future versions of **EXQUIRES**:

- **Add greater support for resuming from crashes:**
  - Currently, *exquires-update* (page 25) depends on a backup of your project file to determine how to modify the database.
  - If *exquires-update* (page 25) is interrupted, the original database file is restored.
  - Version 1.0 will retain the changes made to the database file and will be able to resume the update operation.
- **Add report output formats:**
  - Currently, *exquires-report* (page 25) and *exquires-correlate* (page 26) can produce plaintext or latex output.
  - Other formats will be added as demand arises.
- **Add Kendall's rank correlation:**
  - Currently, *exquires-correlate* (page 26) produces cross-correlation matrices using Spearman's rank correlation.
  - Kendall's rank correlation is more complicated than Spearman's rank correlation, but the result is more informative, so it will eventually be added.
- **Add unit tests:**
  - Every module should have a unit test written for it.
  - Version 1.0 will introduce unit tests.

PYTHON MODULE INDEX

INDEX

Symbols

A

B

C

D

E

G

I

L

M

N

O

P

R

S

T

U

X

Bibliography

[1] Erik H. W. Meijering, Karel J. Zuiderveld, and Max A. Viergever. Image reconstruction by convolution with symmetrical piecewise nth-order polynomial kernels. *IEEE Transactions on Image Processing*, 8(2):192–201, 1999.

[2] Jia-Guu Leu. Image enlargement based on a step edge model. *Pattern Recognition*, 33(12):2055–2073, Dec. 2000.

[3] Guoping Qiu. Interresolution look-up table for improved spatial magnification of image. *Journal of Visual Communication and Image Representation*, 11(4):360–373, Dec. 2000.

[4] Bryan S. Morse and Duane Schwartzwald. Image magnification using level-set reconstruction. In *CVPR (1)*, pages 333–340. IEEE Computer Society, 2001.

[5] King-Hong Chung, Yik-Hing Fung, and Yuk-Hee Chan. Image enlargement using fractal. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*, volume 6, pages 273–276, Apr. 2003.

[6] D. Darian Muresan and Thomas W. Parks. Adaptively quadratic (AQua) image interpolation. *IEEE Transactions on Image Processing*, 13(5):690–698, 2004.

[7] Dan Su and Philip Willis. Image interpolation by pixel-level data-dependent triangulation. *Computer Graphics Forum*, 23(2):189–202, 2004.

[8] Mei-Juan Chen, Chin-Hui Huang, and Wen-Li Lee. A fast edge-oriented algorithm for image interpolation. *Image and Vision Computing*, 23:791798, 2005.

[9] Carlos Miravet and Francisco B. Rodríguez. Accurate and robust image superresolution by neural processing of local image representations. In Włodzisław Duch, Janusz Kacprzyk, Erkki Oja, and Sławomir Zadrożny, editors, *Proceedings of the 15th International Conference on Artificial Neural Networks: Biological Inspirations – Part I*, volume 3696 of *ICANN 2005*, pages 499–505, Berlin, Heidelberg, 2005. Springer-Verlag.

[10] Cheng-Hsiung Hsieh, Ren-Hsien Huang, and Ting-Yu Feng. One-dimensional grey polynomial interpolators for image enlargement. In *ACIS-ICIS*, pages 450–456. IEEE Computer Society, 2007.

[11] Emil Dumic, Sonja Grgic, and Mislav Grgic. Hidden influences on image quality when comparing interpolation methods. In *15th International Conference on Systems, Signals and Image Processing, IWSSIP 2008*, pages 367–372, Bratislava, Slovakia, Jun. 2008.

[12] Alain Horé, François Deschênes, and Djemel Ziou. A new super-resolution algorithm based on areas pixels and the sampling theorem of Papoulis. In Campilho and Kamel [150], pages 97–109.

[13] Luming Liang. Image interpolation by blending kernels. *IEEE Signal Processing Letters*, 15:805–808, 2008.

[14] Nicolas Robidoux, Adam Turcotte, Minglun Gong, and Annie Tousignant. Fast exact area image upsampling with natural biquadratic histosplines. In Campilho and Kamel [150], pages 85–96.

[15] Weisheng Dong, Lei Zhang, Guangming Shi, and Xiaolin Wu. Nonlocal back-projection for adaptive image enlargement. In *ICIP*, pages 349–352. IEEE, 2009.

[16] Nicolas Robidoux, Minglun Gong, John Cupitt, Adam Turcotte, and Kirk Martinez. CPU, SMP and GPU implementations of Nohalo level 1, a fast co-convex antialiasing image resampler. In Bipin C. Desai, Carson Kai-Sang Leung, and Olga Ormandjieva, editors, *C3S2E*, ACM International Conference Proceeding Series, pages 185–195. ACM, 2009.

[17] Pascal Getreuer. Linear methods for image interpolation. *Image Processing On Line*, 2011.

[18] Pascal Getreuer. Roussos-Maragos tensor-driven diffusion for image interpolation. *Image Processing On Line*, 2011.

[19] Michael Unser, Philippe Thévenaz, and Leonid Yaroslavsky. Convolution-based interpolation for fast, high-quality rotation of images. *IEEE Transactions on Image Processing*, 4(10):1371–1381, Oct. 1995.

[20] Keiran G. Larkin, Michael A. Oldfield, and Hanno Klemm. Fast fourier method for the accurate rotation of sampled images. *Optics Communications*, 139(1-3):99–106, Jun. 1997.

[21] Erik H. W. Meijering, Wiro J. Niessen, Josien P. W. Pluim, and Max A. Viergever. Quantitative comparison of sinc-approximating kernels for medical image interpolation. In Chris J. Taylor and Alan C. F. Colchester, editors, *MICCAI*, volume 1679 of *Lecture Notes in Computer Science*, pages 210–217. Springer, 1999.

[22] Erik H. W. Meijering. Spline interpolation in medical imaging: Comparison with other convolution-based approaches. In Moncef Gabbouj and Pauli Kuosmanen, editors, *Signal Processing X: Theories and ApplicationsProceedings of EUSIPCO 2000*, volume IV, page 19891996, Tampere, 2000. The European Association for Signal Processing.

[23] Philippe Thévenaz, Thierry Blu, and Michael Unser. Interpolation revisited. *IEEE Transactions on Medical Imaging*, 19(7):739–758, 2000.

[24] Philippe Thévenaz, Thierry Blu, and Michael Unser. *Handbook of Medical Imaging, Processing and Analysis*, chapter Image Interpolation and Resampling, pages 393–420. Academic Press, San Diego CA, 2000.

[25] Erik H. W. Meijering, Wiro J. Niessen, and Max A. Viergever. Quantitative evaluation of convolution-based methods for medical image interpolation. *Medical Image Analysis*, 5(2):111–126, 2001.

[26] Diego Nehab and Hugues Hoppe. Generalized sampling in computer graphics. Technical Report E022/2011 and MSR-TR-2011-16, IMPA and Microsoft Research, Feb. 2011.

[27] Ellis Freedman. Medical images. `http://remotesensing.page.tl/Medical-Images.htm`, 2012.

[28] James F. Blinn. What we need around here is more aliasing. *IEEE Computer Graphics and Applications*, 9(1):75–79, January 1989.

[29] Ken Turkowski. Filters for common resampling tasks. In Andrew S. Glassner, editor, *Graphics Gems*, pages 147–165. Academic Press Professional, Inc., San Diego, CA, USA, 1990.

[30] Neil A. Dodgson. Quadratic interpolation for image resampling. *IEEE Transactions on Image Processing*, 6(9):1322–1326, 1997.

[31] Amir Said. A new class of filters for image interpolation and resizing. In *IEEE International Conference on Image Processing z(ICIP)*, pages 217–220. IEEE, 2007.

[32] Chantal Racette. Numerical analysis of diagonal-preserving, ripple-minimizing and low-pass image resampling methods. Master's thesis, Laurentian University, Sudbury, Ontario, Canada, 2011. http://arxiv.org/abs/1204.4734.

[33] Neil A. Dodgson. Image resampling. Technical Report UCAM-CL-TR-261, University of Cambridge, Computer Laboratory, Aug. 1992.

[34] Peter J. Burt and Edward H. Adelson. The laplacian pyramid as a compact image code. In Martin A. Fischler and Oscar Firschein, editors, *Readings in computer vision: issues, problems, principles, and paradigms*, pages 671–679. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.

[35] John D. Villasenor, Benjamin Belzer, and Judy Liao. Wavelet filter evaluation for image compression. *IEEE Transactions on Image Processing*, 4(8):1053–1060, August 1995.

[36] Xin Li and Michael T. Orchard. New edge-directed interpolation. *IEEE Transactions on Image Processing*, 10(10):1521–1527, October 2001.

[37] Michal Irani and Shmuel Peleg. Motion analysis for image enhancement: Resolution, occlusion, and transparency. *Journal of Visual Communication and Image Representation*, 4:324–335, 1993.

[38] Shengyang Dai, Mei Han, Ying Wu, and Yihong Gong. Bilateral back-projection for single image super resolution. In *Proceedings of the 2007 IEEE International Conference on Multimedia and Expo, ICME 2007, July 2-5, 2007, Beijing, China*, pages 1039–1042. IEEE, 2007.

[39] Jong-Ki Han and Seung-Ung Baek. Parametric cubic convolution scaler for enlargement and reduction of image. *IEEE Transactions on Consumer Electronics*, 46(2):247–256, May 2000.

[40] Yuval Fisher. *Fractal Image Comression: Theory and Application.* Springer Verlag, New York, NY, USA, 1995.

[41] Shiqi Yu, Jia Wu, Shulin Shang, and Vincent Etienne. SIVP–scilab image and video processing toolbox. http://sivp.sourceforge.net/, 2011. Computer package.

[42] John Cristy, Kelly Bergougnoux, Rod Bogart, John W. Peterson, Nathan Brown, Mike Chiarappa, Thomas R. Crimmins, Troy Edwards, Jaroslav Fojtik, Francis J. Franklin, Markus Friedl, Bob Friesenhahn, Michael Halle, David Harr, Christopher R. Hawks, Paul Heckbert, Peder Langlo, Rick Mabry, Catalin Mihaila, David

Pensak, Chantal Racette, William Radcliffe, Glenn Randers-Pehrson, Paul Raveling, Nicolas Robidoux, Leonard Rosenthol, Kyle Shorter, Lars Ruben Skyum, Alvy Ray Smith, Eric Ray Lyons, Michael Still, Anthony Thyssen, Milan Votava, Fred Weinhaus, and Alexander Zimmermann. ImageMagick. `http://imagemagick.org`, 2012. Computer program.

[43] Nicos Dessipris, Kirk Martinez, John Cupitt, Ruven Pillay, Steve Perry, Lars Raffelt, David Saunders, Jean-Philippe Laurant, Ahmed Abood, Helene Chahine, Joe Padfield, Andrey Kiselev, Lev Serebryakov, Simon Goodall, Konrad Lang, Markus Wollgarten, Jesper Friis, Tom Vajzovic, Chris Leick, Hans Breuer, Dennis Lubert, Jose Manuel Menendez Garcia, Javier Alejandre Arenas, Juan Torres Arjona, Nicolas Robidoux, Chantal Racette, and Adam Turcotte. VIPS (Virtual Image Processing System). `http://www.vips.ecs.soton.ac.uk`, 2011. Computer program.

[44] Anastasios Roussos and Petros Maragos. Vector-valued image interpolation by an anisotropic diffusion-projection pde. In *Proceedings of the 1st international conference on Scale space and variational methods in computer vision*, SSVM'07, pages 104–115, Berlin, Heidelberg, 2007. Springer-Verlag.

[45] onOne Software. Perfect resize. `http://www.ononesoftware.com`, 2012. Computer software.

[46] François Malgouyres and Frederic Guichard. Edge direction preserving image zooming: A mathematical and numerical analysis. *SIAM Journal of Numerical Analysis*, 39(1):1–37, January 2001.

[47] Pascal Getreuer. Image interpolation with geometric contour stencils. *Image Processing On Line*, 2011, 2011.

[48] Angelos Amanatiadis and Ioannis Andreadis. A survey on evaluation methods for image interpolation. *Measurement Science and Technology*, 20(10):104015.1–104015.9, 2009.

[49] Robert G. Keys. Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-29(6):1153–1160, Dec. 1981.

[50] Alan P. Schaum. Theory and design of local interpolators. *Computer Vision, Graphics, and Image Processing*, 55(6):464–481, 1993.

[51] Ismail German. Short kernel fifth-order interpolation. *IEEE Transactions on Signal Processing*, 45(5):1355–1359, May 1997.

[52] Thierry Blu, Philippe Thévenaz, and Michael Unser. Minimum support interpolators with optimum approximation properties. In *Proceedings of the 1998 IEEE International Conference on Image Processing (ICIP'98)*, volume III, pages 242–245, Oct. 1998.

[53] Rich Franzen. Kodak lossless true color image suite. `http://r0k.us/graphics/kodak/`, 2002. Digital image collection.

[54] Thierry Blu, Philippe Thévenaz, and Michael Unser. MOMS: maximal-order interpolation of minimal support. *IEEE Transactions on Image Processing*, 10(7):1069–1080, July 2001.

[55] Thierry Blu, Philippe Thévenaz, and Michael Unser. Linear interpolation revitalized. *IEEE Transactions on Image Processing*, 13(5):710–719, May 2004.

[56] Laurent Condat, Thierry Blu, and Michael Unser. Beyond interpolation: optimal reconstruction by quasi-interpolation. In *Proceedings of the 2005 IEEE International Conference on Image Processing (ICIP'05)*, volume 1, pages 33–36, 2005.

[57] Marco Dalai, Riccardo Leonardi, and Pierangelo Migliorati. Efficient digital prefiltering for least-squares linear approximation. In Luigi Atzori, Daniele D. Giusto, Riccardo Leonardi, and Fernando Pereira, editors, *Proceedings of the 9th international conference on Visual Content Processing and Representation*, volume 3893 of *VLBV'05*, pages 161–169, Berlin, Heidelberg, 2005. Springer-Verlag.

[58] Don P. Mitchell and Arun N. Netravali. Reconstruction filters in computer graphics. *SIGGRAPH Computer Graphics*, 22(4):221–228, Jun. 1988.

[59] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[60] Zhou Wang and Alan C. Bovik. A universal image quality index. *IEEE Signal Processing Letters*, 9(3):81–84, March 2002.

[61] Qianqian Xu, Qingming Huang, and Yuan Yao. Online crowdsourcing subjective image quality assessment. In *Proceedings of the 20th ACM international conference on Multimedia*, MM '12, pages 359–368, New York, NY, USA, 2012. ACM.

[62] Patrick Le Callet and Florent Autrusseau. Subjective quality assessment IRCCyN/IVC database, 2005. `http://www.irccyn.ec-nantes.fr/ivcdb/`.

[63] Hamid R. Sheikh, Zhou Wang, Lawrence Cormack, and Alan C. Bovik. LIVE image quality assessment database release, 2008. http://live.ece.utexas.edu/research/quality/.

[64] Charles Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101, Jan. 1904.

[65] Jerrold H. Zar. *Spearman Rank Correlation.* John Wiley & Sons, Ltd, 2005.

[66] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. The SSIM index for image quality assessment. http://ece.uwaterloo.ca/~z70wang/research/ssim, Jan. 2012. SSIM website.

[67] Wikipedia. Ranking. http://en.wikipedia.org/wiki/Ranking, Jun. 2012.

[68] Karl Pearson. *On further methods of determining correlation.* Drapers' company research memoirs: Biometric series. Dulau and co., 1907.

[69] Maurice G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, Jun. 1938.

[70] Kendall's rank correlation. www.statsdirect.com/help/nonparametric_methods/kend.htm, 2000. StatsDirect Statistical Software online help.

[71] Spearman's rank correlation. www.statsdirect.com/help/nonparametric_methods/spear.htm, 2000. StatsDirect Statistical Software online help.

[72] Charles A. Poynton. Gamma FAQ - Frequently Asked Questions about Gamma. http://www.poynton.com/notes/colour_and_gamma/GammaFAQ.html, Dec. 2002. Typeset version: http://www.poynton.com/PDFs/GammaFAQ.pdf.

[73] Charles A. Poynton. Color FAQ - Frequently Asked Questions about Color. http://www.poynton.com/notes/colour_and_gamma/ColorFAQ.html, Nov. 2006. Typeset version: http://www.poynton.com/PDFs/ColorFAQ.pdf.

[74] Anthony Thyssen. ImageMagick v6 examples – resize or scaling (general techniques). http://www.imagemagick.org/Usage/resize, Oct. 2012. (online documentation).

[75] Charles A. Poynton. Rehabilitation of gamma. In Bernice E. Rogowitz and Thrasyvoulos N. Pappas, editors, *Human Vision and Electronic Imaging III*, volume 3299 of *Proceedings of The International Society for Optical Engineering*, pages 232–249. SPIE, 1998.

[76] Wikipedia. sRGB. `http://en.wikipedia.org/wiki/SRGB`, Jan. 2013.

[77] Wikipedia. CIE 1931 color space. `http://en.wikipedia.org/wiki/CIE_1931_color_space`, Jan. 2013.

[78] Spigget. CIE chart with sRGB gamut. `http://en.wikipedia.org/wiki/File:Cie_Chart_with_sRGB_gamut_by_spigget.png`, Feb. 2007. File from Wikimedia Commons.

[79] International Colour Consortium. ICC frequently asked questions. `http://www.color.org/faqs.xalter`, 2001.

[80] Nicolas Robidoux, Jean-François Avon, Anthony Barnett, John Cupitt, Jana Duncan, Minglun Gong, Holly Graham, Henry Ho, Kirk Martinez, Michael Muré, Mukund Sivaraman, Adam Turcotte, and Luiz E. Vasconcellos. 16bit840x840images test image bank. `http://www.imagemagick.org/download/image-bank/16bit840x840images`, May 2012.

[81] International Colour Consortium. sRGB profiles. `http://www.color.org/srgbprofiles.xalter`, 2001.

[82] Nicolas Robidoux. sRGB v4 is gaining traction. Private communication, Jan. 2013.

[83] Contributors of photivo.org. Photivo. `http://photivo.org`, 2011. Computer program.

[84] Gábor Horváth and contributors. Rawtherapee. `http://rawtherapee.com`, 2011. Computer program.

[85] John Cupitt, Joe Padfield, Hans Breuer, Rich Lott, and Leo Davidson. NIP2 (New Image Processor 2) Version 7.17. `http://www.vips.ecs.soton.ac.uk/index.php?title=Nip2`, 2010. Computer program.

[86] Nicolas Robidoux. Use Perceptual rendering intent across the board. Private communication, Apr. 2012.

[87] Nicolas Robidoux. Colour space conversions should be carefully checked. Private communication, Feb. 2013.

[88] Hokusai Katsushika. Kanagawa oki nami ura (the great wave off shore of Kanagawa). http://www.loc.gov/pictures/resource/jpd.02018, 1826–1833. High quality scan, by the Library of Congress, of a woodcut reproduction.

[89] John Cristy. MagickCore Image Resize Methods - Resize. http://www.imagemagick.org/api/MagickCore/resize_8c_source.html, 2011. Computer program.

[90] Anthony Thyssen. Usage under Windows. http://www.imagemagick.org/Usage/windows, Mar. 2012. Section of ImageMagick v6 Examples (online documentation).

[91] Craig DeForest. On re-sampling of solar images. *Solar Physics*, 219:3–23, 2004.

[92] Paul S. Heckbert. Fundamentals of texture mapping and image warping. Master's thesis, University of California, Berkeley, CA, USA, Jun. 1989.

[93] Anthony Thyssen. ImageMagick v6 examples – resampling filters. http://www.imagemagick.org/Usage/filter, Oct. 2012. (online documentation).

[94] Steven W. Smith. *The Scientist & Engineer's Guide to Digital Signal Processing*, chapter 16. California Technical Pub., Mar. 1998.

[95] Erik H. W. Meijering, Wiro J. Niessen, and Max A. Viergever. The Sinc-approximating kernels of classical polynomial interpolation. In *Proceedings of the International Conference on Image Processing, ICIP 1999*, pages 652–656, 1999.

[96] Nicolas Robidoux. The real name of the "parabola" window function is "Welch". http://www.imagemagick.org/discourse-server/viewtopic.php?f=3&t=21637&p=88743#p88743, Aug. 2012. ImageMagick Forums post.

[97] Thomas Theußl, Helwig Hauser, and Eduard Gröller. Mastering windows: improving reconstruction. In *Volviz*, pages 101–108, 2000.

[98] Nicolas Robidoux. Variants of Kaiser Sinc-windowed Sinc to be tested. Private communication, May 2012.

[99] David P. Morgan. *Surface Acoustic Wave Filters: With Applications to Electronic Communications and Signal Processing*. Academic Press, second edition, 2007.

[100] Anthony Thyssen. ImageMagick v6 examples – distorting images. http://www.imagemagick.org/Usage/distorts, Mar. 2012. (online documentation).

[101] Nicolas Robidoux, Adam Turcotte, Chantal Racette, Anthony Thyssen, John Cupitt, and Øyvind Kolås. GEGL (GEneric Graphics Library) NoHalo sampler Version 0.2.0. http://git.gnome.org/browse/gegl/tree/gegl/buffer/gegl-sampler-nohalo.c, 2012. Computer program.

[102] Nicolas Robidoux. EWA Robidoux. Private communication, Aug. 2012.

[103] Nicolas Robidoux. GEGL (GEneric Graphics Library) LoHalo sampler Version 0.2.0. http://git.gnome.org/browse/gegl/tree/gegl/buffer/gegl-sampler-lohalo.c, 2012. Computer program.

[104] Nicolas Robidoux. RE: BC-splines with 2C+B=1 are optimal for EWA resampling. http://www.imagemagick.org/discourse-server/viewtopic.php?f=22&t=19823&start=15#p78921, Dec. 2011. ImageMagick Forums post.

[105] Nicolas Robidoux. EWA RobidouxSharp. Private communication, Aug. 2012.

[106] Henry Ho. Re: The final version :). http://forums.dpreview.com/forums/read.asp?forum=1006&message=41366611, Apr. 2012. "What about EWA Lagrange or Catrom instead of USM?".

[107] Anthony Thyssen. Interpolated cylindrical filters. http://www.imagemagick.org/Usage/filter/#cyl_interpolated, Oct. 2012. Section of ImageMagick v6 Examples – Resize or Scaling (online documentation).

[108] Nicolas Robidoux. Anthony Thyssen put the EWA Catmull-Rom cardinal basis function on the Web as early as March 2011, maybe before. Private communication, Oct. 2012.

[109] Nicolas Robidoux. Re: proper scaling of the Jinc filter for EWA use. http://www.imagemagick.org/discourse-server/viewtopic.php?f=22&t=19636&sid=0dfac6329f064ec2728052947b602ea0&start=75#p79521, Dec. 2011. ImageMagick Forums post.

[110] Nicolas Robidoux. Re: proper scaling of the Jinc filter for EWA use. http://www.imagemagick.org/discourse-server/viewtopic.php?f=22&t=19636&start=60#p79475, Dec. 2011. ImageMagick Forums post.

[111] Nicolas Robidoux. Re: proper scaling of the Jinc filter for EWA use. http://www.imagemagick.org/discourse-server/viewtopic.

php?f=22&t=19636&p=84238#p84238, Apr. 2012. ImageMagick Forums post.

[112] Ken W. Brodlie, Petros Mashwama, and Sohail Butt. Visualization of surface data to preserve positivity and other simple constraints. *Computers & Graphics*, 19(4):585–594, Jul.–Aug. 1995.

[113] Nicolas Robidoux, Chantal Racette, John Cupitt, and Adam Turcotte. VIPS (Virtual Image Processing System) Nohalo Version 7.22. http://github.com/jcupitt/libvips/blob/master/libvips/resample/nohalo.cpp, 2010. Computer program.

[114] Wikipedia. Color difference. http://en.wikipedia.org/wiki/Color_difference, Jun. 2012.

[115] Roderick McDonald, editor. *Colour Physics for Industry*. Society of Dyers and Colourists, 2nd edition, 1997.

[116] John Cupitt. VIPS (Virtual Image Processing System) conversion from LCh to CMC. https://github.com/jcupitt/libvips/blob/master/libvips/colour/LCh2UCS.c, 2012. Computer program.

[117] Wikipedia. Quasimetrics. http://en.wikipedia.org/wiki/Metric_%28mathematics%29#Quasimetrics, Feb. 2013.

[118] Steve Upton. ColorFAQs - Delta-E - the color difference. *CHROMiX ColorNews*, 17, Feb. 2005. http://www.colorwiki.com/wiki/Delta_E:_The_Color_Difference.

[119] John Cupitt. VIPS (Virtual Image Processing System) computation of dE CMC. https://github.com/jcupitt/libvips/blob/master/libvips/colour/dECMC.c, 2012. Computer program.

[120] Frank J. J. Clarke, Roderick P. McDonald, and Bryan Rigg. Modification to the JPC 79 colour-difference formula. *Journal of the Society of Dyers and Colourists*, 100(4):128–132, 1984.

[121] John Cupitt. VIPS computation of CMC 1:1. Private Communication, Feb. 2013.

[122] Wikipedia. Structural similarity. http://en.wikipedia.org/wiki/Structural_similarity, Feb. 2012.

[123] Zhou Wang. SSIM Index with automatic downsampling, Version 1.0. http://ece.uwaterloo.ca/~z70wang/research/ssim/ssim.m, 2009. MATLAB source code.

[124] Zhou Wang. SSIM Index, Version 1.0. http://ece.uwaterloo.ca/~z70wang/research/ssim/ssim_index.m, 2003. MATLAB source code.

[125] Nicolas Robidoux. Further refactoring of the computation of the Structural SIMilarity Index. Private communication, Oct. 2011.

[126] Michael Foord and Nicola Larosa. ConfigObj 4 introduction and reference. http://www.voidspace.org.uk/python/configobj.html, Feb. 2010. Python module documentation.

[127] Gerhard Häring. sqlite3 — DB-API 2.0 interface for SQLite databases. http://docs.python.org/library/sqlite3.html, 2012. Python module documentation.

[128] Hwaci. SQLite Query Language: ALTER TABLE. http://www.sqlite.org/lang_altertable.html, 2012. Description of SQL syntax understood by SQLite.

[129] Travis E. Oliphant. NumPy. http://www.numpy.org, 2013. Website for the NumPy Python module.

[130] Steven J. Bethard. argparse - Python command line parsing. http://code.google.com/p/argparse, 2011. Website for the argparse Python module.

[131] Guido van Rossum. fnmatch — Unix filename pattern matching. http://docs.python.org/library/fnmatch.html, 2012. Python module documentation.

[132] Johann C. Rocholl. pep8 - Python style guide checker. http://pypi.python.org/pypi/pep8, Jun. 2012. Python Package Index listing.

[133] Guido van Rossum and Barry Warsaw. PEP 8 – Style Guide for Python Code. http://www.python.org/dev/peps/pep-0008, Jul. 2001. Python Enhancement Proposal.

[134] Logilab. pylint. http://www.logilab.org/857, 2012. Website of the Pylint source code analyzer.

[135] David Goodger. reStructuredText. http://docutils.sourceforge.net/rst.html, Sep. 2010. Website for the reStructuredText markup syntax.

[136] David Goodger. Docutils: Documentation Utilities. http://docutils.sourceforge.net, Jun. 2012. Website for the Docutils text processing system.

[137] Georg Brandl. Sphinx 1.1.3 documentation. http://sphinx.pocoo.org, Mar. 2012. Python module documentation.

[138] Zhou Wang, Alan C. Bovik, and Ligang Lu. Why is image quality assessment so difficult? In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 4, pages 3313–3316, 2002.

[139] Hamid R. Sheikh and Alan C. Bovik. Image information and visual quality. *IEEE Transactions on Image Processing*, 15(2):430–444, Feb. 2006.

[140] Xuemei Zhang and Brian A. Wandell. Color image fidelity metrics evaluated using image distortion maps. *Signal Processing*, 70(3):201–214, 1998.

[141] Nicolas Robidoux. Only use C^1 methods to enlarge images by a large factor. Private communication, Jul. 2012.

[142] Nicolas Robidoux. Why the top rank of Nohalo-LBB may be undeserved. Private communication, Jan. 2013.

[143] Eric Brasseur. Gamma error in picture scaling. http://www.4p8.com/eric.brasseur/gamma.html, Aug. 2007. Revised Apr. 2012.

[144] Nicolas Robidoux. Don't use linear light when enlarging with Lanczos et al. http://www.imagemagick.org/discourse-server/viewtopic.php?f=1&t=21422, Jul. 2012. ImageMagick Forums post.

[145] Nicolas Robidoux. "Sigmoidal" minimization of resampling filter haloing. http://www.imagemagick.org/discourse-server/viewtopic.php?f=22&t=21415, Jul. 2012. ImageMagick Forums post.

[146] Nicolas Robidoux. Future of image enlargement? http://www.imagemagick.org/discourse-server/viewtopic.php?f=22&t=21435, Jul. 2012. ImageMagick Forums post.

[147] Adam Turcotte and Nicolas Robidoux. EXQUIRES GitHub code repository. https://github.com/aturcotte/exquires.git, 2012.

[148] Adam Turcotte and Nicolas Robidoux. EXQUIRES examples GitHub code repository. https://github.com/aturcotte/exquires/tree/master/exquires/examples, 2012.

[149] Adam Turcotte and Nicolas Robidoux. EXQUIRES online manual. `http://exquires.ca`, 2012.

[150] Aurélio C. Campilho and Mohamed S. Kamel, editors. *Image Analysis and Recognition, 5th International Conference, ICIAR 2008, Póvoa de Varzim, Portugal, June 25-27, 2008. Proceedings*, volume 5112 of *Lecture Notes in Computer Science*. Springer, 2008.