Metrics for measuring the impact of image processing algorithms on background statistics

Patrick J. Harding[a,b], Gordon Arthur[a], Neil M. Robertson[b]
[a]Thales Optronics Ltd. (United Kingdom);
[b]School of Engineering and Physical Sciences, Heriot-Watt Univ. (United Kingdom)

ABSTRACT

In surveillance and remote sensing applications images are often subject to processing both at the sensor and for display/storage. This paper presents a step towards measuring and understanding how these processes impact on human interpretation of regions of similar statistics within images. The paper describes the methods involved, including image generation, image processing algorithm application, and algorithm impact measurement techniques. A comparison is then made between the impact measurements and human observations. It is suggested that data on mathematical measures of the impact of processing algorithms on statistical regions of images may be usable for intelligent algorithm application.

Keywords: Image Processing, Psychophysics, Image Statistics.

1. INTRODUCTION

This study was motivated by a desire to understand how image processing algorithms impact upon regions of stable texture within a scene. In observing, for example, aerial imagery of countryside an observer will often intuitively pick-out regions of the image based on their texture and classify them as separable fields. Application of image processing algorithms to such an image, transforms the individual textural regions from their initial state to a new set of textures. These textures will be usually mathematically distinguishable. However, some of these process-resultant textures are difficult for an observer to distinguish, even though they result from the same original image. This implies that there exists a “threshold of application” beyond which an observer will find it difficult to distinguish between two processed images generated from the same original texture but using separate image processing parameters. By collecting information on textures under the influence of processing, it may be possible to apply algorithms intelligently using database information. For example, if two textural background regions of an image need to be kept distinct, the database could inform an algorithm about which possible algorithm parameter values would be suitable.

In reality, there may be many processed images generated using an image processing algorithm with different parameter inputs on the same original texture that an observer would group as “similar” or “the same”. Although an algorithm is capable of producing a number of mathematically distinct images equal to the number of parameters used, the number of textures generated that are distinguishable could be considerably less than this number.

In terms of the optimal application of the algorithm upon the texture region, it is worth knowing which parameter steps of the algorithm application will produce perceptually redundant image regions.

The segmentation of real images, such as aerial images, into regions of separate textures does not form a part of this study. Instead, a small set of “background” images is generated mathematically and analysis of this set is performed. The concept of “background” as opposed to foreground is worth considering in more detail here.

In performing a task, such as scanning an image for an object of interest that might or might not be present in the scene, the visual system of the observer is likely to pick out salient cues within the image that could be intrinsic or could be learned. Such cues would provide foreground information. In contrast to this, background information could simply be a region classifiable as being textured in some way.

While changing the content of a digital image using a processing algorithm can change an observer's impression of an image, this is not necessarily related to how the same observer might perform a "task" on the same image. (An example of a task is observer detection or higher discrimination of an object within a scene. This is not necessarily related to general quality perception[1].)

If trying to perform a task looking at a digital image, clearly the content around a task-critical object within the image is core to the performance levels of the task. If the object stands out clearly against the background, performance will improve. If the object blends very closely to its surroundings, task performance may be very difficult.

Since the nature of "background" is so difficult to define in a general sense, it is therefore worth examining isolated "background-like" cases under the influence of processing. Understanding of how isolated statistics act under processing might allow for a better understanding of how regions of differing statistics react to processing within real images gathered from sensors.

In the study, image processing algorithms are applied to statistically generated image textures, and image metrics are used to quantify the effect on the texture as a function of a key parameter of each algorithm. The image metrics are then related to observer perception of the effect on the texture. A knowledge of these factors may assist the intelligent application of an algorithm within different background regions of a scene.

1.1 Image generation

There are many methods for generating textural images. Furthermore the number of possible images that could be used is vast. Images are constructed using mathematical formulae and can be labeled using their generation parameters. For this study it was decided to create two classes of image - Gaussian and Autoregressive. Underlying both methods is the inclusion of pseudo-random number generation. It is necessary to chack that the method used for generating the pseudo-random number chains is sufficient to avoid repetition and therefore loss of randomness.

The Gaussian image generator was based on a random number generator weighted by a gaussian function. Pixel by pixel, the image was generated using this method. As a textural label the gaussian generator input was a desired mean of 128 with a standard deviation of 32.

The Autoregrssive image generator was based upon one described in "Texture Generation for use in Synthetic Scenes", page 38[2]. The texture is based on a random gaussian field (generated in the same manner as the gaussian above). The parameters specified for generation include a mean a standard deviation and two coefficients, a_1 and a_2. The parameters used to generate the random field were governed by the following formulae:

$$RandomMean = mean * (1 - a1 - a2 + a1 * a2)$$

$$Random\sigma = \sqrt{\sigma^2 * (1 - a2 * a2 - a1 * (a1 * (a2 * a2))^2}$$

These values were then used to generate a random gaussian field $R(i, j)$ where i, j are the pixel locations. The autoregressive texture is then grown upon these values with an initial top row and left column seed value of 1 using the following formula:

$$P(i, j) = a_1 * P(i, j-1) + a_2 * P(i-1, j) + a1 * a2 * P(i-1, j-1) + R(i, j)$$

This offers a texture that has some relationship between pixels over a certain distance depending upon the initial coefficients. Because the texture is grown from the seed values at the side it was necessary to carve-off the border regions of initial growth where the texture had not yet reached a recursive stable state.

The parameters for generation for the Autoregressive textures were a mean of 128 and a standard deviation of 32. It was decided to keep $a_1 = a_2$ for simplicity and to choose values 0.3, 0.35 and 0.4. These corresponded to increasing structure within the images.

Because of the random nature involved in generating the images it was necessary to generate sets of images of the same parameter specifications collect statistics of the results. Single examples of the original images generated are shown below. Figure 1 shows an example of a texture labeled "Gaussian" generated using a mean of 128 and a standard deviation of 32. Figure 2 shows examples of

"Autoregressive" images generated with an input mean of 128 and 32 but with different values of the a_1, a_2 autoregressive coefficients.

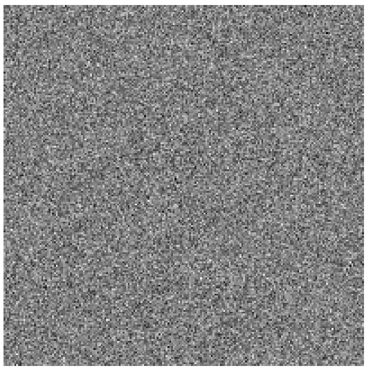
Figure 1. Example of a "Gaussian" image.

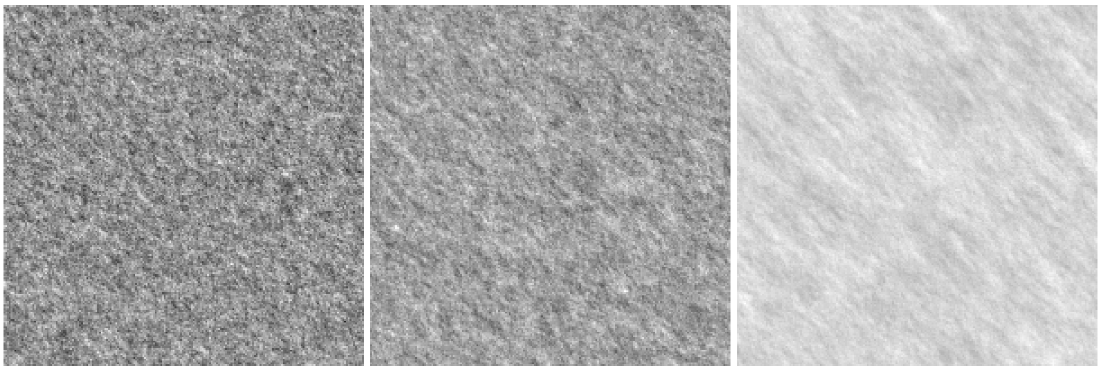
Figure 2. Three "Autoregressive" images. From left to right, $a_1=a_2=0.3$, $a_1=a_2=0.35$, $a_1=a_2=0.4$

The Gaussian image has no specific inter-pixel relationship. The Autoregressive images do have an inter-pixel relationship, the distance of which is governed by a_1 and a_2. This causes the appearance of structure within the images. The Gaussian case is to be considered a control for the experiment while the autoregressive images offer greater degrees of structure as a_1, a_2 increase. It is anticipated that the images should react distinctly and yet understandably to image processing algorithms.

1.2 Image processing algorithm set

In choosing the image processing algorithms set to apply to the image textures, it is necessary to consider at least two points. Firstly, it is necessary to choose algorithms that work on distinct mathematical principles so that any recorded effects can be easily related to the workings of the algorithms. Secondly, to keep the results space simple, the algorithms should have one primary parameter that will be altered, while any other possible parameters will be fixed. This will allow the evolution of impact measurements to be understood with a minimum of potential for confusion.

In light of these limitations, the algorithms chosen were, JPEG, Contrast Limited Adaptive Histogram Equalisation (CLAHE) and Unsharp Masking. A brief description of these algorithms follows:

JPEG is a compression algorithm that is based on the Discrete Cosine Transform (DCT)[3]. The algorithm divides an image into 8x8 blocks (with some adjustment at the boundary regions where this does not compute exactly). Each 8x8 block then has the DCT applied. The DCT elements are then divided on a per-element basis by a quantisation matrix derived from observer experiments to give good perceived image quality across an example set of images. Generally, this quantisation matrix is such that the high frequencies within an image will be cut-down to zero allowing for an efficient encoding of the information across the blocks of DCT information. The quantisation matrix is weighted by a "Quality Factor", Q, with a value of between 1 and 99. The Q_{50} level is such that the initial quantisation matrix is unchanged. Below Q_{50} the stripping of the high frequencies is more extreme and above Q_{50} the stripping of high frequencies is less so. In general therefore quality 1 is very poor, quality 50 is acceptable and quality 99 is high, although the relationship between "Quality Factor" and perceived image quality is non-linear. The JPEG algorithm is intrinsically lossy due to the DCT application and therefore there will always be a mathematical difference between the original and JPEG-processed image. The quality factor is the parameter for this study and is tuned between 1 and 99.

The CLAHE algorithm is a contrast enhancement algorithm. This algorithm works by tiling the image into a specified number of tiles and performing local contrast enhancement upon each tile. The local tile regions are then stitched together using bilinear interpolation in order to avoid the creation of artificial boundaries within the image. The contrast stretching can be limited to avoid damaging relatively homogenous areas due to noise amplification in a regular image. Since for this study the images are statistical and the desire is to see how the algorithm transforms a non-featured texture into another, the contrast-limiting threshold is left on default. The parameter for this algorithm is the number of tiles, $NTiles$ along each axis of the image and is tuned between 2 and 40. For simplicity, the condition, $NTiles(x) = NTiles(y)$, is kept. Note that the total number of tiles imposed upon the image is the square of this number. For parameter $NTiles$ in this study, the total number of tiles over the image is therefore $Totaltiles = (NTiles)^2$.

Unsharp Masking is actually the name of a sharpening process. Its name is derived from the process of taking a blurred version of an image from itself, to create a sharpened image. In this study an “unsharp” Laplacian filter is created that takes the following form:

$$Unsharp = \frac{1}{(1+\alpha)}\begin{pmatrix} -\alpha & \alpha-1 & -\alpha \\ \alpha-1 & \alpha+5 & \alpha-1 \\ -\alpha & \alpha-1 & -\alpha \end{pmatrix}$$

where α is the degree of sharpening and varies between 0 and 1. This unsharp filter is then convolved with the original image to give the sharpened image. In this study, the parameter is α and the values it takes are between 0 and 1 in intervals of 0.01

1.3 Impact metrics

Of the total possible number of metrics which could be used to assess the impact of an image processing algorithm, it has been decided to analyse a small set for the purposes of this study. This is possible because the images are expected to be statistically classifiable by design. That is, due to there being high similarity between generated images of the same input parameters by design, if a set of such images is processed then the output image set should hold similar statistics.

The metrics chosen were the mean, the variance, the skew the kurtosis the Peak Signal to Noise Ratio (PSNR) the correlation coefficient and the Structural Similarity Index (SSIM_index). These statistical metrics are very basic for the most part, but applying them to images that are relatively homogenous (or statistically well-classified) does make sense. The mean and variance are well-known but a review of the other metrics and why they may be appropriate follows.

Skew is a measure based on the 3rd moment of the data. It a measure of how the data is distributed between halves of the curve. In a Gaussian distribution the skew of an image is zero. If the skew is negative the distribution of data is weighted towards the left of the curve. Similarly, if the skew is positive the distribution of data is weighted to the right.

Kurtosis is a measure based on the 4th moment of the data. It measures how the data is distributed between the tails and peak of the curve. In a Gaussian distribution the kurtosis is zero. If the kurtosis is positive the distribution of data is weighted towards having more data in the tails of the curve than in the gaussian case. Similarly a negative kurtosis implies that more data is in the center than is out in the tails.

For the mean, variance, skew and kurtosis the original value is calculated and is compared to the value calculated in the processed image.

PSNR is an intrinsic measure of comparison between two images. In this case the measure is that between original and processed images. In this implementation of the PSNR the difference is taken between the images and the mean of the square error is calculated. The PSNR is then calculated in decibels using the following formula:

$$PSNR = 20 * \log_{10}(255/\sqrt{MSE})$$

The correlation coefficient is calculated using the following formula

$$CorrelationCoefficient = \frac{(N * Cov(A,B))^2}{(N * \mathrm{var}(A)) * (N * \mathrm{var}(B))}$$

where Cov is the covariance between data A and B and N is the number of data points. It is a measure of how similar the data sets are and outputs a value between –1 and 1 where –1 is strongly negative correlation, 0 is no correlation and 1 is strong positive correlation.

SSIM_index is a measure of the structural similarity between two images. It is fully described by Z. Wang[4]. It attempts to separate the components of the images into luminance, chrominance and structure. It acts to take local statistics into account by filtering the image with a local window function (default Gaussian). By subtracting the non-structural components of the images by approximating the luminance to the mean, and the chrominance to the standard deviation a value of the structural difference is then calculated using the following measure

$$SSIM\_map = \frac{((2 * m_{12} + C1) * (2 * \sigma_{12} + C2))}{((m_1{}^2 + m_2{}^2 + C1) * (\sigma_1{}^2 + \sigma_2{}^2 + C2))}$$

(in above and following, all multiplication is done element by element rather than by matrix)

where m_1= filter outcome of window and image1, m_2= filter outcome of window and image2, $m_{12} = m_1 \times m_2$ and σ_1= filter outcome of window and image1, σ_2 = filter outcome of window and image2, $\sigma_{12} = \sigma_1 \times \sigma_2$, C1 & C2 are small constants to avoid division by zero. The index is the mean value of the map output.

2. DATA COLLECTION

2.1 Computational Data Collection

The experimental procedure was as follows. For each set of image generation parameters and a particular algorithm,

1) An image was generated and the appropriate statistical metrics of the original image were collected.

2) The image was processed using the chosen algorithm and with a particular parameter. The appropriate statistics of this single image were collected and the difference metrics were calculated between the original and processed images.

3) The algorithm parameter was increased by one step and stage two above was repeated until the parameter steps were exhausted

Because of the intrinsically random nature of the image generation, the above set of steps was run three times for each set of image generation parameters to gather group statistics. It was found that for the textures involved in this study repetition beyond three did not adequately compensate the necessary outlay of extra computational time by offering a substantial reduction in error.

The overall process was carried out for the three algorithms, across the desired parameter spaces using the specified four textures, repeated three times. For each image processing algorithm and each texture-class defined by its parameter space, the out put was a set of 7 charts (one for each processing metric) showing a single metric with error bars vs. image processing parameter. The output was therefore 12 sets (3 algorithms applied to 4 image classes) of 7 charts. This information can be used to see how well each metric used traces human perception of the impact of the algorithms for different image processing algorithm parameter inputs.

2.2 Collection of Observer data

As described in the introductory section, an aim of the study is to assess the correlation of the mathematical metrics with observer perception (‘impression’) of the effect of an algorithm on a texture.

Therefore the observer experiments need to provide data that indicates which degrees of processing acting on a base texture are reliably perceptibly different or otherwise to a set of observers. This data can then be compared to the mathematical metric data.

The basic question posed to an observer is therefore: can a given pair of texture images from a data set be distinguished? For this test the data set should include the original image and other images produced through the processing of the original image by a particular algorithm but using different parameters. The full set of algorithm parameters will give the following possible sets of image pairs, $\{P(A_i) or O\}$ vs. $\{P(A_j) or O\}$, where O is the original image in the set, P is an image processed version of O processed by algorithm A, and subscripts i, j denote the set of allowed parameter values in the set. Let i be the image parameter that is under test and let j denote the label of the parameter being compared to i. The condition $i = j$ and the display of O vs. O pairs is allowed. To build up observer information, for a fixed $\{P(A_i) or O\}$ observers would be asked to judge whether the textures $\{P(A_j) or O\}$ and $\{P(A_j) or O\}$ are the same, and record the response, say 1 for yes and 0 for no. After stepping through all parameters there would be a function mapping how similar the observer found $\{P(A_i) or O\}$ compared to all other textures in the set. Over a set of observers, this data becomes more credible as the observers performing the same test question will likely tend to respond in a statistically similar fashion that can be regarded as a trend. Taking the data as a Bernoulli process and therefore assuming the number of correct responses given by the observers for given input i to be the sum of random samples with a probability of success p_i allows the textures to be grouped perceptually. A psy-function of algorithmic-texture perception can be built up, specifying the relationship between the underlying probability of a correct response p and the input stimulus intensity. By calculating a psy-function for each value allowed in the set $\{P(A_i) or O\}$ and then summing them, it is possible to get an overall function that displays how similar the set of observers found all other textures relative to a point. This psy-function may then be compared to the mathematical metrics.

After some initial consideration, it was decided to constrain i, j to a subset of the allowed parameters. This was done firstly because upon inspection of the processed images, there were some quite subtle changes at full parameter resolution for many texture/algorithm combinations indicating that full resolution was unnecessary and secondly because this would limit the number of time-consuming observer tests required. In all of the examples, the generated textures used are the four described above.

For CLAHE, the Ntiles values for $\{P(A_i) or O\}$ were taken from the set {*O*, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40} (where *O* is the original, unprocessed image). The comparison values $\{P(A_j) or O\}$ were taken from the set { *O*, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40}. Therefore for each observer 11 psy-functions were generated at a resolution of 2-Ntiles.

For JPEG, the Quality Factor values for $\{P(A_i) or O\}$ were taken from the set {10, 20, 30, 40, 50, 60, 70, 80, 90, *O*} (where *O* is the original, unprocessed image). The comparison values $\{P(A_j) or O\}$ were taken from the set { 5 ,10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, *O*}. Therefore for each observer 10 psy-functions were generated at a resolution of 5-Quality Factor.

For unsharp masking because the difference in output for even quite large parameter step differences was obviously very small, it was decided to take the alpha values for both $\{P(A_i) or O\}$ and $\{P(A_j) or O\}$ were taken from the set {0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, *O*}. Therefore for each observer 12 psy-functions were generated at a resolution of 0.1 alpha.

The appropriate image pairs from the above sets were combined with a black separator-space to form a single image of test pairs. The test pairs were based upon the same original image. A random element ensured that for such tests $\{P(A_i) or O\}$ had an equal chance of appearing on the left or right hand side of the pair image. It was these pair-images that formed the basis of the testing on the observers.

A willing observer set was recruited, consisting of 10 observers. The ages of the group varied between 20 and 58 with an average of 42 years. They all claimed to have normal, or corrected to normal vision. Before testing began, the aims of the experiment were explained to the observers, and were shown example videos of an image being tuned through the parameters of an image processing algorithm.

The image pairs were displayed in a random order for each required value of $\{P(A_i) or O\}$. There was no time limit imposed upon the observer, although observers were advised to try to spend less than four seconds on each image pair. The viewer distance was set at around 60 cm and the screen resolution was adjusted to display the images suitably. The observers were asked to respond to the image pair stimuli by inputting to computer by pushing a particular button whether they perceived the images as “the same” or as “different” This is subtly different from asking if they are “identical” which is likely to elicit a more conservative response. Their responses to the question were saved out to disk. Depending on the algorithm involved, the observers were given a short break approximately every 200 images to reduce fatigue on the observer. Lapse errors (errors that are independent of stimuli, such as pushing the wrong button for no apparent reason[5]) are always present in observer experiments although fatigue can increase their rate of occurrence. A crude estimate of lapse rate was made by collecting the statistics of the error-rate for image pairs of matching images. This is a crude estimate because this depend on the context of the surrounding processed image set – if there is a large number of perceptually-similar images around the “identical” test pair then that will make it harder to easily identify the textures as the same than if the neighboring textures were all easily-distinguishable. The guess rate should also be estimated – this is the rate at which it is generally possible for an observer to input the right answer to the test even if he ignores the content of the test. The overall lapse rate was approximately 5% using the estimate above. The guess rate varies with the number of tests in a block compared to the number of actual correct responses. This rate is between 5% and 10% depending on the algorithm combinations used above. In general a psy-result needs to be above this notional guess-rate to have some significance.

The recorded set of observations for each value of $\{P(A_i) or O\}$ compared to all $\{P(A_j) or O\}$ were averaged across the observer set to give the psy-function for each $\{P(A_i) or O\}$. For each algorithm, the averaged psy-functions for each parameter point were summed to give a master psy-function of the perception of the textures generated by the algorithm. Summing all psy-functions across the parameter space will creates an average of how similar each point is to the other points along the axis. In this way the summed psy-functions will provide a perceptual grouping map that can be compared with the mathematical statistics.

2.3 Computational vs. observer data

The computational and observer data are displayed in figures below. The data is displayed in a format that is convenient for the paper. What is of importance is recognizing the trends and analyzing whether the data is significant, rather than worrying too much about the values of the numerical data. (Such numerical data could change very quickly for different textures. Only the Correlation Coefficient, the SSIM and the PSY functions have been constrained in the y-axis to an interval of [0,1] and the rest are auto-scaled. Following is a little explanation of the results and the trends encountered. The figures contain a lot of information and it is necessary to read the captions to navigate them. See figure 3 for the CLAHE algorithm, figure 4 for the JPEG algorithm and figure 5 for the unsharp algorithm.

Looking down the columns of figure 3, which shows the results for CLAHE, it is interesting to note that the mathematical metrics correspond reasonably well to the shape of the psy-function. The original texture stands out from all the others. The psy-functions generally possess a long grouping of perceptually similar textures between roughly 2 and 25 NTiles and then the perception of the images changes as NTiles approaches 40. The form of the PSNR is very similar to that of the Correlation Coefficient and SSIM index when viewed over a local area of their respective y-axes. The skewness and kurtosis are small, but statistically significant within the context of this study. These graphs are relatable to the psy-function – they have a straight-line growth in value up until the same region and then change more suddenly. Recalling that the metrics were calculated as an average of measures of three textures generated with the same parameters, it is clear that the output of CLAHE remains stable to textures generated within our mathematical constraints. In spite of this general similarity, there are clearly areas where there is difference, most notably for the value of 40 tiles. At forty tiles in the Gaussian case there is a sudden change in perception, which is also matched by many of the metrics. In the Autoregressive cases, the observers did not observe such a phenomenon, although the metrics for the Autoregressive continue to pick-up the same change as in the Gaussian case. Obviously, there are numerous other exceptions. Looking across the rows, it might be possible to distinguish the textures on the strength of the computational metrics alone, but probably not to label them very accurately.

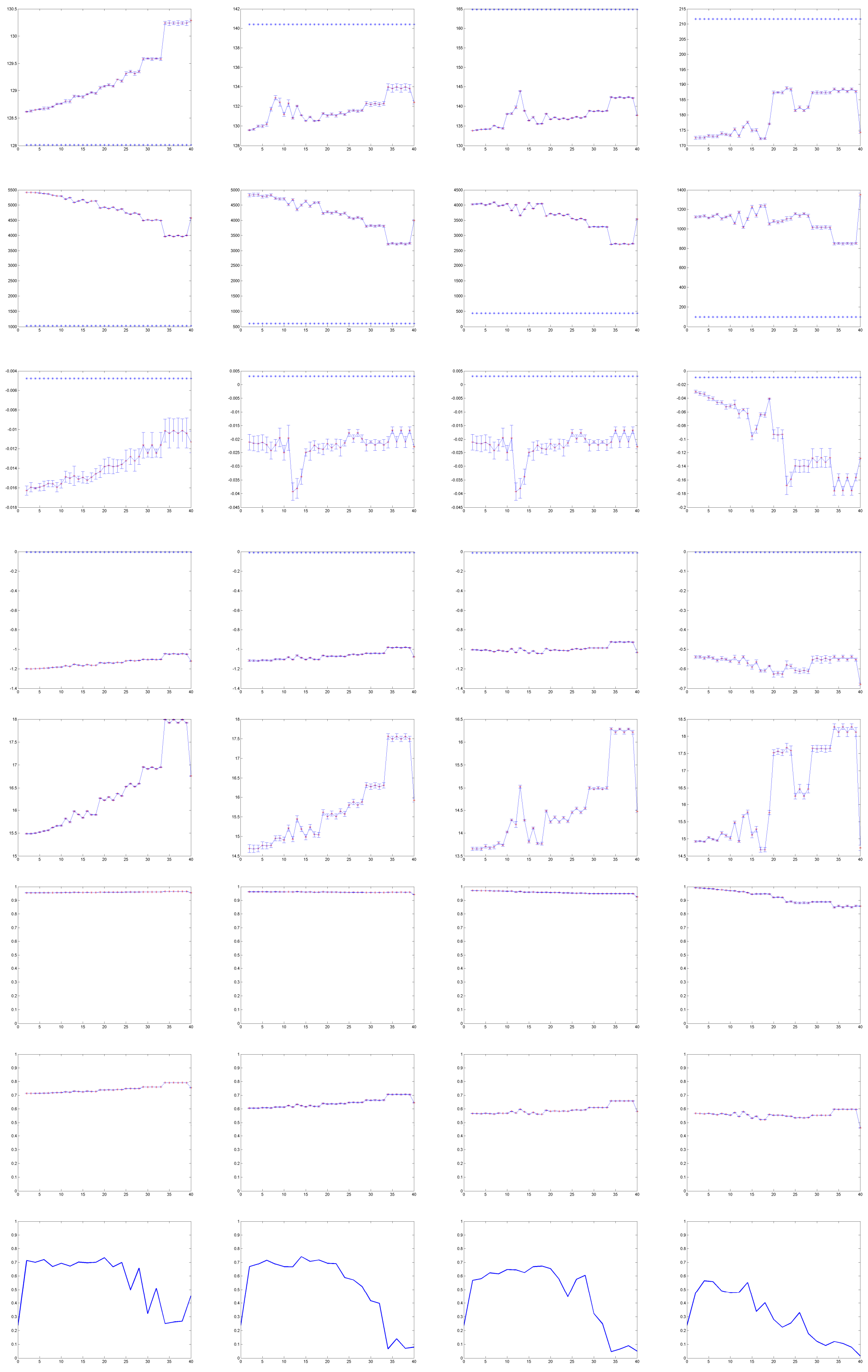

Figure 3. Graphs showing the evolution of the metrics as CLAHE algorithm is tuned through parameter NTiles. The value of NTiles is between 2 and 40 and lies along the x-axis of every chart. The ***columns*** are ordered into texture classes: from left to right, Gaussian, Autoregressive (a_1=a_2=0.3) , Autoregressive (a_1=a_2=0.35), Autoregressive (a_1=a_2=0.4). Each ***row*** represents a distinct metric. From top to bottom these are: mean, variance, skew, kurtosis, PSNR, Correlation Coefficient, SSIM. The first four rows have the metric of the original image drawn across the graph. The final row shows the psy-functions derived from the observer experiments. Psy-measures on the original image have been labeled with NTiles = 1.

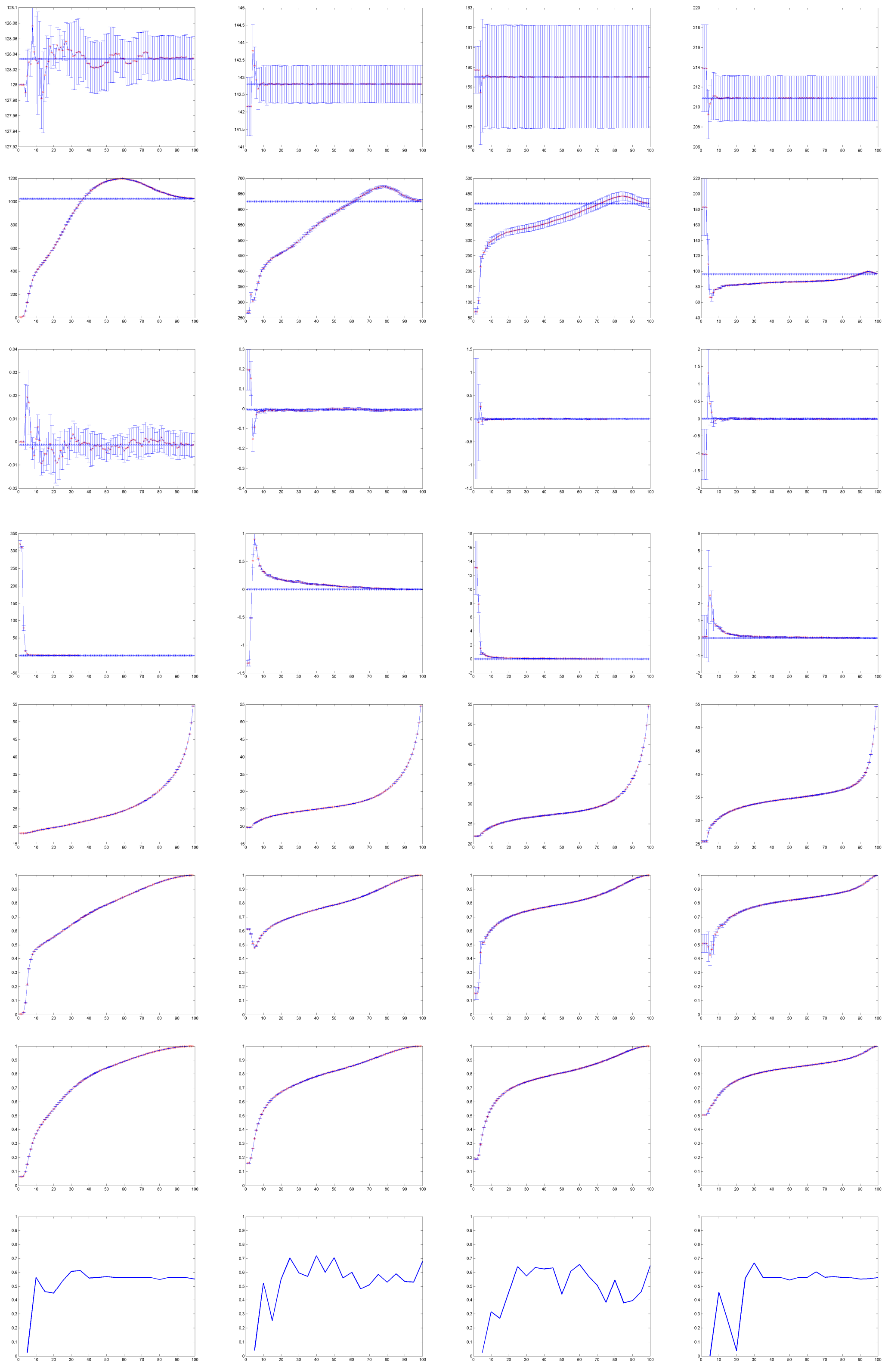

Figure 4. Graphs showing the evolution of the metrics as JPEG algorithm is tuned through parameter Quality Factor. The value of Quality Factor is between 1 and 99 and lies along the x-axis of every chart. The ***columns*** are ordered into texture classes: from left to right, Gaussian, Autoregressive (a_1=a_2=0.3) , Autoregressive (a_1=a_2=0.35), Autoregressive (a_1=a_2=0.4). Each ***row*** represents a distinct metric. From top to bottom these are: mean, variance, skew, kurtosis, PSNR, Correlation Coefficient, SSIM. The first four rows have the metric of the original image drawn across the graph. The final row shows the psy-functions derived from the observer experiments. Psy-measures on the original image have been labeled with Quality Factor = 100.

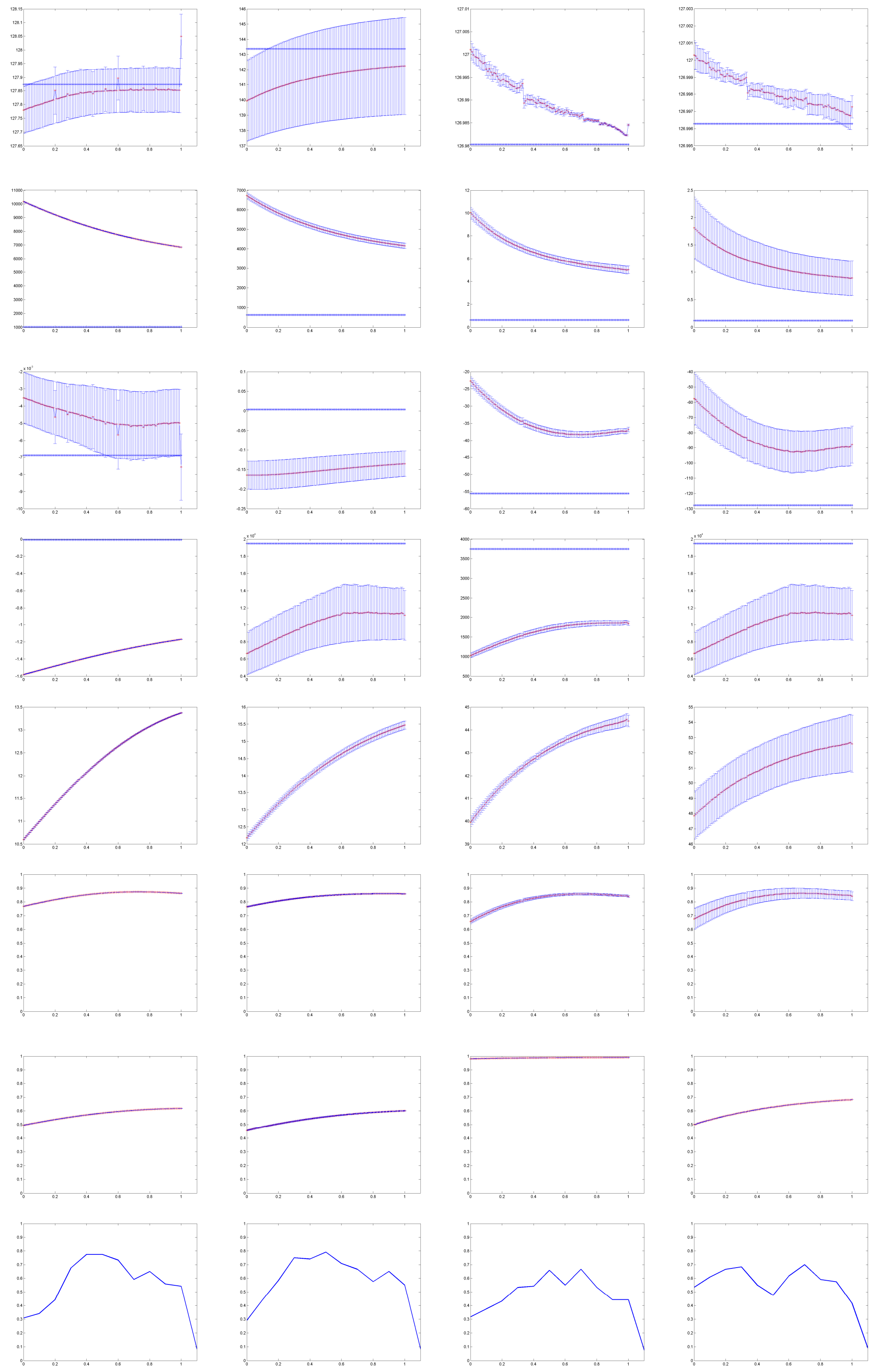

Figure 5. Graphs showing the evolution of the metrics as unsharp masking algorithm is tuned through parameter alpha. The value of alpha is between 0 and 1 and lies along the x-axis of every chart. The ***columns*** are ordered into texture classes: from left to right, Gaussian, Autoregressive (a_1=a_2=0.3), Autoregressive (a_1=a_2=0.35), Autoregressive (a_1=a_2=0.4). Each ***row*** represents a distinct metric. From top to bottom these are: mean, variance, skew, kurtosis, PSNR, Correlation Coefficient, SSIM. The first four rows have the metric of the original image drawn across the graph. The final row shows the psy-functions derived from the observer experiments. The value 1.1 at the far right of the graphs represents the values of the original images.

Considering figure 4, JPEG. The mean is not significant, and the skewness and kurtosis tend to zero quickly. As for CLAHE above the metrics show that the texture-generation is stable when the algorithm is applied. The psy-functions show that, as expected from the design of the algorithm, the observers found images above Quality Factor 40 difficult to distinguish, including the original image. Looking down the columns of figure 4, again it is easy to see that there is an intuitive correspondence between many of the metrics and the psy-functions. The Correlation Coefficient and the SSIM appear to match up well. The PSNR is a poor match. There is an obvious dip in the Autoregressive (a_1=a_2=0.4). case at around Quality Factor 20 which is not detected by the metrics.

In figure 5, unsharp masking, the original image is very different from the processed images in most cases. Again, overall the texture is well-categorised by its generation parameter and remains stable under processing. The mean value is not a significant metric and the skewness is small. The kurtosis reduces as the autoregressive constant increases. Many of the metrics have a similar umbrella-shape to the psy-function but it seems harder to associate the metrics to the Psy-functions.

Conclusion

As a step towards intelligent algorithm application we generated image-sets of known statistics and processed them using mathematically distinct algorithms over a range of possible parameters. For each image processing parameter used, we collected basic statistics of the generated image. We next carried out observer experiments to determine a psy-function describing how an observer might perceptually group neighboring images. It was observed that the metrics had a clear similarity in outlook to the psy-functions across the examined parameter spaces for the algorithms in many cases. Using a database of how various textures behave under processing might allow for the intelligent application of algorithms on certain classes of images, such as aerial imagery or certain medical applications, which are heavily texturally dependent – either by applying limits on the parameters of an algorithm or by selecting the algorithm required to separate a texture into visually-distinct areas automatically. Despite the fact that the metrics used were relatively unsophisticated and the number of textures small, it seems possible that the information yielded from this experiment could be used in this fashion to get a limited general result. A necessary next step for this research would be the expansion of the method to include more sophisticated texture classification schemes, such as wavelets, and the introduction of a wider set of original textures.

[1] Dixon, T.D., Fernández-Canga, E., et al. (submitted July 2007). “Selection of image fusion quality measures: Objective, subjective and metric assessment”. Journal of the Optical Society of America A.
[2] Bleiweiss M.P., Ota C.Z and Rollins J.M., “Texture Generation for use in Synthetic Scenes”, Army Research Laboratory Publication ARL-TR-1015, March 1997.
[3] The JPEG standard is published as - ISO/IEC IS 10918-1/ITU-T Recommendation T.81
[4] 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. 4, Apr. 2004.
[5] Wichmann, F. A. & Hill, N. J. (2001a): *The psychometric function: I. Fitting, sampling and goodness-of-fit.* Perception and Psychophysics 63(8), 1293-1313.