Region-based method for visually lossless image compression

Lenko Grigorov, grigorov@cs.queensu.ca

CISC 857 Course Project

Abstract:
This report proposes a new method for image compression. The compressed images are visually identical to the originals, even though some of the information in the images is discarded. This is achieves by using results in the field of Human Vision Modeling to select the visually significant regions from the images. Two different compression methods are applied, depending on whether the compressed region is visually significant or not. The method is applicable to all types of images RGB images – photographs, diagrams, text, and others. A reference implementation is developed and tested on a variety of examples. For most images the file size is reduced by at least a half.

1. Introduction

Digital images are widely used to store image data, since they can be easily manipulated with the most versatile tool – the computer. Unfortunately, uncompressed digital images occupy a lot of information space, especially when they consist of multiple channels (bands). Users appreciate the availability of a number of different compression methods, which reduce the burden on the computer resources. The compression methods can be separated into two categories: lossless and lossy. The lossless compression methods compact the image data using fully reversible transformations (usually based on probabilistic modeling) and give the users the opportunity to reconstruct the original image completely. Unfortunately, these methods generally do not achieve the desired level of reduction of the size of the images. The lossy methods perform much better, however, they discard information during the process of compression. The original image cannot be fully reconstructed from the compressed data. Clearly, users would prefer to have a very high level of compression while preserving the original data, so neither type of compression method is perfect.

In the case when the digital images will be viewed on the computer screen, one could define the quality criterion as "how much the image appears close to the original". If the image appears identical to the original, this satisfies the highest requirement. One could also consider the fact that not all information present in a digital image would influence the appearance of the image on the screen. Thus by eliminating this information, one would be able to achieve better compression, while the image will continue to appear identical to the original.

The most widely used compression methods (like JPEG) strive to achieve minimal loss of information based on mathematical measurements of difference. The particularities of the human vision system are often not taken into account and not used to the advantage. In certain cases this might not only degrade the image quality noticeably, but also introduce unwanted artifacts. In this work I am exploring the possibilities, which a different approach could bring. I propose a new method for image compression, which is developed in accordance to the properties of human vision. The images reconstructed from the compressed are visually identical to the originals. This is true for all types of images – including photographic pictures and pictures of diagrams and text.

Previous research in the area is summarized very well in [1], where novel ideas are proposed as well. The human vision model is examined in [4] and [5]. [2] serves as a major backgrounder in the area of general image compression. Work with arbitrarily shaped image regions is discussed in both [3] and [6], and [3] proposes a very simple and versatile method for arbitrarily shaped region traversal. [1] uses the human vision model to enhance the JPEG2000 compression standard and achieve outstanding compression results. It served as a great inspiration, however, the goal of the compression method proposed here is to achieve visually lossless compression and thus undertakes a completely different path.

The region-based method compression method works as follows:

1. The input image (in RGB) is converted to the YCbCr color space to decorrelate the image bands.
2. Each band is divided into regions using the properties of the human vision model. The regions containing visually significant information are marked "white" and the rest of the image is marked "black".
3. The white regions are compressed using compression which is appropriate for the response of the human eye to such regions.
4. The black regions are compressed using low quality compression, which does not introduce noticeable artifacts.
5. The final output is compacted further by using the GZIP universal compression method.

In section 2 of this report I introduce the Human Vision Model and explain the YCbCr color model. In section 3 I discuss the methods I use to separate visually significant regions from the image. Section 4 and 5 describe the compressions methods used for the two types of image regions. Section 6 presents the results achieved with a reference implementation and a short conclusion is provided in section 7. Appendix A contains a short manual on how the included software operates.

2. Human Vision Model

The properties of the human vision have been examined since ancient times. Scientists have been trying to create a model, describing people's sight. The contemporary results, even though not yet decisive, are well summarized in [4] and similar works. The human color vision is a very complex process. The physical preceptors of light in the eye respond to the wavelengths of the red, green and blue light. However, the signals are combined before they reach the brain. The signals which are received by the brain can be described as "what is the brightness", "how blue or how yellow it is", and "how red or how green it is". This also explains why humans do not perceive colors such as "yellow-bluish" or "green-reddish". The eye relays the information in the visual field as changes in these signals over distance in the plane of vision. The acuity towards the edges of the visual plane decreases, with color vision decreasing more rapidly.

We are interested in the properties, which relate to the visual inspection of an image on the screen. Will there be changes of the image signal, which will not be noticed by the viewer? A widely used method to measure the sensitivity of the eye is the use of special test images constructed as follows. Stripes of alternating brightness or color are fit together (see Fig. 1). On one of the axes the frequency of the stripes per distance unit increases exponentially. On the other axis, the difference (of brightness or color) between neighboring stripes decreases exponentially. A subject looking at the test images would be able to tell immediately that below a certain level of difference between the neighboring stripes, no changes are perceived (the neighboring stripes appear as a uniform background). Furthermore, this level varies with the frequency of stripes. Thus a function describing this behavior can be drawn (see Fig. 1). Another fact to notice is that this threshold varies with the overall brightness of the surrounding. I.e. the value is greatest for medium-brightness surrounding; with the increase or decrease of the overall brightness of the surrounding the eye is less sensitive to changes. This allows us to define a 3D threshold function, which depends on the spatial frequency of changes, the relative difference of the changes (delta), and the overall brightness of the surrounding.

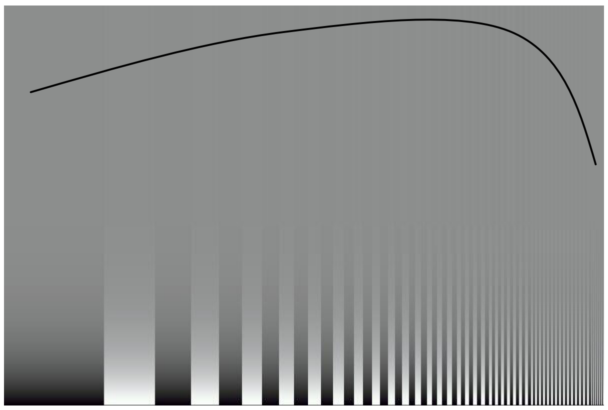


Figure 1 Perception test picture with threshold function

Unfortunately, researches haven't performed enough experiments to map this function, partly because it could be of interest to an area very different from Psychology and Physiology. In [1] a not very encompassing experiment is described. Since this function is of particular interest for the region-based method for image compression (as discussed in section 3), I decided to make some measurements to obtain usable values, even though they might be very inaccurate.

Before I describe these measurements in detail, I will discuss the YCbCr color model. As already mentioned, the human brain receives three signals from the eye – brightness, blueness/yellowness (blue chrominance), and redness/greenness (red chrominance). These signals combine to create the human vision color model. Unfortunately, due to its complexity and the lack of extensive measurements, this model is not directly applicable for the description of digital images. The most widely used color model for digital images is the RGB (red, green, blue) model, because it directly maps to the way computer screens display images. Unfortunately, it is not suitable for experiments examining human vision. Among the standards defined by [7] is the YCbCr color model (brightness, blue chrominance, red chrominance), which approximates much closer how humans see. Its relation to the human color vision model is discussed in greater detail in [1]. One of the most important facts to note is that this model is very suitable for compression purposes – it achieves very good decorrelation of the image bands (i.e. the value of a band cannot be predicted from the values of the other bands). This is the reason why this model is used for television broadcasting and for the highly efficient JPEG2000 compression standard. This is also the reason why the YCbCr model was selected for use with the compression method proposed in this work. An example of the YCbCr bands of an can be seen on Fig. 2.

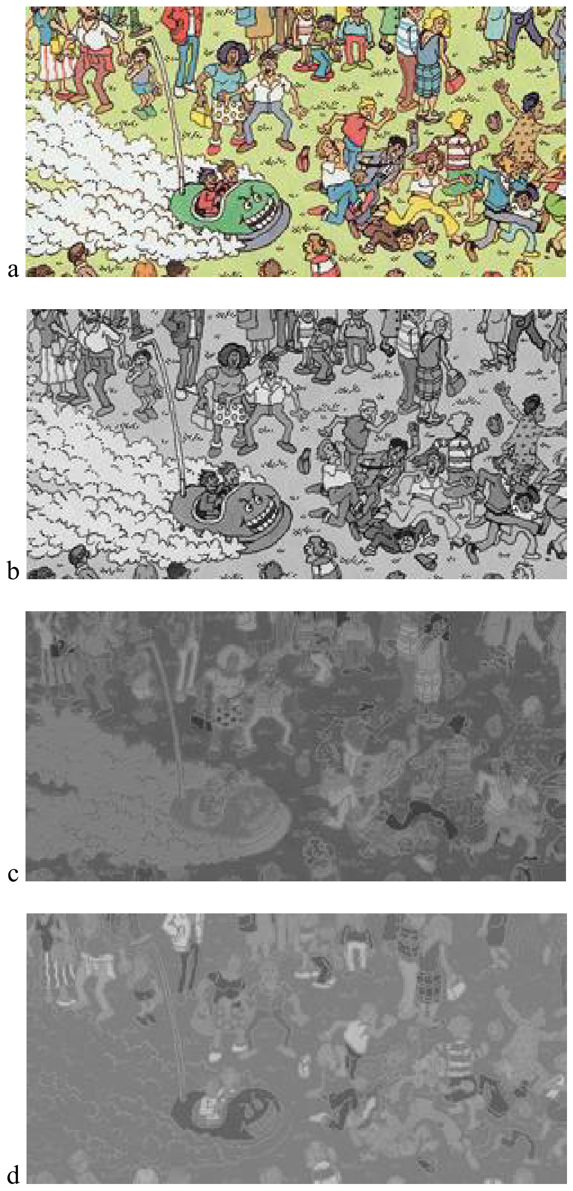


Figure 2 The YCbCr color model. a) Original image; b) brightness band Y; c) blue chrominance band Cb; d) red chrominance band Cr

My attempt was to map the 3D sensitivity threshold function, which was described earlier. Test images (see Fig. 3) were algorithmically generated. Images were generated for the Y, the Cb, and the Cr bands. For each band, three images were generated – for bright surrounding, medium brightness surrounding, and for dark surrounding. Each of the nine images was examined when the stripes were vertical, horizontal, and diagonal (at 45°). The threshold values for each image were mapped and then approximated using linear functions. Since the low resolution and color capability of the screen, on the medium brightness images usually the threshold was not met. The missing region was approximated from the curves of the images at the other brightness settings.

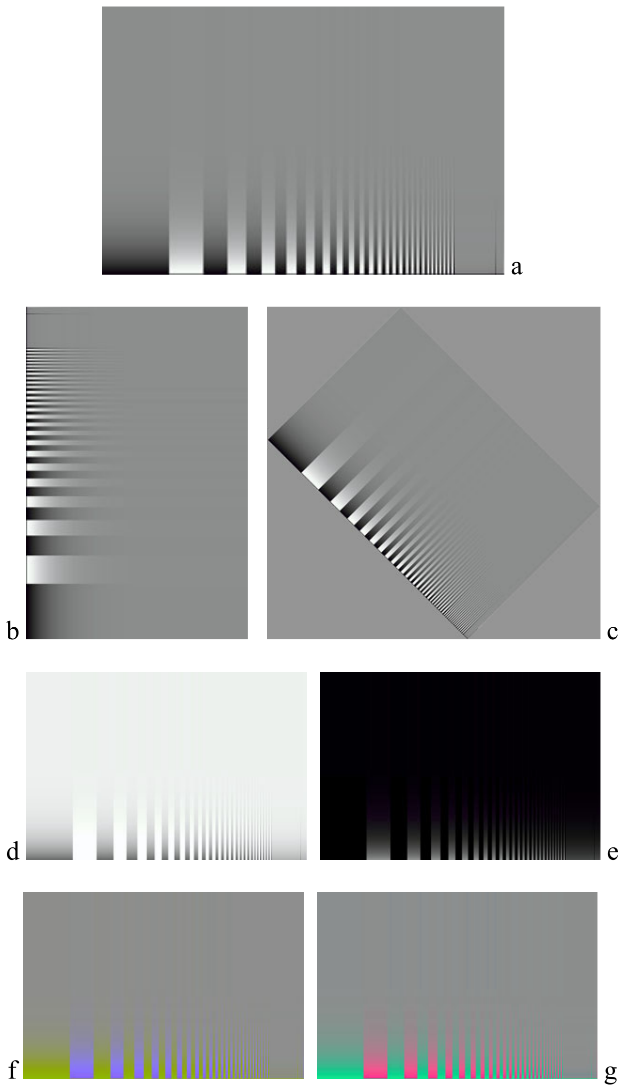


Figure 3 Exaples of perception test pictures. Rotations and changes to overall brightness were used with all bands. a) Y band, medium brightness; b,c) Y band, horizontal and diagonal stripes; d,e) Y band, increased and decreased overall brightness; f) Cb band; g) Cr band

Even though the threshold curves for the vertical, horizontal, and diagonal stripes were not identical, they were considered similar enough and a single threshold curve was used for all rotations. All approximations are very imprecise, however, the goal was to get some model, which can be immediately used, while accurate measurements can be taken later and incorporated in the software at a later stage. The results are shown in the following box.

Technical

Function used to generate the test images (600×400):
width of stripes = $100/1.95^{X/100}$
delta of stripes = $256/1.1^{Y/3.5}$

Linear approximations of threshold function (in terms of x,y from the test image):

Y band

$F_Y = \alpha y$, where
$y = -0.91x+881$ if $x>414$
$y = 0.22x+413$ otherwise and
$\alpha = (1.02\beta+282)/413$ if $\beta<128$
$\alpha = (-1.93\beta+660)/413$ otherwise

Cb band

$F_{Cb} = \alpha y$, where
$y = -0.39x+493$ if $x> 393$
$y = -0.09x+374$ otherwise and
$\alpha = (0.97\beta+250)/374$ if $\beta<128$
$\alpha = (-0.59\beta+450)/374$ otherwise

Cr band

$F_{Cr} = \alpha y$, where
$y = -0.2x+489$ if $x> 415$
$y = -0.11x+452$ otherwise and
$\alpha = (1.93\beta+205)/452$ if $\beta<128$
$\alpha = (-1.93\beta+699)/452$ otherwise

where β is the average brightness of the surrounding on the scale from 0 to 255

All measurements were performed using a high-quality 17” Optiquest V775 monitor at the resolution 1024×768. The correct ICC color profile for the monitor was used and the brightness and contrast settings were adjusted according to the Adobe Gamma correction tool. Unfortunately, the results remain only referential, because of the limited conditions examined (no other screen resolutions) and because I was the only test subject.

3. Selection of image regions

Having approximated the threshold function as described in the previous section, it can be applied to separate the regions of the image where the human eye would be able to perceive noticeable changes. There are three threshold functions, for the Y, Cb, and Cr bands respectively, so they can be separately applied to the corresponding bands. Since the threshold function used is the only difference, the selection of regions is described for a single band only.

The image (band) is divided into two types of regions. The white regions describe pixels with visually significant information – such as sharp edges. These regions are governed by the area beyond the threshold function, which simply means that the observer will perceive a sharp change in the image (be it change of brightness or change of chrominance). The black regions describe the rest of the image. They are governed by the area below the perception threshold, which simply means that the observer might notice changes, but they will be very gradual and appear blurry. See Fig. 4 for an example of the regions selected for an image. Clearly, the black regions can be compressed with loss of information while they would still appear identical to the original.

Why would one use only two types of regions? The sensitivity of the human eye is different to different amounts of changes in the image, so one could possibly use different gray levels to smooth-out the sharp transition defined by the threshold function and thus more closely represent the observer’s perception. Considering however, that the regions are created for the purposes of image compression, the main goal should be the compactness of information. The regions can be of arbitrary shape and thus cannot be described satisfactorily via simple geometric constructs. Since the decompression method needs to know which pixels of the image belong to what type of region, a mapping of the regions has to be included in the compressed stream.

A bitmap of the image, where white pixels stand for white regions and black pixels stand for black regions is the most compact way to do this. Furthermore, the definition of the threshold function lends itself to the bi-polar separation.

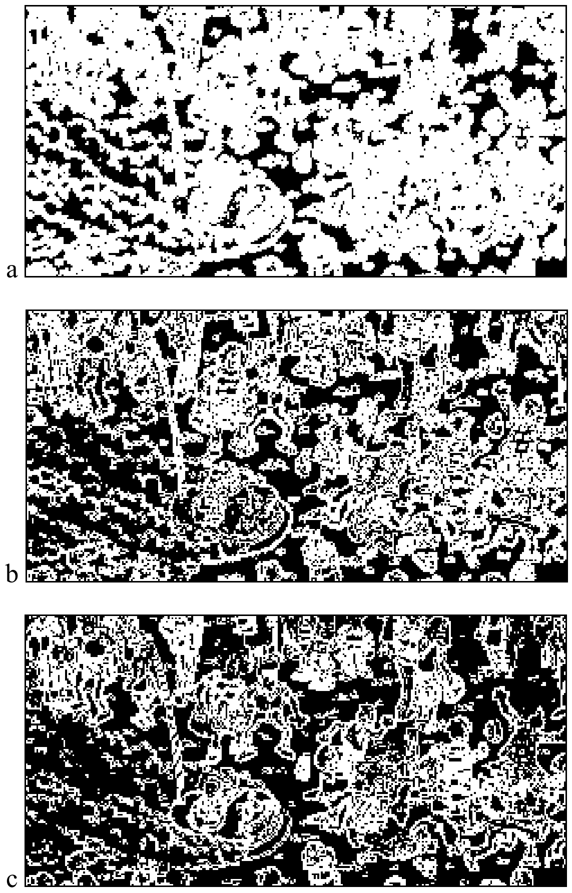
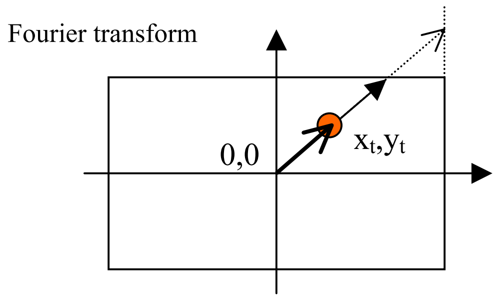


Figure 4 Black and white region selection. a) Y band; b) Cb band; c) Cr band; d) original image

The approach used for the division of the image into black and white regions is based on the gradient of the image (band). Indeed, the higher the gradient, the bigger is the delta of the relative change in the image and thus the stronger the visual perception of the change. So applying a threshold to the gradient would immediately result in the creation of white and black regions, according to our needs. As was discussed earlier, however, the threshold value for the human vision model depends not only on the delta (gradient), but also on the spatial frequency of changes, the average brightness of the surrounding, and on the orientation of the changes (the latter is not modeled). Thus it is needed to adjust the gradient according to the other criteria before applying the thresholding.

The adjustment according to the spatial frequency is executed on the Fourier transform of the image (band). This transform provides a very convenient method for

Technical



Calculation of the *x* coordinate for the perception test image from the position in the Fourier transform (x_t,y_t).

The $\longrightarrow$ vector passes through all spatial frequencies in the image in the given direction. At the tip of the vector are the highest frequencies (with wavelengths of two pixels). At the origin are the frequencies with wavelengths of the extension of the whole image in this direction. Thus by obtaining the ratio between the lengths of the $\rightarrow$ vector and the $\longrightarrow$ vector, we can compute what is the wavelength at point (x_t,y_t).
First, the two lengths of the vectors $\dashrightarrow$ and $\longrightarrow$ are calculated (vectors in the given direction, which go to the boundaries of the image along each axis):

$$\text{length\_to\_x} = d\times \text{width}/\text{abs}(x_t)$$
$$\text{length\_to\_y} = d\times \text{height}/\text{abs}(y_t)$$

where d is the length of the (x_t,y_t) vector.

The shorter of the two is selected (total_length) and used to calculate the wavelength:

$$\text{wl} = \text{total\_length}/(2*d)$$

Having the wavelength, *x* for the perception test image is calculated by the formula:

$$x = 100\times\log_{1.95}(100/\text{wl})$$

work with the frequency components. The image transform is scanned pixel by pixel and for each coordinate, the *x* value for the test image is calculated (see section 2 for the description of the test image). Having the *x* value, the *y* threshold value of the test image can be calculated. If we consider the maximal *y* threshold value as a unit, we can obtain a multiplication coefficient, which can be used to suppress the spatial frequency change at the given coordinate of the Fourier transform. The result would be the flattening of the threshold curve, which is what is needed. Unfortunately, a close examination of the thresholding functions presented in section 2 would reveal that the threshold happens at delta values which for the most part are not achievable on a computer monitor with 255 levels of the brightness or chrominance signal. Furthermore, this method for Fourier transformation modifications is not verified to be correct and was implemented only as an experiment. The results show that the proposed adjustment does not play any significant role to the selection of regions (approximately 1 pixel in each 70000 pixels of the examined images). Thus an implementation is included, but it is not used for the compression method.

The adjustment of the gradient according to the average brightness of the surrounding is much simpler to achieve. For each pixel of the Y band the average value of the 3-by-3 neighborhood is calculated. This is used as the β for the calculation of the adjustment coefficient α as described in section 2. Multiplying the image gradient at each pixel by the α for this pixel results in the flattening of the threshold curve along the “average brightness of the surrounding” axis.

After the gradient has been adjusted, a flat threshold value can be applied across the whole image (band) and the resulting division to black and white regions constitutes the desired extraction of visually significant regions. The threshold value can be chosen, depending on the desired level of detail preservation (the higher threshold value increases the black regions and preserves less detail). Experimentally it was observed that a threshold value of 0.1 produces satisfactory results. As discussed in the next sections, setting this value too low (less than 0.05) or too high (more than 0.15) would result in unwanted image degradation due to the compression methods used.

It is also important to note that for this method the gradient is calculated using a modified Sobel operator (a compacted Sobel operator). The horizontal and vertical filters are respectively

$$\begin{pmatrix} 1 & 2 & 1 \\ -1 & -2 & -1 \end{pmatrix} \text{ and } \begin{pmatrix} 1 & -1 \\ 2 & -2 \\ 1 & -1 \end{pmatrix}.$$

This modification is required, because the original Sobel operator does not consider 1-pixel differences in the image.

4. Compression of visually significant regions

The white regions contain visually significant information and thus it is desirable that any compression method applied to the pixels belonging there does not cause unwanted loss of detail. As discussed in section 3, the white regions are selected based on the value of the gradient (i.e. the gradient has a large value). This implies that there are big differences in the values of the neighboring pixels. The human sight loses precision with the increase of stimulus. In other words, the eye cannot tell between small variations of large differences – the same way people can tell there is a big difference between \$1 and \$2, but tend to disregard the difference between \$200,010 and \$200,020.

Experiments showed that the two least significant bits of pixels in the white regions for both the brightness and the chrominance bands do not play a role for the visual discrimination between the original and the compressed image. Thus for all pixels in the white regions the two least significant bits are discarded.

As suggested in [2], the coding of each bitplane separately could offer a better level of compression. A bitplane consists of the bits of all pixels in the image at a certain level of significance – for example, all most significant bits. To achieve a better compressability, the pixel values are coded first using the so-called gray code. The code is calculated using:

$$g_m = b_m$$
$$g_i = b_i \otimes b_{i+1} \qquad \text{for } 0<i<m$$

where m is the number of bits used to represent the number, b_i are the original bits, g_i are the bits of the gray code representation, and $\otimes$ is the XOR operation.

The gray code is a reversible function, which produces binary representation of numbers in a way such that for numbers, whose difference is one, there is only one different bit in their binary representations. This can be illustrated by considering 7 and 8. The binary representation of 7 is 0111. The binary representation of 8 is 1000. Even though the numbers are very close (their difference is 1), their standard binary representation differs along all bits. However, the gray codes corresponding to 7 and 8 are: 0100 and 1100. The difference is in a single bit only.

By gray-coding the pixel values before compression results in a greater chance that neighboring pixels in the image will have same bits in a given bitplane. Compressing large regions with same values results in a much better compression ratio.

Summarizing, the following method for compression of the white regions is used: gray-code the pixel values and then output the 6 most significant bitplanes separately.

This method, however, poses a limitation to the threshold value which is used to select the visually significant regions. If the value is set too low, regions with a smaller gradient will be compressed and the two least significant bits discarded. This can result in significant degradation of the image, since the bit discarding is acceptable only when the gradient is sufficiently high. A threshold value of less than 0.05 is not recommended.

5. Compression of the visually less significant regions

The compression of the black regions is a more sophisticated process than the compression of white regions, since there is greater freedom in discarding visually insignificant information. As discussed in section 3, the gradient in the black regions is very small and the human eye perceives only smooth changes in such regions. A method which can be successfully used in these circumstances is the prediction-based coding.

The prediction-based coding works as follows: after each pixel is coded, a predictor is queried about the value of the next pixel. The predictor provides a prediction, which is then compared with the actual value of the pixel. The comparison provides the prediction error, which is then used for compression. After the pixel is processed, the predictor is updated with the actual value of the pixel and the compression proceeds with the next iteration. Depending on how successful the predictor is, the prediction errors can be very small and thus much fewer bits could be used to code each pixel. The black regions have a small gradient and thus the prediction-based coding is very suitable – pixel values can be predicted accurately from their neighborhoods. In order to increase the chance of successful prediction, additionally the black regions can be blurred before the start of compression. This is acceptable, since the observers already perceive the regions as blurred. Experiments showed, however, that heavy blurring becomes noticeable. Thus the following filter was selected for convolution with the image (band):

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 9 & 1 \\ 1 & 1 & 1 \end{pmatrix} \times (1/17)$$

Since the black regions can have arbitrary shape, the definitions of neighborhoods had to be changed accordingly. A *same-region neighborhood* consists of all pixels from the standard neighborhood, which lie in a black region. Both the convolution and the prediction use this modified notion of neighborhoods.

After blurring the black regions, the prediction-based coding becomes even more efficient. The expected prediction error is small, however, in certain cases it can be quite big. If a fixed number of bits is reserved for the coding of the prediction errors, then is has to be sufficiently large to describe the greatest possible error (unless we accept that the prediction errors can be recorded inaccurately). Having a large number of bits for the prediction error, however, degrades the performance of the compression, since one expects that the predictor will be successful most of the time. Thus a variable length binary coding for the prediction error was selected.

Experiments showed that the Y band can be compressed much more efficiently than the Cb and Cr bands. It was discovered that even a single-bit inaccuracy in the description of the chrominance bands leads to a perceptible "coloring" of the image in the black regions. Unlike the white regions, where inaccuracy is compensated for by the high gradient, the black regions have smooth areas and the eye is capable to detect the overall change of color easily. On the other hand, the overall change of brightness is not detected so easily. These results are also supported by the thresholding functions described in section 2. The above facts led to the selection of two different coding methods, depending on whether the brightness band or the chrominance bands are coded.

For the coding of the Y band, the prediction error is quantized (with a step of 2) before writing to the output. This does not result in perceivable degradation of the image. Due to the quantization, the values written out are expected to be small and the variable length binary coding used is designed to favor small values (see the table below).

Code for Y band		Code for Cb and Cr bands	
Range of number	Bit representation	Range of number	Bit representation
0	**0**	0	**00**
		1	**11**
[−2;2]	**10xy** if positive, x=1 else x=0, y = bit representation of (abs(value) −1)	−1	**01**
		[−5;5]	**100xyy** if positive, x=1 else x=0, yy = bit representation of (abs(value) −2)
else	**11xy...y** if positive, x=1 else x=0 y...y = bit representation of abs(value) # of bits governed by the maximal prediction error possible	else	**101xy...y** if positive, x=1 else x=0 y...y = bit representation of abs(value) # of bit governed by the maximal prediction error possible

For the coding of the Cb and Cr bands, the prediction error is not quantized and a different coding scheme is used. The variable length binary coding is adjusted to larger expected prediction error (see the table on the previous page).

Unlike with the standard prediction-based coding, where the image is scanned pixel after pixel, in this case the coded regions have arbitrary shape. Thus a method suitable for the traversal of an arbitrary region is required. In [3] such a method is described. Before a pixel is processed, all of its same-region neighbors which have not been processed so far are pushed into a FIFO structure. After the pixel is processed, the next pixel is retrieved from the FIFO structure. In this simple manner, regions of arbitrary shape can be traversed. Experiments were done with 4-neighborhoods and 8-neighborhoods. The experiments were not very extensive, however, it appears that 4-neighborhoods offer a slightly better compression ratios in the general case.

Two types of predictors have been used: a sliding window predictor (used in standard prediction-based coding), and a neighborhood predictor. The sliding window predictor uses the average of the last 10 values to predict the new value. The neighborhood predictor uses the average of the already processed same-region neighbors for the same purpose. As expected, the neighborhood predictor is much more accurate. Used on a single photographic image, the sliding window had a standard deviation of 5.83 for the Y band and 6.42 for the Cb band, while the neighborhood predictor had a standard deviation of 3.85 for the Y band and 2.61 for the Cb band. This resulted in choosing the neighborhood predictor. For each black region, the predictor is initialized with the first byte of the region. This byte is stored in the output stream directly.

Summarizing, the following method for compression of the black regions is used: for each black region initialize the neighborhood predictor with the first pixel in the region, then traverse the region and output the prediction errors for the pixels. If the Y band is compressed, the prediction errors are quantized. The prediction errors are coded with a variable length binary code, which is different for the brightness band and the chrominance bands.

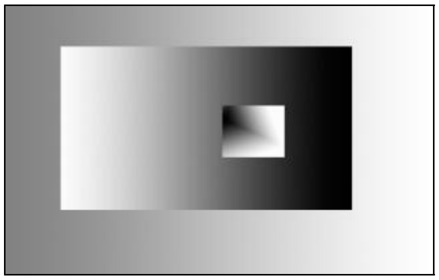
This method, however, poses a limitation to the threshold value which is used to select the visually significant regions. If the value is set too high, regions with a large gradient will be blurred and this will result in image degradation. Furthermore, a greater number of bits will have to be reserved for the prediction error and this will decrease the compression ratio. A threshold value greater than 0.15 is not recommended.

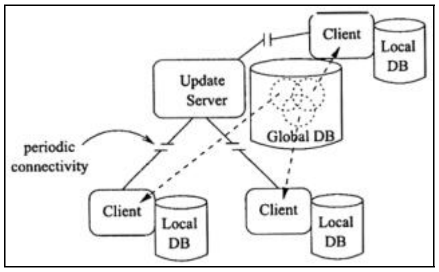
6. Results

The results achieved with the reference implementation are not completely disappointing, even though they do not match the results in [1]. For the test images, a compression of at least 50% over the uncompressed images was achieved. Please refer to the given table for details. In all cases, however, there was no visible image degradation. This has less significance in photographic images, where JPEG-style compressors have outstanding performances (high compression with little visible image degradation). However, the proposed region-based method is applicable also to images of vector clipart, diagrams, text, etc. where the JPEG compression is known to introduce unwanted artifacts. Since it is based on the human vision model, it is completely versatile.

A table with the results of compression, as well as the test images are given below. The images have reduced in size.

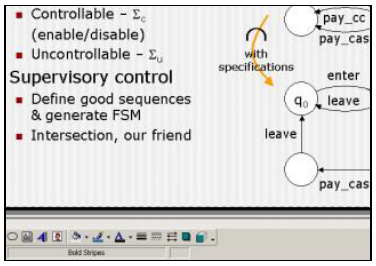
Dcp.jpg


Grad.tif



Infra.tif

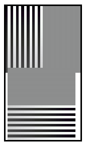
Kingston.jpg

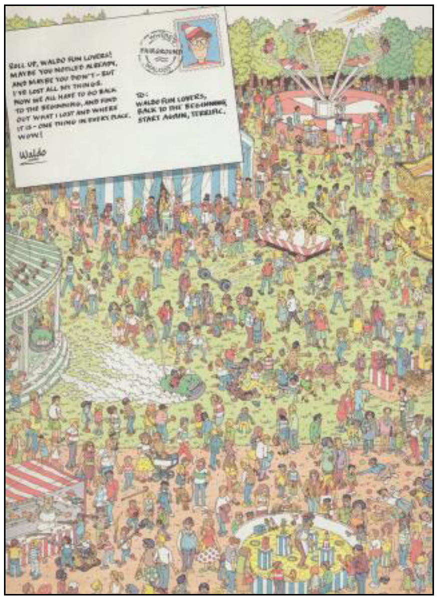

- Controllable – Σ_c (enable/disable)
- Uncontrollable – Σ_u

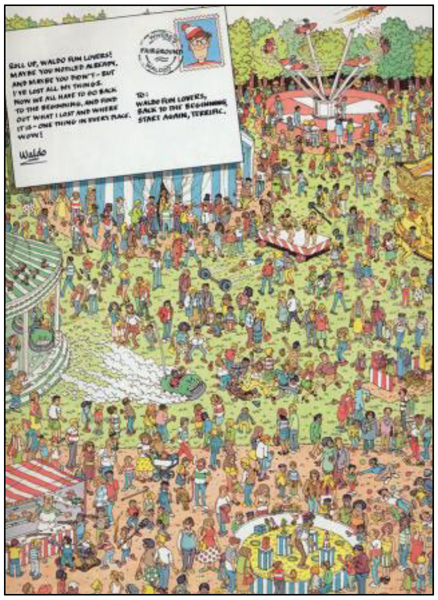
Supervisory control

- Define good sequences & generate FSM
- Intersection, our friend

Ppts.tif


test.tif

Waldo1.bmp



Waldo2.tif
(Waldo1.bmp reduced in half)

Name of image	Uncompressed size	TIFF compressed (lossless)	JPEG compressed (highest quality)	Region-based compressed
Dcp.jpg	5.88 MB	6.53 MB	1.89 MB	2.74 MB
Grad.tif	301 KB	126 KB	30.2 KB	13 KB
Infra.tif	362 KB	35.4 KB	41.3 KB	17.9 KB
Kingston.jpg	524 KB	548 KB	139 KB	209 KB
Pptsm.tif	647 KB	60.6 KB	143 KB	62.8 KB
Test.tif	11.3 KB	1.5 KB	1.45 KB	0.86 KB
Waldo1.bmp	6.32 MB	6.99 MB	2.25 MB	3.15 MB
Waldo2.tif	1.58 MB	1.83 MB	674 KB	927 KB

The method performed poorly with well-compressable images, since in addition to the compressed data it has to store the region bitmaps for the three bands. As expected, the JPEG compression proved to be more desirable for high-frequency images (photographic images), where the slight degradation is visually acceptable.

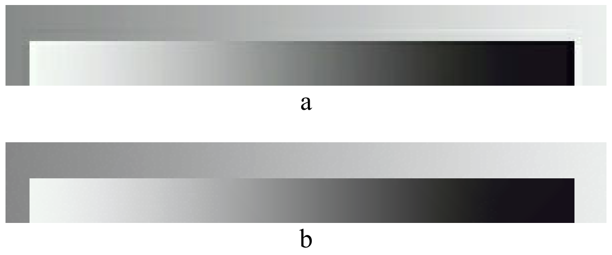


Figure 5 Comparison of quality. a) image compressed with JPEG (an artefact line can be seen above the dark region); b) same image compressed with the region-based method (degradation is not visible).

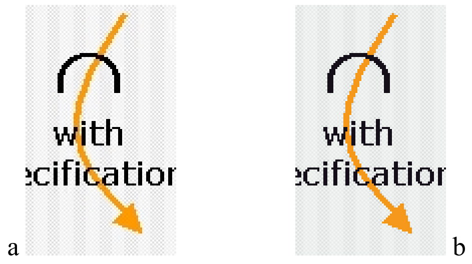


Figure 6 Comparison of quality. a) image compressed with JPEG (arrow has blurred edges); same image compressed with region-based method (edges remain sharp).

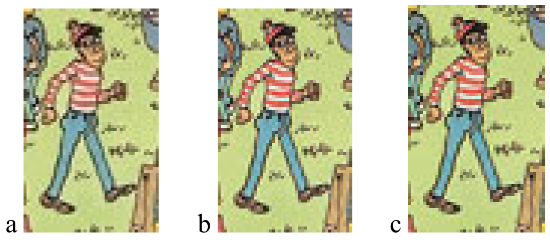


Figure 7 Comparison of quality. a) image compressed with JPEG (hat and stripes are blurred, change of color); b) original image; c) same image compressed with region-based method (degradation not visible).

The implementation used for the testing of the method consists of a number of separate modules, some written in Java and others in Matlab. Thus it is virtually impossible to predict what is the time complexity of the implementation relative to other methods for compression. The JPEG compression processes an image in a single pass, while the proposed method has to make a number of passes. On the other hand, most of the calculations involve very simple operations. The space complexity of the method also exceeds that of JPEG, since the bitmap of the regions is required in addition to the image data. Thus the timings provided below are stated simply for the purpose of relative comparison between executions of the reference implementation. The results were obtained using a VIA C3 766MHz processor, 128MB SDRAM machine with Microsoft Widows XP.

Name of image	Compression (in sec.)	Decompression (in sec.)	Size (in pixels)
Dcp.jpg	887	462	1760×1168
Grad.tif	82	45	400×250
Infra.tif	80	49	451×274
Kingston.jpg	98	54	474×377
Pptsm.tif	155	85	564×391
Test.tif	17	16	28×50
Waldo1.bmp	1503	455	1265×1745
Waldo2.tif	243	121	633×873

Note: the compressed images appear to be slightly more greenish than the originals. This is due to the imperfect RGB→YCbCr→RGB conversion. The compression method uses the intermediary files with the Y, Cb, and Cr bands (please see Appendix A). Thus for correctness, the output should be compared with these images and not with the RGB original. The implementation uses the built-in JAI colorspace converter. A better YCbCr separation should be used to achieve better results.

7. Conclusion

The proposed method for region-based visually lossless compression combines results from different fields of research to offer a way to compress digital images without any loss of visually significant information. The user would not be able to tell a difference between the original and the reconstructed images when they are displayed on the screen. The properties of the Human Vision Model are employed to take advantage of different ways to reduce the visually insignificant data in the compressed image. The YCbCr color model is used, both because of its suitability for compression and because of its resemblance of how people see. Each band is divided into regions with visually significant data and with visually less significant data. The regions are coded using the most appropriate coding method to achieve high compression performance. The size of the compressed images is reduced at least by half. The method is applicable to all types of images, since it takes into account the properties of the human sight rather than mathematical definitions of closeness.

This work serves only as an introduction of this method and many possible improvements are not considered. The proposed modification of the Fourier transform does not yield useful results, so another method has to be applied. The additional employment of mathematical approximation models might also improve the performance. Yet higher compression ratio can be achieved by the use of noise modeling as discussed in [1]. Improvements on all parts of the algorithm are also possible – resulting in higher speed and/or better compression.

References

1. Nadenau M., *Integration of human color vision models into high quality image compression*, Thesis #2296, Federal Technical University of Lausanne, 2000

2. Gonzalez R., Woods R., *Digital image processing*, 2nd ed, Prentice Hall, 2002

3. Fernandez i Ubiergo G., *Lossless region-based multispectral image compression*, Image Processing and Its Applications Conference, Dublin, Ireland, 1997, vol. 1, pp. 64-68

4. Coren S., Ward L., Enns J., *Sensation and Perception*, 5th ed, Harcourt Brace College Publishers, 1999

5. Padgham C., Saunders, J., *The Perception of Light and Color*, Academic Press Inc., New York, 1975

6. Chang S., Messerschmitt D., *Transform Coding of Arbitrarily-Shaped Image Segments*, In proceedings of first ACM international conference on Multimedia, Anaheim, California, 1993, ACM Press, pp. 83-90

7. YCbCr Color Model, ITU-R BT.601 Recommendation, International Telecommunication Union, `http://www.itu.int`

Appendix A

Manual for the reference implementation

The method is implemented using two development environments: Java and Matlab. The following additions to the standard distributions are needed: for Java – JAI and the JAI ICC profiles; for Matlab – IPT. Please note that a beta version of JAI was used, so the function with newer versions is not guaranteed. The required ICC color profiles are included with the distribution of my implementation.

Generation of the perception test pictures

All test images, as well as the code needed to generate them in included in the "perception" subfolder. The raw data from the measurements is available in the Excel file there.

Compression

1. Input – RGB images ONLY
2. Name of input file: "filename.tif" (TIFF/JPEG/BMP accepted) **"filename" is without extension!**
3. Separation to YCC
    a. `java YCCSeparator filename.tif filename`
    b. output – filenameY.tif, filenameCb.tif, filenameCr.tif (grayscale of each band)
4. Selection of regions
    a. In Matlab, run "`regions('filename')`"
    b. Output – filenameYR.tif, filenameCbR.tif, filenameCrR.tif (bitmaps with regions)
5. Compression
    a. `java RegionCodec -c 0.1 filename`
    b. intermediary outputs – filename.cod, filenamei.cod (data, initial pixels for predictor)
    c. output – filename.rbc (GZIP compressed region bitmaps and the intermediary files)
6. Output of YCC separation can be removed.

Decompression

1. Input – "filename.rcb" image obtained from compression, **"filename" is without extension!**
2. Decompression
    a. `java RegionCodec -d 0.1 filename`
    b. intermediary files – filename.cod, filenamei.cod, filenameYR.tif, filenameCbR.tif, filenameCrR.tif (data, initial pixels for predictor, bitmaps with regions)
    c. output – filename_Y.tif, filename_Cb.tif, filename_Cr.tif (grayscale of YCC bands)
3. Merging of bands
    a. `java YCCSeparator -m filename_` (underscore is required)
    b. output – filename_.tif (reconstructed RGB image)
4. Output of decompression can be removed.

The above procedures are automated by the "compress" and "decompress" functions for Matlab (can be used only under MS Windows). Invoke with "`compress('filename.tif')`" (or .jpg/.bmp) and "`decompress('filename')`" (**without extension**). These functions also provide timing. Both functions accept the threshold value as a second parameter, but it is optional. If it is used, the decompression has to be invoked with the **same** value which was used for the compression.

Further documentation can be found in the source files.