Perception-based
second generation image coding
using variable resolution

Joakim Rydell
Master’s Thesis in Image Coding
at Linköping university

LiTH-ISY-EX-3310-2003

7th April 2003
Perception-based
second generation image coding
using variable resolution

Joakim Rydell
Master’s Thesis in Image Coding
at Linköping university

LiTH-ISY-EX-3310-2003
Perceptionsbaserad andra generationens bildkodning med variabel upplösning
Perception-based second generation image coding using variable resolution
Joakim Rydell

Abstract
In ordinary image coding, the same image quality is obtained in all parts of an image. If it is known that there is only one viewer, and where in the image that viewer is focusing, the quality can be degraded in other parts of the image without incurring any perceptible coding artefacts. This master's thesis presents a coding scheme where an image is segmented into homogeneous regions which are then separately coded, and where knowledge about the user's focus point is used to obtain further data reduction. It is concluded that the coding performance does not quite reach the levels attained when applying focus-based quality degradation to coding schemes not based on segmentation.

Nyckelord
image coding, perception, variable resolution, segmentation
Abstract

English

In ordinary image coding, the same image quality is obtained in all parts of an image. If it is known that there is only one viewer, and where in the image that viewer is focusing, the quality can be degraded in other parts of the image without incurring any perceptible coding artifacts. This master’s thesis presents a coding scheme where an image is segmented into homogeneous regions which are then separately coded, and where knowledge about the user’s focus point is used to obtain further data reduction. It is concluded that the coding performance does not quite reach the levels attained when applying focus-based quality degradation to coding schemes not based on segmentation.

Svenska

Vanligtvis kodas bilder så att samma bildkvalitet erhålles i alla delar av bilden. Om kodaren känner till att endast en person tittar på den kodade bilden, och även vet var i bilden denna person fokuserar, kan en försämring av bildkvaliteten tillåtas i andra delar av bilden eftersom användaren inte kommer att uppleva någon förändring. Denna examensarbetesrapport presenterar en kodningsmetod där en bild delas upp i ett antal homogena segment som sedan kodas separat, och där information om var användaren fokuserar används för att ytterligare sänka datatakten. Det konstateras att den uppnådda kodningsprestandan inte når riktigt samma nivå som när fokus-baserad kodning används i bildkodning som inte är baserad på segmentering av bilden.
## Contents

1 Introduction ......................................................... 11
   1.1 Purpose of this work ........................................... 11
   1.2 Outline .......................................................... 12
   1.3 Acknowledgements ............................................... 12

2 Background .......................................................... 13
   2.1 Region-based image coding ..................................... 13
      2.1.1 First generation image coding .......................... 13
      2.1.2 Second generation coding techniques ................... 16
   2.2 Variable resolution reduction ................................. 17

3 Segmentation ......................................................... 21
   3.1 Feature spaces .................................................. 21
   3.2 The segmentation algorithm ................................. 22
      3.2.1 Tessellation ............................................... 24
      3.2.2 Mean shift ............................................... 24
      3.2.3 Splitting .................................................. 26
      3.2.4 Pruning .................................................... 26
      3.2.5 Validation ............................................... 26
   3.3 Parameters ...................................................... 27
   3.4 Performance ..................................................... 29
1.1 Purpose of this work

There are several situations where it is useful to be able to transmit real-time video from an unmanned vehicle to an observer. The observer can either be controlling the vehicle remotely, or simply watching a video stream sent from a fully autonomous vehicle. In many of these situations the bandwidth available for video transmission is limited, which also limits the quality of the video data. In almost all existing lossy video coding schemes the image quality is uniformly reduced, thus incurring the same amount of visual artifacts in every part of the image. If it is known that only one person is watching the video, and where in the image this person is focusing, a better approach is to encode that part of the image in higher quality, while reducing the quality of the other parts of the image. This improves the subjective image quality while maintaining the same bitrate as a conventional encoder would produce. An alternative is to only reduce the quality in the other parts of the image and thereby obtain a lower bitrate. This has been done by modifying a standard JPEG encoder, producing a bitrate reduction of 60 - 75% without incurring any visible coding artifacts [1].

This thesis will present an alternative coding scheme based on region coding, where the image is first segmented into homogeneous regions which are then separately coded. This method provides two different ways to reduce the quality in the non-focused parts of the image. The work is restricted to coding of still images, and unless otherwise noted a video sequence is considered to be a sequence of independent images.
1.2 Outline

In the next chapter some background information is presented. Region-based image coding is described along with its advantages and drawbacks. Focus-based resolution reduction applied to traditional image coding is also briefly covered.

Chapters 3 to 5 each present one of the three main parts of the coding algorithm. In chapter 3, the segmentation procedure that finds homogeneous regions in an image is described. That chapter also includes a motivation of why second generation coding techniques may be a good choice for perception-based image coding. Chapter 4 focuses on how the regions are coded, and chapter 5 describes how the resolution is reduced in the image parts outside of the viewer’s focus.

In chapter 6 the results are presented. Chapter 7 concludes the thesis with a discussion about suggested further work.

1.3 Acknowledgements

I would like to thank my examiner Robert Forchheimer and my supervisor Peter Bergström for giving me the opportunity to work on this project, and for valuable input throughout the process. I would also like to thank my opponent Maria Axelsson for her suggestions and comments on both the implementation and this thesis.
This chapter introduces a traditional image coding scheme (JPEG), and briefly explains how this can be extended to region-based coding. The basics of perception-based image coding are also described.

2.1 Region-based image coding

2.1.1 First generation image coding

In traditional, or first generation, image coding techniques an image is considered to be a number of highly correlated pixels. The coding is performed by uncorrelating the pixel values, and thereby reducing the statistical redundancy in the image. In many widespread coding standards such as JPEG, this is accomplished by dividing the image into square blocks of pixels and transforming each of these blocks using for example the discrete cosine transform (DCT). The transformed blocks are then quantized, and the quantized transform coefficients are zigzag-scanned and run-length coded. Finally, the results of the run-length coding are Huffman coded. Each of these steps is briefly described below.

Discrete cosine transform

The DCT step transforms a block of, for example, $8 \times 8$ pixels into a block of 64 different frequency components. This in itself does not compress the data, since the output has the same number of parameters as the input. The transformed block, however, has a distribution of values that is very different from the image block (see figure 2.1). A typical image has very little energy in its high frequency components, and thus the components of the transformed block that correspond to high frequencies are often close or equal to zero while the low frequency components have large values.
The quantization step is the only step where distortion is introduced. All transform components in each block are divided by individual quantization values from a quantization matrix, known to both the encoder and the decoder, and are then rounded to the nearest integer. This limits the number of possible values for the components and reduces the number of non-zero components, which prepares the blocks for efficient coding. See tables 2.1 and 2.2 for an example.

**Zigzag-scanning and run-length coding**

After quantization, the transform components (except the DC component, which is coded separately) in each block are zigzag-scanned. See figure 2.2. This produces a one-dimensional array of values, where most of the non-zero values (low frequency components) are grouped together in the beginning. Because of this, run-length cod-
2.1 Region-based image coding

Table 2.2: Transform coefficients after quantization

| 1 3 0 0 1 0 0 0 0 0 |

EOB

Table 2.3: Sequence before and after run-length coding

| 1 3 0 0 1 0 0 0 0 0 |

EOB

Figure 2.2: Zigzag scanning order. The top left component, that is not included in the zigzag scanning pattern, is the DC component.

encoding is an efficient way to encode the array. When performing run-length coding, each non-zero value is replaced by a tuple \( (r, n) \), where \( r \) is the number of zeros preceding the value, and \( n \) is the value itself. A special value, end of block (EOB), is reserved for denoting that all remaining values are zeros. (See table 2.3 for an example of run-length coding). The tuples are entropy coded using Huffman coding, and the resulting bit-stream is, after some post-processing (see [2] for details), the final representation of the coded image.
Performance

Using coding techniques such as JPEG, a gray-scale image can be represented in reasonable quality using 0.5 to 1 bit per pixel, which means that a compression ratio of about 1:10 can be achieved. In a typical image, however, not all blocks are efficiently coded, since the assumption about the frequency distribution does not hold for blocks with sharp edges, such as blocks on the border between two differently colored objects. This causes either a higher bitrate, a lower image quality near object boundaries, or both. Another typical artifact in transform coded images is “blockiness”, caused by abrupt transitions on the borders between different transform blocks.

To overcome these limitations, it would be advantageous to divide the image into homogeneous segments prior to coding, and to code each of these segments separately. This is called region oriented coding or second generation coding, and is described in the next section.

2.1.2 Second generation coding techniques

When using region coding, an image is first divided into segments, which are then separately coded. For each segment, information has to be transmitted about its shape, location and content (in the following referred to as texture).

From an image coding point of view, an optimal segmentation is one that produces a set of regions which can be well represented using as few bits as possible, i.e. a set for which

\[ \sum_{i}^{n} (S_i + T_i) \]  

is minimized while the image quality is maintained. \( n \) denotes the number of segments, \( S_i \) is the number of bits needed to represent the shape and location of segment \( i \) and \( T_i \) is the number of bits needed to represent the texture of the \( i \):th region. Finding the optimal segmentation of an image is a very difficult problem to solve, but non-optimal segmentations such as those found by available segmentation algorithms are generally sufficient for reasonable coding performance.

The segmentation as well as the coding can be done in a number of different ways, one of which is described in chapters 3 and 4. For detailed descriptions of other methods, see for example [3].
2.2 Variable resolution reduction

The resolution of the human eye decreases rapidly with increasing angular distance from the focus point. If it is known where in an image the observer focuses, this can be used to achieve efficient image coding by representing the image parts far from the focus point in low resolution.

There are two measures for the resolution at different angles; the minimum size of resolution (MSR), which is the smallest size of a visible object, and the maximum visual frequency (MVF), which is the highest frequency of a visible grating. These measures are related to each other by

\[
MVF = \frac{1}{2MSR}
\]  

In order to utilize the varying resolution, the MSR must be known for each pixel in the image being coded. This can be calculated, given the position of the observer relative to the monitor, the focus point, the size of the image and the minimum angle of resolution (MAR). The MAR is a measure of the smallest angle an object must occupy in order to be visible, and varies with the angular distance between the object and the focus point. In the following, this distance will be referred to as the eccentricity.

Figure 2.3 introduces the necessary variables. Their meanings are as follows:

\[
\begin{align*}
p &= (p_x, p_y) & \text{the point for which MSR is calculated} \\
f &= (f_x, f_y) & \text{the focused point} \\
d &= \text{the distance between the observer and the display} \\
e &= \text{the eccentricity} \\
\text{MAR} &= \text{the minimum angle of resolution at the current eccentricity} \\
\text{MSR} &= \text{the minimum size of resolution} \\
d_p &= \text{the distance between the observer and the current point} \\
d_f &= \text{the distance between the observer and the focus point} \\
r_p &= \text{the distance between the origin and the current point} \\
r_f &= \text{the distance between the origin and the focus point} \\
r_{fp} &= \text{the distance between the focus point and the current point}
\end{align*}
\]

To simplify the calculations, the monitor is considered to be perfectly flat. The origin is always located as close to the observer as possible in the image plane.

The MSR depends on the direction in which it is measured, and is largest in the direction away from the focus point. It is smallest in a direction perpendicular to this. For eccentricities below 30 degrees the difference is small, and since \(e\) is rarely
larger than 30 degrees for computer monitors, the smallest MSR is always used. In
the following, MSR denotes the smallest MSR, which is calculated as in equation
2.3.

\[ MSR = 2 \sqrt{d^2 + r_p^2} \tan \frac{MAR(e)}{2} \] (2.3)

If only objects smaller than the MSR are deleted, no visible distortion is incurred.
Then, however, data reduction is only possible if the MSR is larger than one pixel in
any part of the image. If small amounts of visible distortion is tolerable, the MSR
can be normalized so that the same amount of distortion is visible in every part of
the image. This normalization is done as in equation 2.4, i.e. so that the normalized
MSR \((MSR_N)\) equals the width of a pixel at a chosen angular distance from the
focus point. In the equation, \(e_f\) denotes the fovea angle, which is the eccentricity at
which the MSR is to be equal to the size of one pixel.

\[ MSR_N(e) = \frac{MSR(e)}{MSR(e_f)} \] (2.4)

Figure 2.4 displays the MSR and normalized MSR in an image of 512 × 512 pixels,
viewed from a distance of 50 cm. The image is assumed to be 30 cm wide, and
the observer is located in front of the center of the image. Images 2.4a and b show
the situation where the user focuses on the center of the image, while images c and
d show the MSR and normalized MSR when the pixel at coordinates (100, 100) is
focused. It is evident from the figure that a substantial amount of data reduction can
be obtained by using normalized MSR instead of MSR. The mean MSR values in the
different cases are listed in the figure captions.
2.2 Variable resolution reduction

Figure 2.4: MSR and normalized MSR with different focus points

(a) MSR, $f = (256, 256)$, mean MSR 0.78

(b) Normalized MSR, $f = (256, 256)$, mean MSR 2.42

(c) MSR, $f = (100, 100)$, mean MSR 1.15

(d) Normalized MSR, $f = (100, 100)$, mean MSR 3.56
In first generation image coding, the resolution reduction can be performed by calculating the MVF and discarding DCT transform components that correspond to higher frequencies. This is done for each block in the image. Further information about this can be found in [1]. Chapter 5 describes how the MSR and MVF are used in the proposed scheme.
Several methods exist, which, given a gray-scale or color image, produce a set of homogeneous regions that is suitable for coding. This chapter describes one such algorithm, based on feature space analysis using mean shift. The mean shift operation and its properties are described in detail in [4], and more information about the segmentation method that the suggested algorithm is a variation of can be found in [5]. An advantage to this particular algorithm compared to many other alternatives, is that it does not require any *a priori*-knowledge about the number of regions to be found.

The next section describes what a feature space is and how it can be used for image segmentation. The section after that covers the steps of the segmentation algorithm. Section 3.3 describes the parameters that can be used for tuning the segmentation process, and the chapter is finally concluded with a section about the performance of the algorithm.

### 3.1 Feature spaces

A feature space is a space, in which each sample of a signal is represented by a vector of some dimensionality that varies depending on the application. A gray-scale image, for example, can be represented in either a one- or three-dimensional space. In the one-dimensional case, only the luminance of each pixel is represented, while in the three-dimensional case the spatial coordinates of the pixel are also taken into account. A color image needs two more dimensions; one for each chromaticity component. In image segmentation, the spatial coordinates are important, and are therefore used as two dimensions in the feature space. Figure 3.1 shows an image and its corresponding feature space representation. The feature space coordinates correspond to spatial position and pixel luminance, but are scaled to facilitate the
3.2 The segmentation algorithm

When segmenting an image, a feature space representation is useful, allowing segments to be found by locating points in the feature space where many vectors are clustered together. Since pixels that are spatially close to each other and have similar luminance (and, in the case of color images, chrominance), tend to be close to each other in the feature space, most of the vector clusters represent groups of connected pixels with similar properties. These groups can be thought of as homogeneous regions in the image.

The delineation of vector clusters is a complex problem, and the algorithm is divided into several steps. These steps are explained in the next subsections, and figure 3.2 shows an example image and its segmentation after each of the steps. Since the only difference between the segmentation after the mean shift and that after the splitting is that the non-connected regions have been split into multiple parts, there is no visible difference between figure 3.2b and 3.2c. Despite this, both figures have been included to give a complete description of the algorithm.
3.2 The segmentation algorithm

Figure 3.2: Foreman image and segmentation after each step of the algorithm
3.2.1 Tessellation

Ideally, one would start out at each vector in the feature space and find the closest cluster center. All starting points for which the same cluster center is found would then be considered as one segment or region. Performing this search for every vector would, however, be a very slow process, and thus the search for cluster centers is only carried out for a few selected prototypes. These are chosen such that they are as representative for the whole set as possible. The selection of starting points is referred to as tessellation of the feature space.

The selection of a prototype is done by picking a random vector in the feature space. If this vector is located in a sufficiently dense part of the space (i.e. if there is a sufficient number of other vectors within a hypersphere centered at the selected vector) and there are no other prototypes within a certain distance from the vector, it is accepted as a prototype. New prototypes are added until none of the last 20 selected vectors have been accepted. At that point, it is assumed that there are enough prototypes to represent the entire feature space.

Figure 3.3 shows the selected prototypes in the three-dimensional feature space also seen in figure 3.1.

3.2.2 Mean shift

For each prototype vector, mean shift is used to find a density maximum in the feature space. The mean shift algorithm is based on gradient search, and is performed as described below.

1. Initialize $p$ to the starting point (prototype vector).
2. Find all vectors within a certain distance from \( \mathbf{p} \) and calculate their mean position.

3. Update \( \mathbf{p} \), moving it in the direction of the mean position in 2 (see equation 3.1, where \( \mathbf{x}_k \) is the \( k \):th vector within distance \( d \) from \( \mathbf{p}_i \), \( N \) is the number of vectors within this distance, \( i \) is the number of the current iteration and \( \eta \) is the update coefficient, controlling how fast \( \mathbf{p} \) is moved towards the convergence point). A large \( \eta \) causes \( \mathbf{p} \) to move in larger steps. For reasonable values of \( \eta \), this results in faster convergence, but a too large update coefficient may cause oscillation or even divergence, thereby preventing the convergence point from being found.

4. If the update to \( \mathbf{p} \) was smaller than a certain threshold, stop the algorithm and consider the present value of \( \mathbf{p} \) to be the convergence point. Otherwise, repeat from 2.

\[
\mathbf{p}_{i+1} = \mathbf{p}_i + \frac{\eta}{N} \sum_{x_k | (p_i - x_k)^2 \leq d^2} (x_k - \mathbf{p}_i) \tag{3.1}
\]

At each iteration, all vectors in the feature space are associated with the convergence point if \( \mathbf{p}_i \) is closer to them than any other path between a prototype vector and its convergence point has been before. This is done under the assumption that the same convergence point would have been found if the mean shift algorithm had been used with these vectors as starting points.

At some times, \( \mathbf{p} \) converges to a point in the feature space that is not a density maximum. These false convergence points do not represent regions in the image, and are therefore eliminated. The elimination is done by perturbing each convergence point by a small vector and running the mean shift algorithm again, starting at the perturbed point. If the algorithm does not converge to the original convergence point, that point probably does not represent a region in the image and is therefore disregarded.

If, even after this elimination step, two density maxima are located within the distance \( d \) from each other, this is considered an error and they are replaced by a point at their mean coordinates.

The mean shift is a more general method than has been described here. For example, non-uniform weight functions (called kernels) can be used when calculating how \( \mathbf{p} \) is to be updated in each iteration. The use of different kernels affect the convergence and speed of the algorithm, but the simple uniform kernel works well for the purpose described here. A more detailed description of the properties of the mean shift algorithm can be found in [4].
3.2.3 Splitting

After running the mean shift algorithm, an initial segmentation of the image is obtained. This segmentation has some obvious shortcomings. One of the most important of these is that most segments are not spatially connected, and since distributed segments are not suitable for coding, they have to be split into multiple parts.

This is done by iterating over the pixels in the segmentation map and, at each point, comparing the set of pixels belonging to the same segment as the current pixel to the set of pixels found by performing a flood-fill operation at the current pixel. The difference between these pixel sets is the parts of the current region that is not connected to the current pixel, and a new label is assigned to the pixels in this part of the region. After this step, every region is spatially connected.

3.2.4 Pruning

The splitting step creates many new regions, some of which consist of only a few pixels. These small regions rarely correspond to actual features in the image, and are thus merged with other regions that are adjacent in the image space and close in the feature space. This step generally reduces the number of regions from several thousands to a few hundred.

3.2.5 Validation

The last step of the segmentation algorithm is to merge similar regions, such as large backgrounds that have been split into several segments. For each pair of adjacent regions it is validated that the two regions do not correspond to the same feature in the image. Between the two regions’ density maxima $m_1$ and $m_2$, a straight line in the feature space is found. The density $d(p_i)$ is calculated in surroundings to evenly spaced points $p_i$ along this line, and the ratio

$$\frac{\min d(p_i)}{\min(d(m_1), d(m_2))}$$

is calculated. If this ratio is above a threshold, the two regions are merged.

See figure 3.4 for an example of two regions and the density as a function of position along the connecting line.
3.3 Parameters

There are several parameters that affect the performance of the segmentation algorithm. In the tesselation, the minimum feature space density for accepting a prototype and the minimum distance between two prototypes can be set. The mean shift algorithm is affected by two parameters controlling how close to a feature point must be to affect the update in each iteration. One of these parameters (the *spatial resolution*) controls the distance in the dimensions corresponding to spatial coordinates, and the other (the *range resolution*) controls the distance in the dimensions corresponding to luminance and chrominance.

The splitting step is not controlled by any parameters, but the pruning is affected by a parameter deciding the smallest possible region size. In the validation, the density ratio threshold for merging regions can be set.

Suitable values for all these parameters have been found using trial-and-error. The spatial and range resolution, and to some extent the minimum segment size, can be used to control the number of segments (see figure 3.5), but most of the other parameters affect the performance of the algorithm in too complex and unpredictable ways to be suitable for tuning the segmentation results. The values that have been found seem to work well for different images in different resolutions.

Figure 3.4: Density variation function used for segment validation
(a) Spatial and range resolution = 4
(b) Spatial and range resolution = 7 (the default setting)

(c) Spatial and range resolution = 14

Figure 3.5: Segmentation using different spatial and range resolution
3.4 Performance

Figure 3.6 shows three images and the result of segmenting them using the algorithm described in this chapter. As can be seen, the segments correspond to relatively homogeneous image regions, which suggests that this might be a good preprocessing step for image coding applications.
(a) Claire image  
(b) Segmentation of Claire image

(c) Foreman image  
(d) Segmentation of foreman image

(e) Lenna image  
(f) Segmentation of Lenna image

**Figure 3.6:** Three images and their segmentations
This chapter describes how an image sequence is coded. Since the proposed coding scheme treats a video sequence as a sequence of independent still images, this text will mostly focus on how coding of a single image is performed.

For each image in a video sequence, the location and shape of each segment is coded. This is explained in the next section. In section 4.2 coding of the texture in each segment is described. The chapter ends with a section discussing the obtained results.

4.1 Segment shape coding

Segment shape and position can be coded in many different ways, most of which belong to one of two groups. These groups are contour oriented schemes and shape oriented schemes. In contour oriented schemes, a segment is described by tracking its edges and encoding the parameters that describe them. Shape oriented techniques decompose a segment into simple components which are then encoded. The coding scheme presented in this thesis uses a shape oriented technique based on the binary alpha maps used in MPEG-4 [6].

When encoding the shape of a segment, the bounding box (rounded to multiples of 8 pixels, see figure 4.1) of that segment is found and its location and size are transmitted to the decoder. Then a bitmap of the same size as the bounding box, where each pixel corresponds to one pixel in the bounding box, is created. If the corresponding pixel lies within the segment to be coded, the bitmap pixel is set to one. Otherwise it is set to zero. For each block of $8 \times 8$ bits in this bitmap, if all bits are zeros, one bit ([1]) is transmitted. If all bits in the block are ones, two bits ([0 1]) are transmitted. If the block lies on the border of the segment (some pixels are ones while others are zeros), the two bits [0 0] are sent, the block is divided into four blocks of $4 \times 4$ pixels and this procedure is repeated for each of these blocks. In case any of these smaller blocks
also lies on the border of the segment, it is divided into even smaller blocks and so on until every sub-block only contains either ones or zeros. At worst, this is repeated until the blocks consist of only one pixel each. See figure 4.2 for an example of how a block is decomposed into smaller sub-blocks.

There are, however, situations where this algorithm does not provide efficient coding. For example, the bounding box of the segment shown in figure 4.3a is very large compared to the actual segment, and therefore a significant number of bits are needed to represent the area outside of the segment. One solution to this problem is to, before dividing it into blocks, skew the segment and thereby change its shape into the one shown in figure 4.3b. This causes some additional overhead as information about the amount of skewing has to be transmitted, but the gain is generally larger than the cost.

The choice of skew amount can be made in several ways, the best of which is to encode the segment shape after skewing with all different amounts and to pick the skew yielding the lowest bitrate. This would, however, be a time-consuming alternative. A more efficient heuristic employed in the implementation is to perform the skewing with all available skew amounts, and choose the skew yielding the smallest bounding box.

An alternative to using the decomposition method described here is to encode the segment shape using run-length coding. This was implemented, but resulted in a slightly higher bitrate. In MPEG-4, context-based arithmetic coding (see [7]) is used. This would probably decrease the bitrate, but was not tried because of its implementation complexity.
4.1 Segment shape coding

**Figure 4.2:** Block decomposition in segment shape coding (gray areas belong to the segment being coded)

**Figure 4.3:** Optimal and non-optimal segment shapes and their bounding boxes
4.2 Coding of segment texture

Coding the texture of a segment has much in common with coding an image using a conventional, not region-oriented, approach. There is, however, one more thing to consider; a region, unlike an ordinary image, is not necessarily rectangular. This would not be a problem if the pixels were independently encoded, but since the point with segmenting the image and coding each region separately is to get homogeneous regions that can be efficiently coded, that homogeneity should be used to gain as much coding efficiency as possible. Transform coding schemes such as JPEG perform well when coding images with mostly low frequency content, and would thus be a good choice for coding the textures. But, as described in chapter 2, such coding schemes operate on blocks of 8×8 pixels. Since regions can be arbitrarily shaped, many blocks do not cover 8 × 8 pixels that actually belong to the region. The simplest solution to this problem is to ignore it and encode all blocks where any pixel lies within the region. Then, however, nothing is gained compared to when simply encoding each pixel block without considering the segments at all. Some blocks are even coded more than once. Two better solutions are presented in the next subsections.

4.2.1 Shape-adaptive DCT

Shape adaptive DCT (SA-DCT [8]) is a variation of the standard DCT described in chapter 2, where only those image pixels that belong to the segment being coded are used for calculating the frequency components. Like ordinary DCT, it works on square blocks of pixels.

First, for each column in the block to be transformed, all pixels that belong to the current segment, called active pixels, are moved to the top of the block (figure 4.4b). Then the active pixels in each column are transformed using one-dimensional DCT (the inactive pixels do not affect the transform). After that, the active values in each row are moved to the left (figure 4.4d), and finally the rows are also transformed using the one-dimensional DCT. All information needed to reconstruct the block is contained in the shape description, and thus no extra data needs to be transmitted. Refer to figure 4.4 for an example of coding using SA-DCT.

The number of output coefficients is equal to the number of input pixels, which makes SA-DCT an efficient choice for small segments. The drawback is that pixels in different columns in a block are independently moved before the transformation is performed, which reduces the pixel-to-pixel-correlation and thus increases the bitrate.
4.2 Coding of segment texture

Figure 4.4: Shape adaptive DCT (light: pixel values, medium: vertically transformed, dark: vertically and horizontally transformed)
4.2.2 Low pass extrapolation

Another alternative is to use ordinary DCT for transforming the pixel block, but to alter the block prior to transforming it. Since the objective is to get an easily codable block, as much high frequency content as possible should be eliminated without adversely affecting the reconstruction of the active pixels in the block. One alteration that eliminates unnecessary high frequency components is called low pass extrapolation (LPE) and is used in, for example, MPEG-4 [6]. After LPE, the whole block of 8 × 8 pixels is encoded.

LPE is performed by first calculating the mean value of all active pixels, and assigning that value to the inactive pixels:

\[ f_{r,c} |_{(r,c)\notin R} = \frac{1}{N} \sum_{(r,c)\in R} f_{r,c} \]  

(4.1)

where \( f \) is the block, \( r \) and \( c \) are row and column coordinates respectively, \( R \) is the region being coded and \( N \) is the number of active pixels in the block. This is called mean repetition.

Then, for each inactive pixel \( f_{r,c} |_{(r,c)\notin R} \), the mean value of the surrounding active pixels is calculated:

\[ f_{r,c} = \frac{f_{r,c-1} + f_{r,c+1} + f_{r-1,c} + f_{r+1,c}}{M} \]  

(4.2)

Only active pixels and pixels that have been already been modified are considered in the right hand side of equation 4.2, and the denominator \( M \) is set to the actual number of pixels contributing to the sum. This last step is repeated until no pixels are updated.

Figure 4.5 shows an example of low pass extrapolation. Even though it may not be obvious from the figure, the high frequency transform components are lower after LPE than before.

Since the coding performance of shape-adaptive DCT and low pass extrapolation is roughly the same, LPE was chosen for the presented coding scheme because of its more straightforward implementation.

To make use of the fact that pixel values are correlated not only within a block, but also between different blocks, either the first half row or column of the transform coefficients is predicted from the block beside or above, and the differences are encoded. The choice of prediction direction is made in such a way that the smallest
number of non-zero components need to be coded. This is an extension of the DC component prediction commonly used in transform coding and slightly improves the coding performance.

### 4.3 Results and discussion

For evaluation of the presented coding scheme, both synthetic and natural images were encoded. The same images were also encoded using JPEG, and the results were compared in terms of bitrate, signal to noise ratio (SNR) and subjective image quality. SNR is defined in equation 4.3, where $N$ is the number of pixels, $f_i$ is the value of the $i$:th pixel and $\hat{f}_i$ is the reconstructed value of the $i$:th pixel after coding. Figure 4.6 shows three of the evaluation images and table 4.1 contains information about bitrates and SNR. As can be seen in the figure and table, both the natural images have a lower image quality despite the slightly higher bitrate when coded using the segmentation-based approach. The synthetic image, however, is very efficiently coded.

$$\text{SNR} = 10 \log_{10} \frac{\sum_{i}^{N} f_i^2}{\sum_{i}^{N} (f_i - \hat{f}_i)^2} \quad (4.3)$$

There is no big difference in subjective image quality between the JPEG images and the region-encoded images. The errors in the JPEG images are comparatively small, but spread over a larger area, while the errors in the region-coded images are larger but fewer, and concentrated to segment boundaries. The difference in SNR is largely due to these misclassified pixels near segment boundaries, i.e. pixels that are placed in the wrong region. These pixels can have values very far from those in the original image, and thus lower the SNR significantly. Most of these misclassification errors are not clearly visible in the reconstructed images, which makes SNR a questionable estimate of image quality. This issue is further discussed in chapter 6.

The higher bitrate when using region-oriented coding can be explained by the fact that pixel blocks on region boundaries are difficult to encode efficiently. If the borders between regions had been sharp, very few high-frequency components would have had to be encoded since only pixels on one side of the border is considered in any one block. As it is, most borders in natural images are a few pixels wide, and thus there are generally more high frequencies in blocks on the border between regions than in other blocks. In the synthetic test image, all edges are sharp, and thus this effect is not as apparent as in the natural images.

Table 4.2 shows the coding performance when encoding the foreman test image with
Figure 4.5: Low pass extrapolation
### Table 4.1: Coding results using the proposed scheme and JPEG

<table>
<thead>
<tr>
<th>Image</th>
<th>JPEG Bitrate (bpp)</th>
<th>SNR (dB)</th>
<th>Proposed scheme Bitrate (bpp)</th>
<th>SNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthetic test image</td>
<td>0.18</td>
<td>41.16</td>
<td>0.21</td>
<td>311.9</td>
</tr>
<tr>
<td>Claire</td>
<td>0.58</td>
<td>30.69</td>
<td>0.66</td>
<td>27.82</td>
</tr>
<tr>
<td>Foreman</td>
<td>1.09</td>
<td>24.31</td>
<td>1.32</td>
<td>22.35</td>
</tr>
<tr>
<td>Lenna</td>
<td>1.05</td>
<td>22.25</td>
<td>1.32</td>
<td>20.56</td>
</tr>
</tbody>
</table>

### Table 4.2: Coding performance for different segmentations

<table>
<thead>
<tr>
<th>Segmentation</th>
<th>Bitrate (bpp)</th>
<th>SNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 3.5a</td>
<td>1.60</td>
<td>22.56</td>
</tr>
<tr>
<td>Figure 3.5b</td>
<td>1.32</td>
<td>22.35</td>
</tr>
<tr>
<td>Figure 3.5c</td>
<td>1.07</td>
<td>22.10</td>
</tr>
<tr>
<td>One segment</td>
<td>0.73</td>
<td>21.79</td>
</tr>
</tbody>
</table>

the different segmentations shown in figure 3.5 and when the entire image is considered to be one segment.
Figure 4.6: Images used for evaluation of the coding scheme. Left: original images, middle: absolute error in images coded using the proposed scheme, right: absolute error in images coded using JPEG
There are three different ways to reduce the amount of information that needs to be transmitted to the receiver. Entire segments can be removed if they are small and located far from the focus point, segment boundaries may be encoded in lower quality, and high frequency components of segment texture may be discarded. The third alternative is also used when reducing resolution in traditional image coding schemes.

Excluding entire segments is mostly of theoretical importance since segments would need to be very small if they were to be removed without incurring huge amounts of distortion. Encoding images with such small regions would require a very high bitrate, and the gain achieved by removing some of the segments would not be enough to compensate for this. Therefore, no segment removal is used in the proposed coding scheme. The other two methods for data reduction are described in the next sections. In both these sections, MSR refer to the mean MSR value in the current block of $8 \times 8$ pixels.

5.1 Lossy coding of segment shapes

When coding segment shapes as described in chapter 4, no distortion is incurred. In image parts far from the focus point, this is generally overly conservative. Thus, lossy coding of segment shape is used in image parts where the minimum size of resolution is greater than one pixel.

The lossy coding is implemented by an alteration of the alpha block decomposition algorithm described in chapter 4. Instead of stopping the decomposition when the current block consists of only active or only inactive pixels, the block is divided into smaller blocks until the largest group of connected active pixels in a mostly inactive block (or inactive pixels in a mostly active block) is smaller than the MSR. Figure
Chapter 5. Reduction of resolution

5.1 shows how the example block from chapter 4 would be coded in an image part where MSR is 0, 2 and 4 pixels respectively.

There are two problems with lossy coding of segment shapes. Since the boundaries of the segments are changed by the coding process, after coding some regions will be partially overlapping, thus causing some pixels to belong to more than one region. Also, small gaps may be introduced between regions, and pixels in these gaps will not belong to any region. The first problem can be ignored, since single misplaced pixels do not significantly lower the perceived image quality. The other problem, however, results in pixels with no assigned value in the reconstructed image. The solution employed in the proposed scheme is to iteratively replace all pixels which do not belong to any segment with the mean value of the pixels above, below, to the left and to the right, according to equation 5.1, where \( f_{r,c} \) is the value of the pixel at coordinates \((r, c)\). The iteration is stopped when the sum of squared differences between two successive iterations is less than a certain threshold. Iteration is used because there may be several connected pixels that do not belong to any region, and such pixels are not always assigned a reasonable intensity value in just one iteration. An example of a reconstructed image before and after this correction is shown in figure 5.2.

\[
f_{r,c} = \frac{f_{r-1,c} + f_{r+1,c} + f_{r,c-1} + f_{r,c+1}}{4}
\]  
(5.1)
5.2 High frequency elimination

Reduction of bitrate is also performed by eliminating frequency components above the maximum visible frequency (MVF). This is done by, for each frequency component in the DCT block, calculating the geometric mean of the horizontal and the vertical frequencies (see Table 5.1), and comparing this value to the MVF. The components corresponding to higher frequency than the MVF are then set to zero. Since MVF equals $1/2 MSR$, in most parts of an image it is in the range of 0.1 through 2. Thus, in these image parts high frequencies are removed, and the data rate is reduced.

As can be seen in Table 5.2, the frequency elimination is not as efficient as the lossy shape coding in terms of data reduction. This is a result of high frequencies mostly being represented as segment boundaries rather than as texture content. In the table, the distance to the observer is assumed to be 0.5 m, the image width is 0.3 m and the
### Chapter 5. Reduction of resolution

#### 5.3 Fovea angle selection

The fovea angle introduced in section 2.2 determines the compromise between bitrate and image quality. A fovea angle of about 2 degrees is often considered a good choice [1], and that is also what has been used to produce the results in this thesis. Figure 5.3 shows the Lenna image encoded with fovea angles of 0, 2 and 5 degrees and the corresponding absolute errors. All the images were coded with parameters corresponding to a viewer centered in relation to the images, at 50 cm distance from the monitor. The images were assumed to be 30 cm wide, and the focus point was placed at the center pixel.

<table>
<thead>
<tr>
<th>Focus point</th>
<th>Without reduction</th>
<th>Frequency elimination</th>
<th>Full reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>(256, 256)</td>
<td>1.32</td>
<td>1.25</td>
<td>1.02</td>
</tr>
<tr>
<td>(450, 50)</td>
<td>1.32</td>
<td>1.03</td>
<td>0.69</td>
</tr>
</tbody>
</table>

Table 5.2: Bitrates (in bpp) obtained when coding the Lenna test image (512 × 512 pixels) with different focus points

segmentation shown in figure 3.6f is used.

As an extension of the algorithm, it would also be possible to utilize the varying contrast sensitivity at different angular distances from the focus point to achieve further bitrate reduction. This, however, is not a part of the proposed scheme.
5.3 Fovea angle selection

Figure 5.3: Lenna image encoded with different fovea angles and corresponding absolute error images
A number of test images have been encoded using the proposed scheme for evaluation of the coding performance in terms of bitrate and image quality. The images and their reconstructions are shown in figures 6.1, 6.2 and 6.3, and information about the focus points, resulting bitrates and SNR is available in table 6.1. The gain refer to the relative data reduction compared to encoding the same image without reducing the resolution, as described in chapter 4. All images were encoded assuming that they were 30 cm wide and that the observer was located 50 cm from the image plane, in front of the center pixel of the image. As the figures clearly show, the image quality is highly degraded far from the focus point, while it is well maintained close to where the user focuses. The bitrate is significantly reduced, especially in the more complex foreman and Lenna images. It is also evident from the table that the coding gain is heavily affected by the location of the focus point. This is caused by the rapid growth of MSR at large eccentricities, which can be seen in figure 2.4.

<table>
<thead>
<tr>
<th>Image</th>
<th>Size (pixels)</th>
<th>Focus point</th>
<th>Bitrate (bpp)</th>
<th>SNR (dB)</th>
<th>Gain (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Claire</td>
<td>352 × 288</td>
<td>(176,144)</td>
<td>0.57</td>
<td>24.82</td>
<td>13.6</td>
</tr>
<tr>
<td>Claire</td>
<td>352 × 288</td>
<td>(10,10)</td>
<td>0.40</td>
<td>19.74</td>
<td>39.4</td>
</tr>
<tr>
<td>Foreman</td>
<td>352 × 288</td>
<td>(176,144)</td>
<td>1.04</td>
<td>18.12</td>
<td>21.2</td>
</tr>
<tr>
<td>Foreman</td>
<td>352 × 288</td>
<td>(10,10)</td>
<td>0.81</td>
<td>15.91</td>
<td>38.6</td>
</tr>
<tr>
<td>Lenna</td>
<td>512 × 512</td>
<td>(256,256)</td>
<td>1.02</td>
<td>17.58</td>
<td>22.7</td>
</tr>
<tr>
<td>Lenna</td>
<td>512 × 512</td>
<td>(450,50)</td>
<td>0.69</td>
<td>13.68</td>
<td>47.7</td>
</tr>
</tbody>
</table>

*Table 6.1:* Resulting SNR and bitrate for the test images
Figure 6.1: Claire test image and its reconstructions after coding with different focus points
Figure 6.2: Foreman test image and its reconstructions after coding with different focus points
Figure 6.3: Lenna test image and its reconstructions after coding with different focus points
6.1 Error measurement

In perception-based image coding, ordinary SNR as defined in equation 4.3 does not accurately reflect the perceived error in a reconstructed image. Since much larger errors can be tolerated outside of the viewer’s focus area than close to the focus point, a better error measurement would be obtained by weighting the errors at low eccentricities higher than those at high eccentricities. Even better would be to have the image quality assessed by real users, to get a measure of the subjective image quality. Obtaining a correct weighting function would also require experiments where users assess the quality of a large number of images, and since that has not been possible within this project, SNR has been used despite its disadvantages.
7.1 Conclusion

In [1], perception-based variable resolution reduction has been applied to first generation image coding, yielding 60 - 75% data reduction. In this thesis, a region-based coding scheme utilizing perception-based resolution reduction has been proposed, and it has been shown that the resulting coding performance does not equal that of the traditional coding scheme. The bitrate needed to represent the segment shape information is too high to be compensated by the reduced texture coding bitrate. Variable resolution reduction has also been shown to be less efficient in the proposed scheme than in traditional image coding.

7.2 Further work

Three extensions to the algorithm, that would probably improve the coding performance, are suggested in the next subsections.

7.2.1 Motion compensation

In this thesis, a video sequence has been considered as a number of independent still images. Since almost every real video sequence has a high degree of frame-to-frame correlation, this leads to a high redundancy in the encoded data stream. By matching the segments in one frame to those in the previous frame and encoding the motion parameters, a good approximation of the current frame can probably be obtained. This would leave only a residual image to be encoded, and thereby significantly reduce the bitrate.

In first generation image coding, such motion compensation between frames is used
on square pixel blocks and has been proven a very efficient method for data reduction. In region-based image coding the gain would probably be even higher, since the regions (as opposed to the square pixel blocks) correspond to real features in the images, and thus can be expected to move in a regular fashion between frames.

7.2.2 Improved segmentation

Since the proposed coding scheme only handles gray-scale images, the input to the segmentation algorithm is spatial coordinates and intensity values for each pixel. By incorporating chromaticity information, it would be possible to produce a better segmentation of an image. If multiple consecutive images from a video stream were segmented at once, it is even likely that a set of semantically meaningful segments could be obtained. This would further increase the benefits of both segmentation and motion compensation, if that was also used.

7.2.3 Utilization of varying contrast sensitivity

As was mentioned in chapter 5, frequency components are only classified as visible or invisible. If the varying contrast sensitivity at different eccentricities were also utilized, a further bitrate reduction could be achieved by implementing a coarser quantization of high frequency texture components in image areas far from the focus point.


På svenska

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extra-ordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsmann i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

In English

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© Joakim Rydell