UNIVERSITY OF SOUTHAMPTON

Content-Driven Superpixels and their Applications

by

Richard Lowe

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the
Faculty of Physical and Applied Sciences
Department of Electronics and Computer Science

March 2013

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL AND APPLIED SCIENCE
DEPARTMENT OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Richard Lowe

This thesis develops a new superpixel algorithm that displays excellent visual reconstruction of the original image. It achieves high stability across multiple random initialisations, achieved by producing superpixels directly corresponding to local image complexity. This is achieved by growing superpixels and dividing them on image variation.

The existing analysis was not sufficient to take these properties into account so new measures of oversegmentation provide new insight into the optimum superpixel representation. As a consequence of the algorithm, it was discovered that CDS has properties that have eluded previous attempts, such as initialisation invariance and stability. The completely unsupervised nature of CDS makes them highly suitable for tasks such as application to a database containing images of unknown complexity.

These new superpixel properties have allowed new applications for superpixel pre-processing to be produced. These are image segmentation; image compression; scene classification; and focus detection. In addition, a new method of objectively analysing regions of focus has been developed using Light-Field photography.

Contents

List of Figures

List of Tables

Declaration Of Authorship

I, Richard Lowe declare that this thesis titled, “Content-Driven Superpixels and their Applications” and the work presented in it are my own and has been generated by me as the result of my own original research. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University;
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- Where I have consulted the published work of others, this is always clearly attributed;
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;

Signed: ______________________________

Date: ______________________________

Acknowledgements

I would like to thank Mark for his help, encouragement and allowing me to go off on my own tangent, without which this thesis would not exist. I would also like to thank John Bustard for his useful suggestions and his enthusiasm. Additionally, I am thankful for the useful comments made by my examiners. I would like to thank Jess for her constant support and attempting to keep me sane in the final few months. Finally, I would like to thank my family.

Chapter 1

Context and Contributions

1.1 Context

Image representation is fundamental to image analysis. Most images use a regular rectangular tessellation which can impair interpretation and analysis of underlying structure. For example, representing a circle using a square grid will fundamentally lead to errors. Superpixels aim to resolve this by representing the image in a more logical manner, grouping pixels based on homogeneity criteria and restoring the object boundaries. This provides a differing representation of each image, where the boundaries are irregular and the superpixels are of different size. This is achieved by image oversegmentation, the process of reducing an image into a number of regions, by covering images in such a way as to create non-overlapping regions of homogeneous colour.

A superpixel can be defined as a spatially coherent homogeneous structure[Ren and Malik, 2003]. Figure 1.1 shows an image represented using the basic approach to superpixel generation. Each superpixel contains a small number of colours, yet the area or shape of each superpixel can be allowed to vary. The primary use of superpixels is to significantly reduce the number of pixel regions, typically by two orders of magnitude. This reduction in pixels naturally leads to faster implementation of further image processing algorithms [Rohkohl and Engel, 2007], particularly those concerned with segmentation or classification. However, there is a loss in information where pixels are merged.

There are many potential applications for such a technique, for example in large Landsat imagery, where millions of pixels could be represented as several thousand

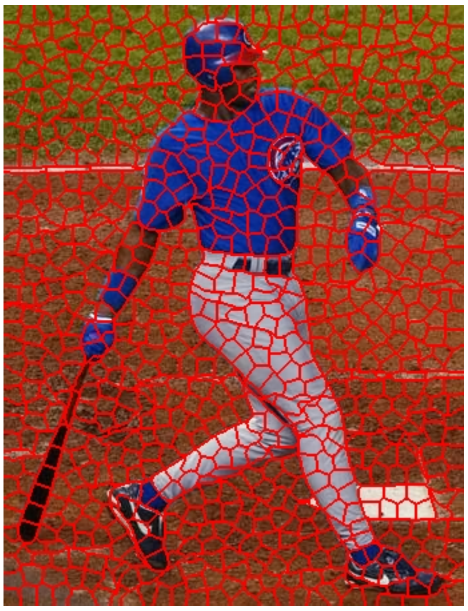


FIGURE 1.1: A superpixel representation of a simple image, generated using N-cuts [Ren and Malik, 2003]

superpixels. There are several notable examples of superpixel use. These include: determining complex body-poses[Ren et al., 2005]; making 3D images from 2D pictures[Hoiem et al., 2005b,a] and videos[Van den Hengel et al., 2007]; object detection in UAV imagery[Rasmussen, 2007]; motion segmentation [Ayvaci and Soatto, 2009]; and as a pre-cursor to scene understanding [Kaufhold et al., 2006].

Superpixel algorithms are usually designed such that there is absolute control over the number of generated superpixels and, historically, algorithms have been criticised for lacking such control. However, when selecting superpixel initialisations, changing the quantity of superpixels even slightly can lead to dramatic changes in the result [Tuytelaars and Mikolajczyk, 2007].

There are two main philosophies to generating superpixels. The first uses a pre-defined number of superpixels with which to partition the image and the other uses image information to generate an unknown number of superpixels based on pre-determined image criteria or thresholds. These are referred to here as explicit and implicit generation respectively. An example of each type of algorithm is shown in Figure 1.2.

1.1.1 Explicit Methods

The first method of superpixel generation using a pre-determined number of superpixels was developed by Ren and Malik [Ren and Malik, 2003]. They used superpixels as a pre-processing stage to achieve segmentation of an image by matching superpixel boundaries with human labeled data. This formulation uses N-cuts graph segmentation [Shi and Malik, 2000] by recursively dividing the image into a pre-determined number of superpixels. N-cuts superpixels are designed to be compact and uniform with respect to size and colour.

The N-cuts based approach is comparatively slow. Turbopixels [Levinshtein et al., 2009], improved the speed of superpixel generation considerably by basing their algorithm on level-sets [Caselles et al., 1997]. A fixed number of seeds is used that are dilated to obtain the superpixels. The seed placement is optimised to best extract the homogeneous regions, and overlap is prevented by way of a skeleton frame. More recently, a new method called SLIC Superpixels has emerged that is designed to produce compact, uniform superpixels that adhere to image edges [Achanta et al., 2012], performing better than N-cuts and faster than Turbopixels. It is inherently a special case of the K-means algorithm as it clusters the data into k points by successively associating pixels to the best matching superpixel ‘cluster’.

Another method, called ‘Lattice Cut’, imposes a lattice structure on the creation of superpixel boundaries [Moore et al., 2010, 2008], thereby retaining pixel-like structure. This restriction on superpixel generation does not hinder the ability of the algorithm to extract well-defined superpixels, but creates a clear way to index superpixels and create a superpixel ‘neighbourhood’.

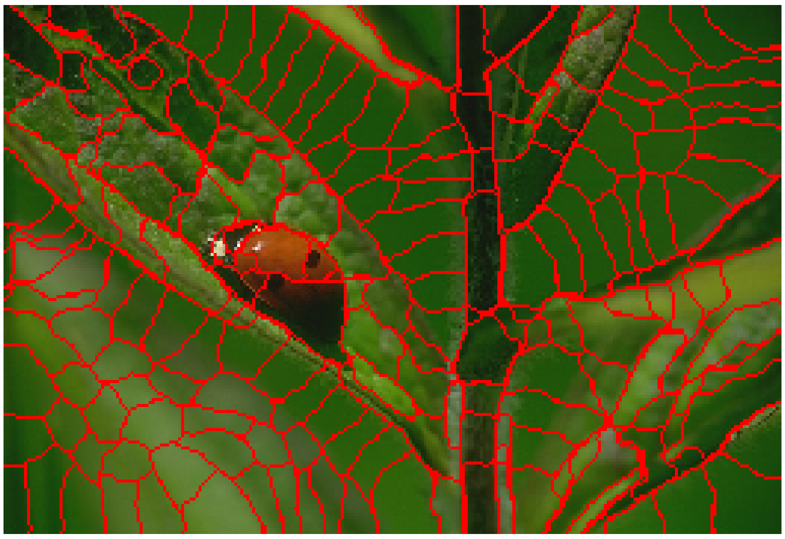
Other methods include those by Zhang et al. [2011] that uses Pseudo-Boolean optimisation and Wang and Wang [2012] that uses Voronoi Tessellations.

1.1.2 Implicit Methods

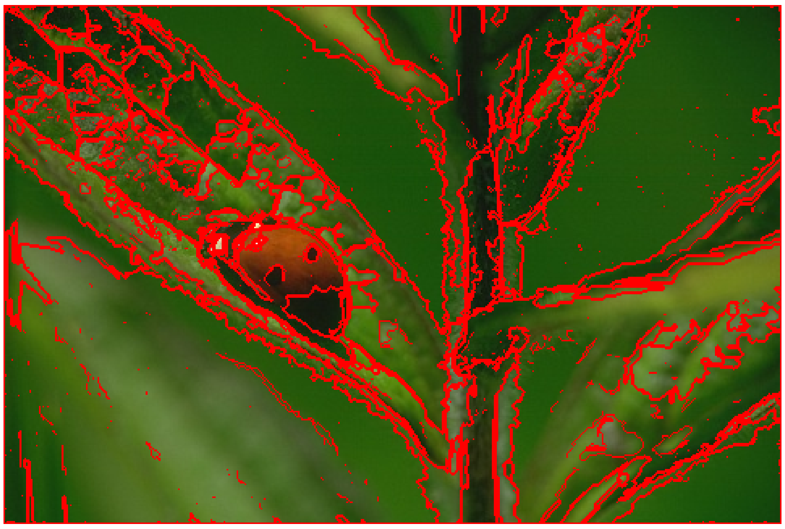
Implicit methods include a parameter that tunes attention to scale within the image. Typically these algorithms are not originally designed to be superpixel algorithms but have found application in this area.

Felzenszwalb and Huttenlocher [Felzenszwalb and Huttenlocher, 2004] created an algorithm initially designed for global image segmentation. They examined the

(a) Original Image



(b) N-cuts superpixels



(c) Felzenszwalb and Huttenlocher

FIGURE 1.2: A typical image from [Martin et al., 2001] represented using different superpixel algorithms

evidence for a boundary between nodes in a graph-based representation of the image. It is controlled by the use of k, which sets the scale of observation, where a larger k causes a preference for larger components. This value is fixed before processing and does not take image complexity into account.

Mean-shift [Comaniciu and Meer, 2002] is predominantly used for clustering, however it is often used for comparison with superpixel algorithms. It replaces each pixel with the mean of a region given a capture range and colour distance between pixels. Again, this capture range and colour distance is chosen prior to operation. To produce superpixels, this output must be clustered using, for example, K-means.

Mean-Shift was extended by Quick-shift [Vedaldi and Soatto, 2008]. It is used as a pre-processing stage in mean-shift. It is, as the name suggests, much faster than mean-shift yet uses a similar algorithm.

Other algorithms include those by Wang et al. [2011] that used Cellular Automata and Liu et al. [2011] that used measures of entropy.

1.2 Contributions

This thesis introduces a novel superpixel algorithm that differs from current superpixel methods. As in traditional superpixel algorithms, the aim is to reduce the number of pixels for further processing. This technique, however, dramatically improves the visual quality of the image when represented by superpixels.

A set of superpixels is evolved, without any initialisation parameters, by growing and dividing one or more superpixels. These new superpixels are formed using a Distance Transform and a variation on Active Contours without Edges segmentation, where the superpixel is separated by optimising the distance between colour values within the superpixel.

While existing algorithms can control the number of superpixels, this control provides change on a linear scale. The result is that all regions will grow or shrink accordingly. Our new algorithm is not designed to scale regions linearly, rather it is designed to produce varying sizes of superpixels dependent on the local complexity of the image. This method still allows images to be reduced as in a traditional superpixel algorithm while retaining almost all of the detail in the original image. By allowing new superpixels to be generated as required, a representation approaching multi-resolution superpixels is obtained, as more superpixels are required to represent more complex regions. As a direct consequence the algorithm adapts to changes in the image, leading to high stability across a set of random initialisations.

The term content-driven superpixels is used as the algorithm directly responds to image variation and the results reflect the local properties of an image. For convenience, the algorithm will henceforth be referred to as 'CD superpixels' or CDS.

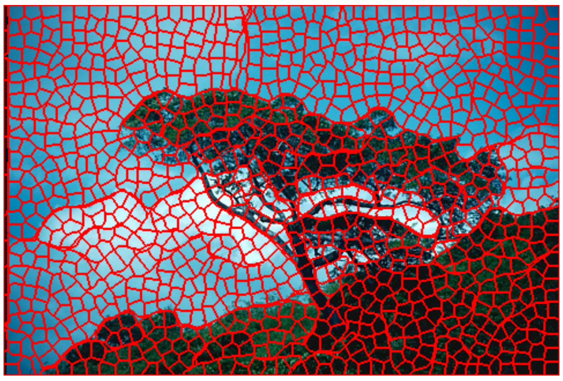
Figure 1.3 shows the difference in approach between N-cuts and our approach. The reconstruction is performed by drawing the mean colour of each superpixel for the area, shown in red, that it encompasses. This is an excellent example of what the new algorithm can achieve. As new superpixels only occur when the current representation is no longer sufficient, the sky contains very few new superpixels, with borders mainly being due to initialisation. In contrast to this, the tree and surrounding grassland is, in some places, almost at pixel resolution. The result is a situation where the superpixels in the sky are reduced yet the foreground remains clear.

Superpixels remain an untapped resource as a pre-processing step, largely because the superpixels generated contain little information about the image content. Generating equal sized regions does reduce the number of regions to process, but provides no other information. By generating superpixels based on image content and structure, there are several new applications that can be explored.

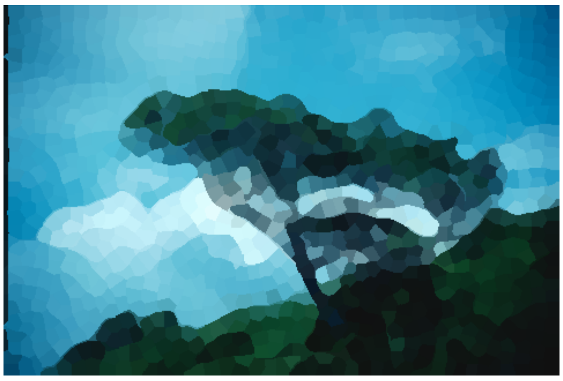
1.3 Thesis outline

Chapter 2 presents the algorithm for CDS, showing the alternatives that were tested. In addition, problems with existing analyses are outlined and solutions are suggested and implemented. The remainder of the chapter presents the quality of the algorithm when compared with other algorithms and displays the improvement of CD superpixels over those algorithms. In addition the robustness to noise and initialisation is demonstrated.

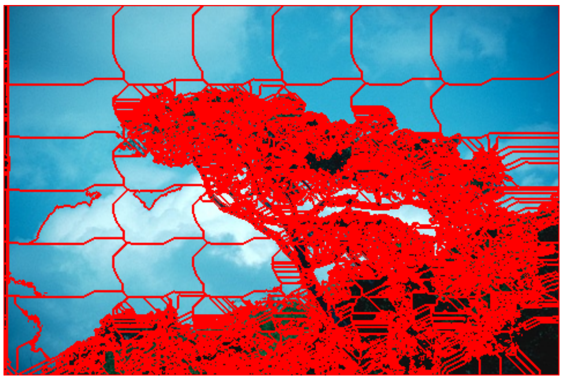
The new technique is then applied in general and in a specific application. Chapter 3 discusses the use of CDS in three applications which are general to computer vision: segmentation; compression; and scene classification. Segmentation



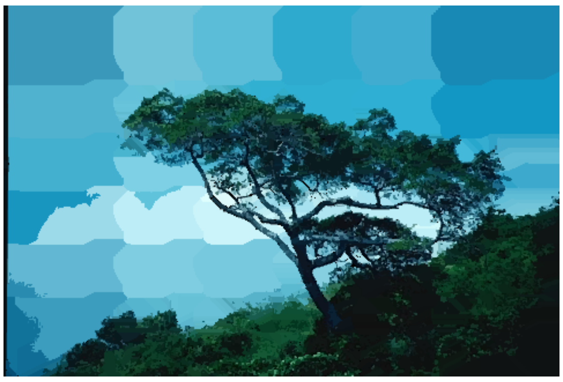
(a) Superpixels generated using N-cuts.



(b) Image reconstruction using N-cuts.



(c) Superpixels generated using CD superpixels.



(d) Image reconstruction using CD superpixels.

FIGURE 1.3: A comparison of CD superpixels with existing work using the same number of superpixels. As CD superpixels are not enforced to be uniform in size, it produces larger regions, notably in the sky.

is achieved by modifying the representation of the superpixels into a graph structure. Scene Classification also relies on the same structure to generate the feature vector required for learning. Machine learning techniques are then applied to recognise the content of an image.

Chapter 4 discusses the application of CDS to finding image regions that are in focus. Generation of test images is provided by using Light-Field images, which is a first for focus detection. The use of Light Fields provides unique information thereby allowing the capabilities of the new focus algorithm to be objectively demonstrated.

There are many other applications that are touched on during the course of the thesis. Chapter 5 discusses further work in analysing and applying the new content driven superpixel approach.

Finally, Chapter 6 concludes.

The publications arising from this thesis are as follows:

- R.J. Lowe and M.S. Nixon. Evolving Content-Driven Superpixels for Accurate Image Representation. In *ISVC2011*, pages 192–201, 2011
- R.J. Lowe and M.S. Nixon. Detecting Focal Regions using Superpixels. In *VISAPP*, 2013

Chapter 2

Developing Content-Driven Superpixels

Existing superpixel approaches all require some form of initialisation. This is either in the form of specifying the number of regions, or a parameter that controls the variance within the superpixels. The result is that either some images are over-represented, containing more superpixels than necessary, or under-represented, containing fewer superpixels than necessary. In addition, many algorithms are unstable as changing the initialisation can drastically alter the result [Tuytelaars and Mikolajczyk, 2007].

To combat this, the superpixels developed here are allowed to evolve through the image in order to develop into the ‘best’ superpixel representation without constraining them through initialisation. The overall scenario for the CDS approach is that a set of seed points is initialised on an image, shown in Figure 2.1. Given this initialisation, the aim is to determine the superpixels which are derived by content driven analysis. On a blank image there is no content and the result would be a number of superpixels equal to those used for initialisation. For an image with content, the superpixels will adapt in order to represent the content faithfully; large structures (areas of similar colour) aim to be represented by a small number of large superpixels and small structures to be represented by a large number of smaller superpixels. Accordingly, the new approach evolves from the initialisation to the final representation. This requires a growth stage when the areas of similar content are to be determined. Given that the representation adapts to content, this predicates a division stage when change in content is encountered. The choice of the number of seeds and their location does not to significantly affect the result. This is demonstrated in Section 2.6.2 .

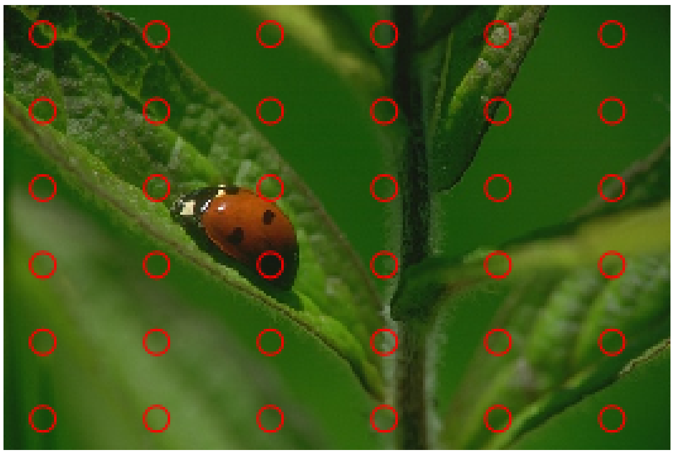


FIGURE 2.1: Initialisation of CDS on an image, shown in red.

A superpixel is defined as the set of pixels over which it has grown. Growth adds new pixels to this set; division creates new superpixels by making new sets that correspond to the newly segmented pixels. The algorithm is seeded in a regular grid pattern, where a single pixel is used to initialise each superpixel.

2.1 Growth Phase

Growing a set of superpixels independently is a difficult task as there is no information passed between each superpixel. With no information, each superpixel will grow to fill the same space as another superpixel, which is obviously inefficient. Several methods are explored below to solve this problem. These are presented along with the chosen method: the Distance Transform.

2.1.1 Active Contour Model

The work by [Kass et al., 1987, Cohen et al., 1990] on parametric active contours is suitable to adapt for superpixel growth. Cohen and Cohen devised a method to make active-contours grow outwards by inserting a normal force (which they called a balloon force) into the snake evolution equation, given in Equation 2.1, that would suit superpixel growth. The response of the snake to the elasticity and curvature control mechanisms are represented by $\alpha(s)$ and $\beta(s)$ respectively, $\rho\underline{\hat{n}}$ determines the normal force and $v(s)$ describes the contour.

Figure 2.2 illustrates the calculation of the normal force of each point. Contour

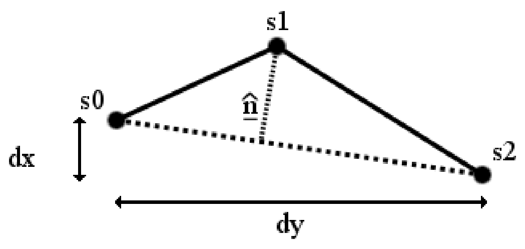


FIGURE 2.2: Illustrating the normal force calculation

points s0 and s2 are used to calculate the force on s1, which then acts outward from s1.

$$\begin{aligned} E_{\text{snake}} &= \int_{s=0}^{1} E_{\text{int}}(v(s)) + E_{\text{image}}(v(s)) + E_{\text{con}}(v(s))ds \\ E_{\text{int}} &= \alpha(s)\left|\frac{\mathrm{d}v(s)}{\mathrm{d}s}\right|^2 + \beta(s)\left|\frac{\mathrm{d}^2v(s)}{\mathrm{d}s^2}\right|^2 + \rho\underline{\hat{n}} \end{aligned} \tag{2.1}$$

One modification must be made to the original snake to use it for superpixels. This is an additional term that relates to the proximity of other superpixels, to avoid the superpixels overlapping as they grow. This additional constraint is shown in Equation 2.2, where E_{sp} represents the combination of all other superpixel boundaries in the image; analogous to image edges. This could be combined with the image energy however including an additional energy term allows different weighting to be applied. The edges of each superpixel are treated as image edges such that each superpixel will attempt to adhere to the edges of neighbouring superpixels.

$$E_{\text{snake}} = \int_{s=0}^{1} E_{\text{int}}(v(s)) + E_{\text{image}}(v(s)) + E_{\text{con}}(v(s)) + E_{\text{sp}}(v(s))ds \tag{2.2}$$

As the Cohen balloon forces the contour to expand, avoiding superpixel overlap becomes more difficult. The points on the snake become further apart and so there is an increasing amount of information between these points that gets ignored, shown in Figure 2.3, where the point in red appears within another shape. To maintain stability with large contours, new points are added to the snake as it grows. The main problem with this approach is that it is slow to check for superpixel overlap at an increasing number of points and with an increasing number of superpixels.

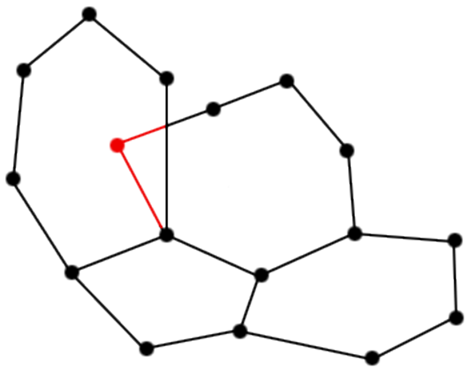


FIGURE 2.3: Illustrating the effect of contour overlap

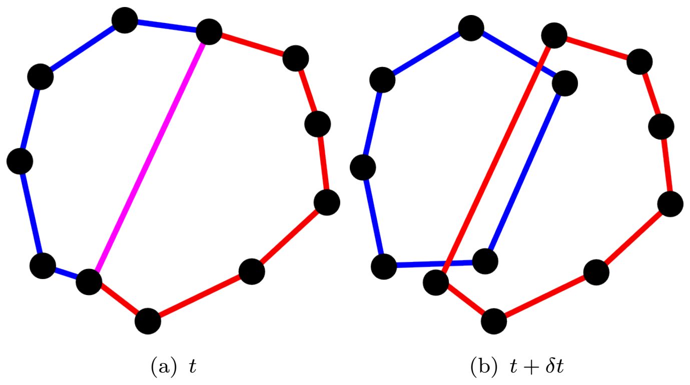


FIGURE 2.4: Overlapping snakes by dual-superpixel growth

Figure 2.4 shows a problem that occurs when growing two superpixels, labelled red and blue, close to one another with their joint boundary shown in purple. Despite the additional term in Equation 2.2 the superpixels do not identify each other as edges and so they willingly grow into one another. This is an inherent problem with this style of active contour and is difficult to overcome without exhaustively checking for the existence of other adjacent contours; increasing the computational load of the algorithm.

Many people have moved away from this implementation for segmentation due to its inability to adhere to edges when the initialisation is too far from the desired contour. Like all explicit contour methods the Kass snake suffers from topological changes, which is the major argument for using more developed, and complex, methods such as Gradient Vector Flow (GVF) [Xu and Prince, 1997], level-set implementations [Osher and Sethian, 1988] and geodesic active-contours [Caselles et al., 1997].

2.1.2 GVF field

One way to alleviate the problem of contour overlap could be to extend the capture range of the contours. The most well-established method for doing so is Gradient Vector Flow (GVF) [Xu and Prince, 1997]. GVF replaces the image force E_{ext} with a vector field that pulls the snake toward strong features from a long distance. The snake evolves according to Equation 2.3 where v represents the vector field, x represents the contour and α and β are consistent with the parameters in Equation 2.1. The vector field can be solved using a finite-difference method [Xu and Prince, 1998].

$$x_t(s,t) = \alpha x^{''}(s,t) + \beta x^{''''}(s,t) + v(s) \tag{2.3}$$

There are however some changes to be made to the original formulation if this is to be used for superpixels. Instead of contours growing to find objects, contours will grow freely unless in the presence of another contour. To achieve this, the vector field only acts outward from the contours, such that any overlap with other contours is seen as an edge during evolution. Figure 2.5 and Figure 2.6 illustrate this. To do this efficiently and without heuristic derivations, the normal gradient to the contour is computed for all points on each contour, then the subsequent gradient field is diffused, to increase the capture range of the field. This process is given in Equation 2.4. Here, u and v represent the two components of the vector field at iteration n. The rate of diffusion is controlled by r and should not exceed the Courant-Friedrichs-Lewy limit [Xu and Prince, 1998]. u_0,v_0 represent the initial conditions, that is, the sum of components i, j of the normal n_k to each superpixel C_k.

When computed for all superpixels, this has the effect of imposing reduced growth in the presence of additional contours and unimpeded growth otherwise. This method replaces the balloon force that was previously required for contour growth.

$$\begin{aligned}
u_{i,j}^{<n+1>} &= u_{i,j}^{<n>} + r(u_{i+1,j}^{<n>} + u_{i-1,j}^{<n>} + u_{i,j+1}^{<n>} + u_{i,j-1}^{<n>} - 4u_{i,j}^{<n>}) \\
v_{i,j}^{<n+1>} &= v_{i,j}^{<n>} + r(v_{i+1,j}^{<n>} + v_{i-1,j}^{<n>} + v_{i,j+1}^{<n>} + v_{i,j-1}^{<n>} - 4v_{i,j}^{<n>}) \\
u_{i,j}^{<0>} &= \sum_k \underline{\hat{n}_k}_i C_k(i,j) \\
v_{i,j}^{<0>} &= \sum_k \underline{\hat{n}_k}_j C_k(i,j)
\end{aligned} \tag{2.4}$$

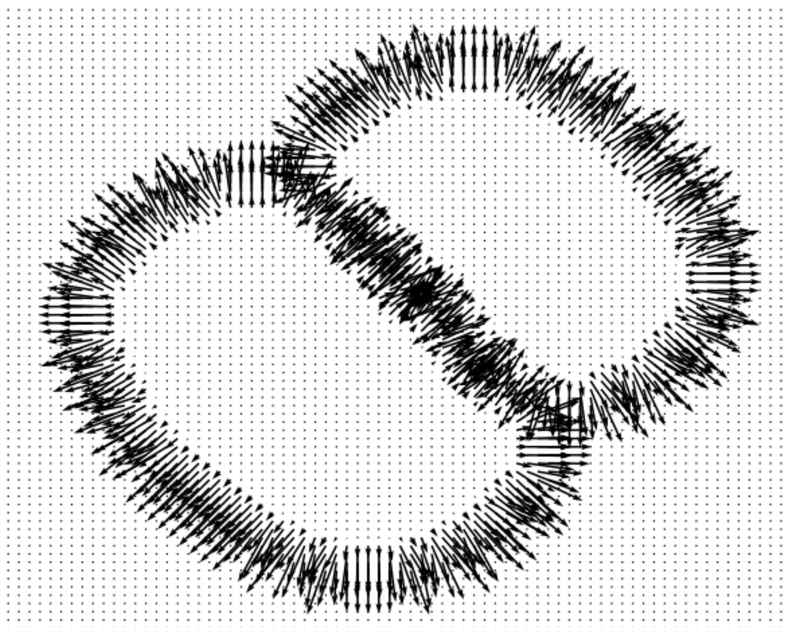


FIGURE 2.5: Vector field produced by two neighbouring contours

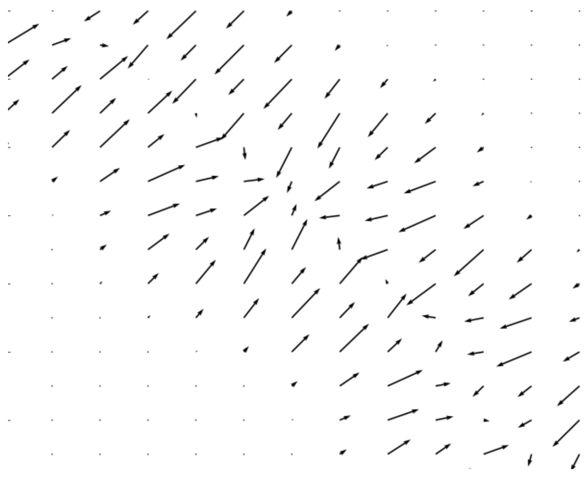


FIGURE 2.6: Zoom on the superpixel interface

GVF fields work well in theory however in practice there is one flaw. The fields created by each superpixel have to perfectly balance out else one superpixel will 'push against' the other causing it to retreat. What occurs then is that the superpixel that advances then finds image variation and divides, creating a new superpixel. Were this to be used, this process would then repeat, causing numerous superpixels to occur erroneously.

2.1.3 Distance Transform

2.1.3.1 Introduction

When controlled by image properties, as has been the case so far, superpixel growth is difficult to control and produces initialisation problems. If the superpixel is designed to halt growth on reaching image features, it is unlikely to cover the entire image without at least estimating the number of superpixels required prior to computation. The minimum number of superpixels is also limited by the initialisation. Hypothetically, one would need to know the number of superpixels and an estimate of optimum location (see, for example, [Levinshtein et al., 2009]). Therefore, the best way to cover the image without requiring a-priori knowledge is not to consider image properties during superpixel growth. By only considering superpixel boundaries, each superpixel can grow unimpeded unless in the presence of another superpixel.

To grow superpixels without considering the image, the distance transform is considered. The distance transform [Borgefors, 1986] can be considered equivalent to binary dilation from mathematical morphology but can be computed significantly faster. In morphology, the object is successively eroded until it disappears, with the value of each pixel corresponding to the number of iterations until that particular pixel disappeared.

It is typically used for skeletonisation of image objects where the skeleton is described by transforming each object such that it displays the distance of object pixels to its boundary with the background.

The algorithm transforms an image to show the distance to a specified colour. This can be used in binary images by showing the distance to either state. The transform also returns an array that determines the closest location in the image that matches the specified colour or state.

2.1.3.2 Use of the distance transform on superpixels

To grow a superpixel, a distance transform of **every** superpixel that exists is taken. The superpixels are not transformed individually to eliminate superpixel overlap. This is illustrated later in Figure 2.8.

The algorithm transforms each superpixel S such that the set of pixels at each location (i, j) within the superpixel display the distance D to the background (in

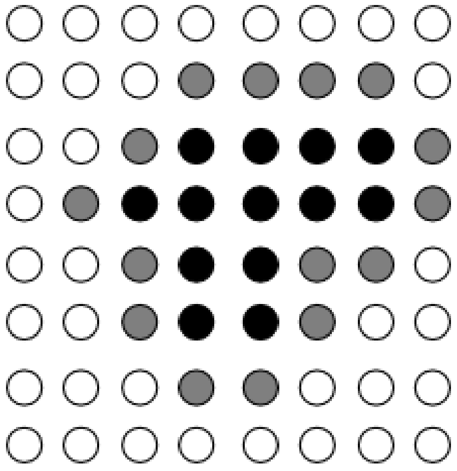


FIGURE 2.7: Binary Dilation operator on a superpixel shape. The grey pixels indicate where the superpixel will grow.

this case, the region in which superpixels have yet to form). Superpixel edges therefore have a distance of one from the background. This is shown in Equation 2.5, where the locations (k, l) form the set of points that constitute the background. The image I used to calculate the distance transform is a binary image where True denotes that a superpixel covers this point in the image and False otherwise. The background is therefore all the False points in this image. The same image is used to individually grow each superpixel.

$$D(i,j) = \min_{k,l:\ I(k,l)=False} \sqrt{(i-k)^2 + (j-l)^2} \tag{2.5}$$

$$S^{<t+1>} = S^{<t>} \cup \{(k,l)\colon D(i,j) = 1\} \tag{2.6}$$

Equation 2.6 shows the iteration, t, of the superpixel to include the background location (k, l) that is adjacent to the superpixel edge. By only considering the pixels that have a distance of one from the background, superpixel overlap is handled implicitly. Any pixel inside the superpixel or adjacent to another superpixel is not connected to the background and hence the superpixel cannot grow at these locations. As the distance transform is stable for any shape, the superpixels can grow from any initial size and shape. The stopping criterion is a direct consequence of the algorithm; terminating once the superpixels cannot grow any further. This occurs when they are completely bordered by other superpixels or image boundaries.

Figure 2.7 shows how the binary dilation works on a superpixel. The black pixels

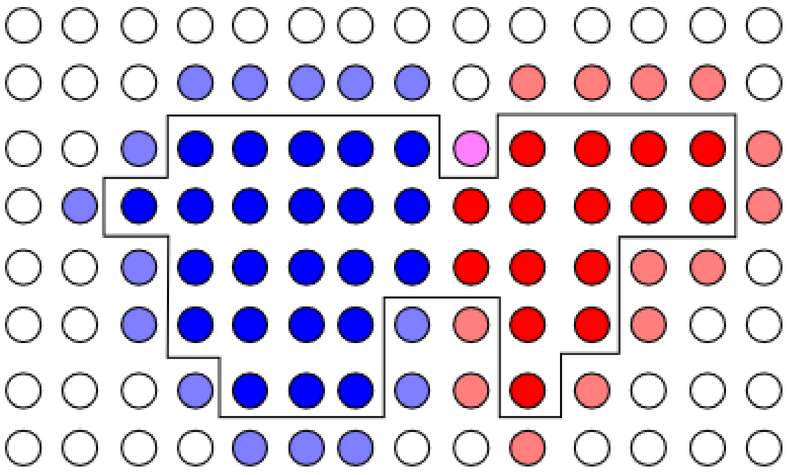


FIGURE 2.8: Dilation operator on two neighbouring superpixels. The red and blue pixels show the two different superpixels. The lighter pixels show the new pixels for each superpixel. The outline shows the current envelope of the superpixel area.

represent the original shape, the grey pixels are the parts to which it grows in the next iteration and the white pixels are the background.

Figure 2.8 demonstrates how this extends to two neighbouring superpixels. The two superpixels shown in opposing colours, red and blue, only grow where they are not bordered by another superpixel. The outline shows the current envelope of both of the superpixels. There is a single purple pixel in the image to show the potential overlap of two superpixels in close proximity. As each superpixel grows independently, both superpixels will occupy this location after growth. Post processing is performed to remove these overlapping pixels from one of the conflicting superpixels.

2.2 Division Phase

As superpixel growth does not take the image information into account, it is required that the superpixel division method must take the image into account in order to fully describe the image. What follows is a comparison of several methods of segmentation available to divide superpixels.

2.2.1 Graph Partition

Using the N-cuts algorithm it would be possible to partition the superpixel using a pixel-based graph created within the superpixel. Weights are applied to graph edges according to the Euclidean colour distance between nodes. The minimum cut through the edges can be taken in order to segment the superpixel. However, segmentation will always occur if there is not a minimum weight, so a superpixel containing one colour will divide into two superpixels of the same colour. This is an undesirable property. If at each iteration the superpixel area increases and is subsequently halved, it will be very difficult to achieve much less than pixel resolution.

2.2.2 Watershed

The watershed algorithm [Meyer and Beucher, 1990] ‘floods’ an image from a chosen number of local minima until the sources meet. The boundaries of these independent floods become the segmentation boundaries. Usually this would require knowledge of the number of regions required, however binary segmentation would be possible given two sources. This method would always create two regions irrespective of image content, so a way of sensing variation or a bi-modal distribution would have to be produced in order to trigger the segmentation. The location of the initialisation would also significantly affect the result for example if the two minima were close together or near the edge of the superpixel.

2.2.3 Region Growing

Similarly to the Watershed algorithm, Region Growing involves growing a single pixel from local minima until no new pixels can be added. It will only add pixels that are similar to pixels already contained in the region. However, Region Growth can suffer from initialisation variation. One initialisation can give a vastly different result to another. This can be seen in Figure 2.9. As the algorithm only adds similar pixels, growth stops once the colour boundary is reached. The result is that the algorithm is susceptible to noise. One advantage over the Watershed algorithm is that it would only produce two superpixels if there was a boundary created within the superpixel. If the algorithm were to reach the superpixel boundary without stopping then the superpixel is homogeneous with respect to the splitting criterion.

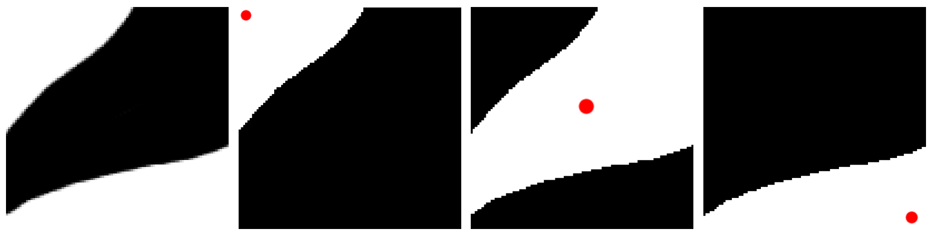


FIGURE 2.9: Region growing initialisation problem, with the seed point marked in red. Only the centre seed gives the correct result.

2.2.4 Local Active Contours without Edges

As superpixel division (via segmentation) is occurring at a small scale, more sophisticated segmentation algorithms become viable. Using, for example, a kernel based on Mumford and Shah [1989] leads to a situation where region-based segmentation algorithms can be used to generate new superpixels. Active Contours Without Edges (ACWE) cannot normally be used in complex images due to its creation of only two regions: object and background. However in this case, region-based segmentation becomes an ideal solution.

A benefit of ACWE is the addition of localised smoothing introduced by the approach. This helps to restrict superpixel division; a necessary requirement due to the greedy nature of the algorithm. In addition, division will not occur if the colour of the superpixel is uni-modal.

2.2.4.1 The Basics of ACWE

Active Contours Without Edges (ACWE) aims to partition an image u_0 into two piecewise-constant intensities of distinct values u_1 and u_2. These piecewise regions are separated by a boundary c_0 such that Equation 2.7 is minimised.

$$F_1(C)+F_2(C) = \int_{\text{inside}(C)} |u_0(x,y)-c_1|^2 dxdy + \int_{\text{outside}(C)} |u_0(x,y)-c_2|^2 dxdy \quad (2.7)$$

where F describes the force from inside and outside the contour and c_1, c_2 are the averages of the regions inside and outside the contour. It can be easily seen from this that if the boundary lies outside c_0, then $F_1(C) > 0$ and $F_2(C) = 0$. If the

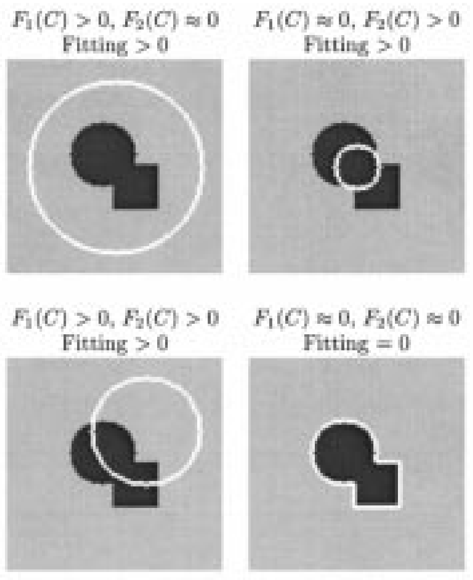

$F_1(C) > 0, F_2(C) \approx 0$
Fitting > 0
$F_1(C) \approx 0, F_2(C) > 0$
Fitting > 0
$F_1(C) > 0, F_2(C) > 0$
Fitting > 0
$F_1(C) \approx 0, F_2(C) \approx 0$
Fitting $= 0$

FIGURE 2.10: Illustrating the evolution of the boundary [Chan and Vese, 2001]

boundary is inside c_0 then $F_1(C) = 0$ and $F_2(C) > 0$. This is shown in Figure 2.10.

Next some regularising terms are added to control the length of the contour and the area of the region inside C. This can all be represented by an energy function F in Equation 2.8, where $\mu, v, \lambda_1, \lambda_2$ are weighting co-efficients.

$$\begin{aligned} F(c_1, c_2, C) = \mu\cdot \text{Length}(C) + v\cdot \text{Area}(\text{inside}(C)) \\ + \lambda_1 \int_{\text{inside}(C)} |u_0(x,y) - c_1|^2 dxdy \\ + \lambda_2 \int_{\text{outside}(C)} |u_0(x,y) - c_2|^2 dxdy \end{aligned} \tag{2.8}$$

2.2.4.2 Level Set Method

ACWE can be solved using level sets. Equation 2.9 introduces ϕ as a function of the image, and defines the contour C as the points in the image Ω where $\phi = 0$. The points inside the resulting contour are denoted by ω.

$$\begin{cases} C = \delta\omega = (x,y)\epsilon\Omega : \phi(x,y) = 0 \\ \text{inside}(C) = \omega = \{(x,y) \in \Omega : \phi(x,y) > 0\} \\ \text{outside}(C) = \Omega/\omega = \{(x,y) \in \Omega : \phi(x,y) \leq 0 \end{cases} \tag{2.9}$$

Using the Heaviside function H, and the Dirac δ function, we can write F as in Equation 2.10 where u_0 is the image and c_1, c_2 are the averages as described in 2.11. μ,v and λ are positive parameters.

$$\begin{aligned} F(c_1, c_2, \phi) = \mu \int_\Omega \delta(\phi(x,y)|\nabla\phi(x,y)|dxdy + v \int_\Omega H(\phi(x,y))dxdy \\ + \lambda_1 \int_\Omega |u_0(x,y) - c_1|^2 H(\phi(x,y))dxdy \\ + \lambda_2 \int_\Omega |u_0(x,y) - c_2|^2 (1 - H(\phi(x,y)))dxdy \end{aligned} \tag{2.10}$$

where

$$\begin{aligned} c_1(\phi) &= \frac{\int_\Omega u_0(x,y)H(\phi(x,y))dxdy}{\int_\Omega H(\phi(x,y))dxdy} \\ c_2(\phi) &= \frac{\int_\Omega u_0(x,y)(1 - H(\phi(x,y)))dxdy}{\int_\Omega (1 - H(\phi(x,y)))dxdy} \end{aligned} \tag{2.11}$$

2.2.4.3 ACWE for Vector Valued Images

ACWE can easily be extended [Chan et al., 2000] to any size of vector for each pixel by averaging over the vector length. For example, over the three colours in RGB space. This is shown in Equation 2.12.

$$\begin{aligned} F(c_1, c_2, \phi) = \mu \int_\Omega \delta(\phi(x,y)|\nabla\phi(x,y)|dxdy \\ + v \int_\Omega H(\phi(x,y))dxdy + \lambda_1 \int_\Omega \sum_{i=1}^{3} |u_0(x,y) - c_{i1}|^2 H(\phi(x,y))dxdy \\ + \lambda_2 \int_\Omega \sum_{i=1}^{3} |u_0(x,y) - c_{i2}|^2 (1 - H(\phi(x,y)))dxdy \end{aligned} \tag{2.12}$$

2.2.4.4 Adaptation for Local Area

Chan-Vese segmentation of superpixels, shown in Figure 2.11, works by considering two regions u, v that form the positive and negative parts of a signed distance function, Ω_D. A force F iteratively updates the distance function (Equation 2.13) such that each pixel is 'moved' toward the region it best matches by adding the force F to the surface ϕ. The new superpixels, C_u, C_v, are taken to be the positive and negative parts of ϕ.

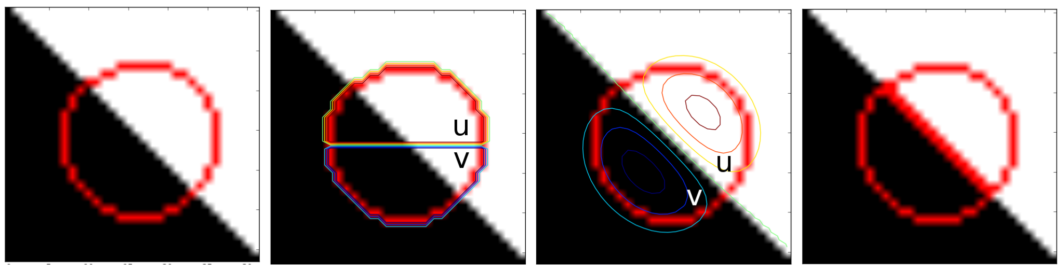
In this application, the problem is further simplified by considering only a subset of the image: the area within the superpixel. Considering these smaller regions makes the problem tractable as an iterative algorithm. This is achieved by the inclusion of a binary function $S(x, y)$ that is greater than zero when inside the superpixel and zero otherwise. In addition, the length and area constraints have been removed as the smaller area of the superpixel does not require them. Both weighting parameters λ are set to one to give equal weight to either side of the contour.

The final force equation is given in Equation 2.13, where t denotes the iteration of ϕ. The updated average calculation for $c_{1,2}$ is given in Equation 2.14, where the same constraint on $S(x, y)$ is applied.

$$\begin{aligned}
F(c_1, c_2, \phi) &= \int_\Omega \sum_{i=1}^{3} |u_0(x,y) - c_{i1}|^2 H(\phi(x,y)) S(x,y) dxdy \\
&\quad - \int_\Omega \sum_{i=1}^{3} |u_0(x,y) - c_{i2}|^2 (1 - H(\phi(x,y))) S(x,y) dxdy \\
\phi^{<t+1>} &= \phi^{<t>} + F
\end{aligned} \tag{2.13}$$

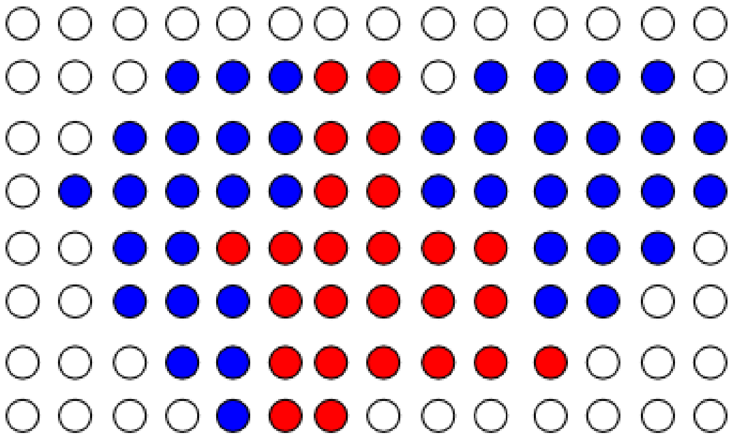
$$\begin{aligned}
c_1(\phi) &= \frac{\int_\Omega u_0(x,y) H(\phi(x,y)) S(x,y) dxdy}{\int_\Omega H(\phi(x,y)) S(x,y) dxdy} \\
c_2(\phi) &= \frac{\int_\Omega u_0(x,y)(1 - H(\phi(x,y))) S(x,y) dxdy}{\int_\Omega (1 - H(\phi(x,y))) S(x,y) dxdy}
\end{aligned} \tag{2.14}$$

To retain the property of spatial connectivity, there is one final problem to overcome. It is quite common in variational segmentation for the result to be spatially separated. In fact, it is often a desired result of the segmentation. However, this leads to problems when attempting to grow a superpixel that is not completely connected. Therefore, any parts of the segmentation that are not connected are allocated to separate superpixels using connected component labelling [Shapiro and Stockman, 2001] as shown in Figure 2.12. The red and blue regions correspond to two superpixels, where the blue superpixel needs to be split.

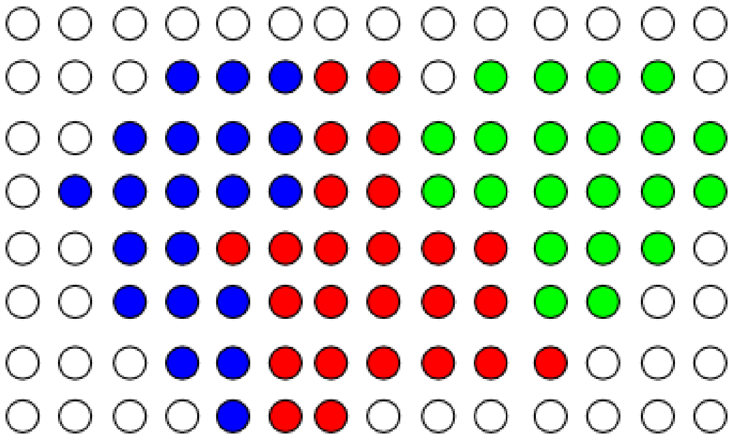


(a) Before (b) Contour plot showing the initialisation of the distance function (c) Contour plot showing the final distance function (d) After

FIGURE 2.11: Illustrating the mechanism of division. The superpixel is divided by 'moving' the distance function causing pixels to be either positive or negative.



(a) Before labelling



(b) After labelling

FIGURE 2.12: The split in the segmentation is corrected by using connected component labelling. The green superpixel is the new superpixel.

2.2.4.5 Choice of Initialisation

ACWE is initialised by setting the distance function ϕ to be a function in the range of ± 1. When considering only a small region of the image, it is expected that initialisation will be important, and that the faster it is possible to arrive at the best solution, the faster the algorithm will perform. To select the best initialisation, a set are tested on a modulated cosine signal (Figure 2.13(a)) to observe the amount of incorrect labelling.

As ACWE can be realised in N-dimensions, a 1D signal is used to observe the results in a concise way. The chosen initialisations to test are:

- $\phi[x] = 0$ (Figure 2.13(b));
- $\phi[x] = 1$ (Figure 2.13(c));
- $\phi[x] = -1$ (Figure 2.13(d));
- ϕ is alternately ± 1 (Figure 2.13(e));
- ϕ is the signal, normalised to the range ± 1 (Figure 2.13(f))

As the chosen signal varies uniformly around zero, the resulting segmentation should be to separate the signal into two signals either side of zero. The orientation of the result does not matter (some are flipped), what is important is that the two extracted regions (shown in different colours) correspond to all of the correct points in the signal. It is important to make the distinction that the values of ϕ are not the image values, they are the values of the distance function that are used to segment the signal. Figure 2.13 shows the results of this test where red circles denote error. The only two results without error are the cases where $\phi = 0$ and where ϕ relates to the signal. However, normalising the image over the range ± 1 will force large changes in ϕ that should not necessarily exist in like-pixels. This could force unnecessary superpixels and remove the smoothing effect of ACWE. Consequently, the initialisation will be $\phi = 0$.

2.3 Control

CDS is designed not to require controlling parameters as the idea is to create the best possible reconstruction of the image. However, one can still influence

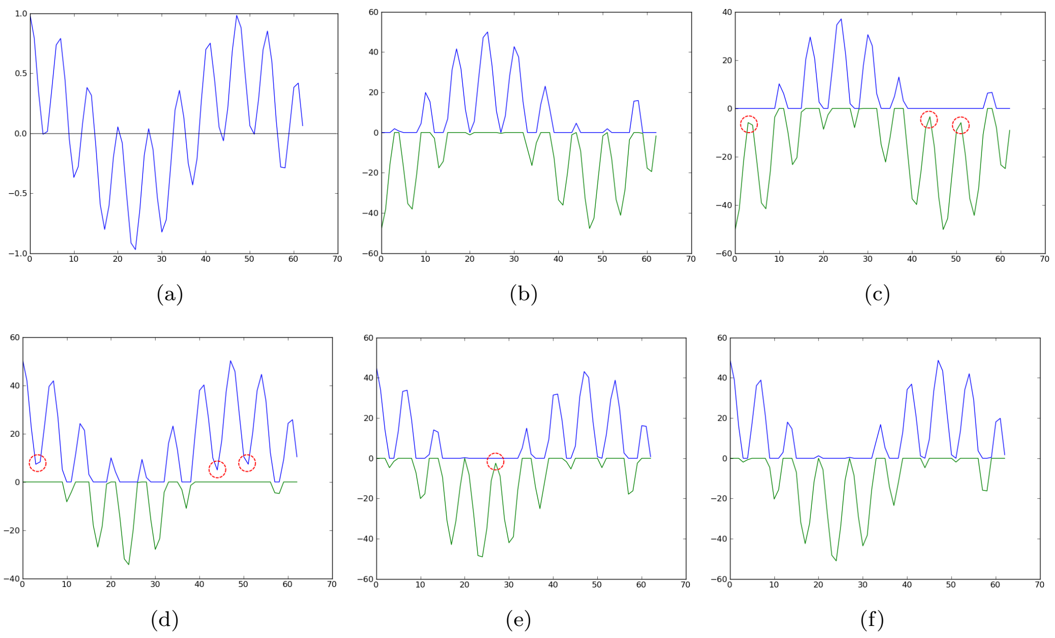


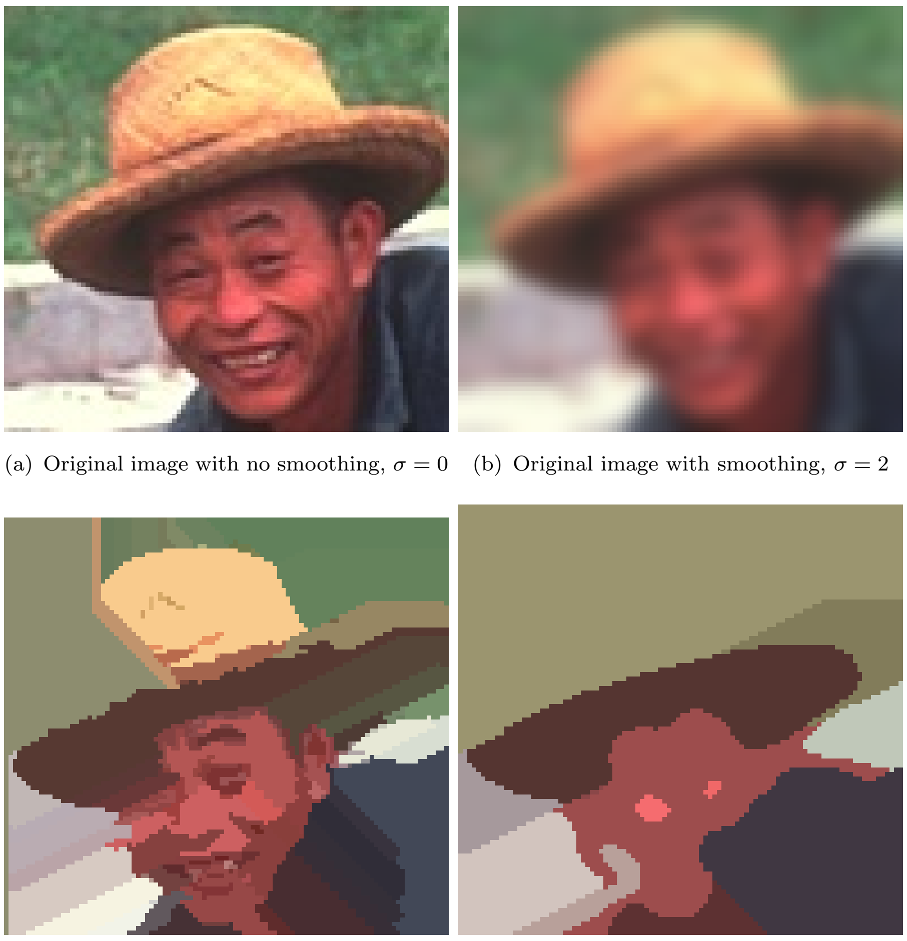
FIGURE 2.13: Investigating the effect of initialisation on ACWE

the result in the following two ways. Firstly, the level of detail detected by the algorithm can be controlled by smoothing the image using a simple Gaussian filter with standard deviation σ. This still retains the larger image variation however details such as facial features are missing. Figure 2.14 shows this effect. Adding the smoothing removes all facial features and treats the face as a single superpixel. Also, the brim of the hat still exists but the rest of it is merged with the grass behind it. The clothes are also merged into a single superpixel.

Secondly, the number of superpixels can be controlled by the initialisation of the superpixel seeds. The number of seeds is the minimum possible number of superpixels that are to be generated. This has a small effect on the result as the final shape and distribution might be different, however the same features will be detected regardless of the initialisation, and the reconstruction quality is unaffected.

2.4 Final method

By using a modified Distance Transform in conjunction with localised ACWE, there are numerous benefits. The Distance Transform allows implicit handling of an arbitrary number of superpixels of any size and any shape. Allowing them to grow unimpeded by image properties ensures total image coverage from any initialisation. New superpixels are handled from within the superpixel, as they


(a) Original image with no smoothing, $\sigma = 0$ (b) Original image with smoothing, $\sigma = 2$

(c) Reconstruction, $\sigma = 0$ (d) Reconstruction, $\sigma = 2$

FIGURE 2.14: Showing the effects of Gaussian based control

are formed when the superpixel is no longer uniform in colour, irrespective of size. The combination of these two algorithms is such that they can produce accurate superpixels that are stable and will not overlap under any condition.

A pseudocode implementation of the algorithm is given in Appendix B.

2.5 A Discussion on Region Merging

There is a temptation once the superpixels have been generated to remove the artificial boundaries produced where superpixel seeds meet. This could be achieved by merging them where the colour distance between neighbours is small. This is perhaps useful to further reduce the 'resolution' of the superpixels.

The ability to merge regions will not be developed for several reasons. The first

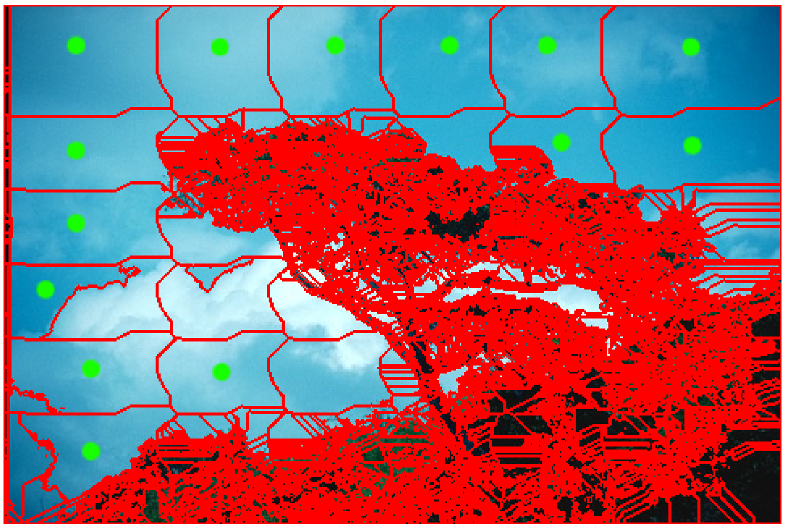


FIGURE 2.15: Borders that exist due to seeds meeting, with some of the superpixels that could be merged marked with green dots

is that if one reduces the number of superpixels, the superpixel representation no longer reflects the image content as there would have to be a threshold in order to merge superpixels. The second reason is that this would only really affect the borders between seed superpixels. Theoretically, borders only exist between distinct superpixels and that merging them again appears counter-intuitive to the original reasoning behind the algorithm. Removing borders between seed superpixels, shown in Figure 2.15, actually removes a very small percentage of superpixels. The final reason is that this step is tantamount to clustering, and if this is the desired effect then it could easily be achieved via k-means or other such algorithms, using the superpixels as a pre-processing step. An initialisation of one superpixel would solve this minor problem, but it will require additional processing time.

Figure 2.15 also shows that the superpixel growth algorithm causes anisotropic boundaries to occur. This is a consequence of the distance transform and does not affect the quality of the reconstruction so it was not altered to produce isotropic boundaries.

2.6 Analysis

2.6.1 Method of Analysis

Superpixels are only useful if they can capture relevant image information. As the focus is on image representation, evaluation must focus on evaluating the ability to reconstruct an image by losing as little information as possible. Reconstruction is defined as the process of replacing the colour of each pixel (x, y) with the mean of the superpixel $\mu_i(x, y)$ it is contained in, given in Equation 2.15

$$I(x, y) = \mu_i(x, y) \tag{2.15}$$

The results come from the test set from the Berkeley Segmentation Dataset (BSDS) [Martin et al., 2001]. BSDS includes human segmented annotations of the original images, typically five for each image. Given that the labels are much larger than a typical superpixel, each label will contain multiple superpixels. Mode label analysis is introduced in order to identify undersegmentation error, the average proportion of each superpixel that matches the modal annotated label. In addition, the percentage of label boundaries that match superpixel boundaries, the boundary recall rate, is computed. These are weighted such that borders included by all subjects are stronger than those occurring in one image. What is not computed is the boundary precision rate. This is the percentage of superpixel boundaries that match label boundaries. Using recall rate alone has been considered sufficient in previous research and gives a good result even if many superpixels edges do not occur at label edges. This is because the focus in previous research has been on avoiding undersegmentation rather than oversegmentation.

The use of metrics on this database is not sufficient. During the generation of BSDS, subjects were instructed to make sure all labels were of equal importance and size within the image. Consequently, small detail could easily be falsely attributed to incorrect or insufficient labelling. To provide a measure independent of the human labels, the 'explained variation' Moore et al. [2008] is calculated, providing a measure of superpixel accuracy, which helps to evaluate superpixels that do not correspond to the human-labelled edges. It calculates how accurately the mean of each superpixel matches the pixels within by calculating the variation about the global mean. This is given in Equation 2.16, where x_i represents the pixel value, μ_i is the mean of the superpixel containing the pixel x_i and μ is the global mean of the image.

$$R^2 = \frac{\sum_i(\mu_i - \mu)^2}{\sum_i(x_i - \mu)^2} \tag{2.16}$$

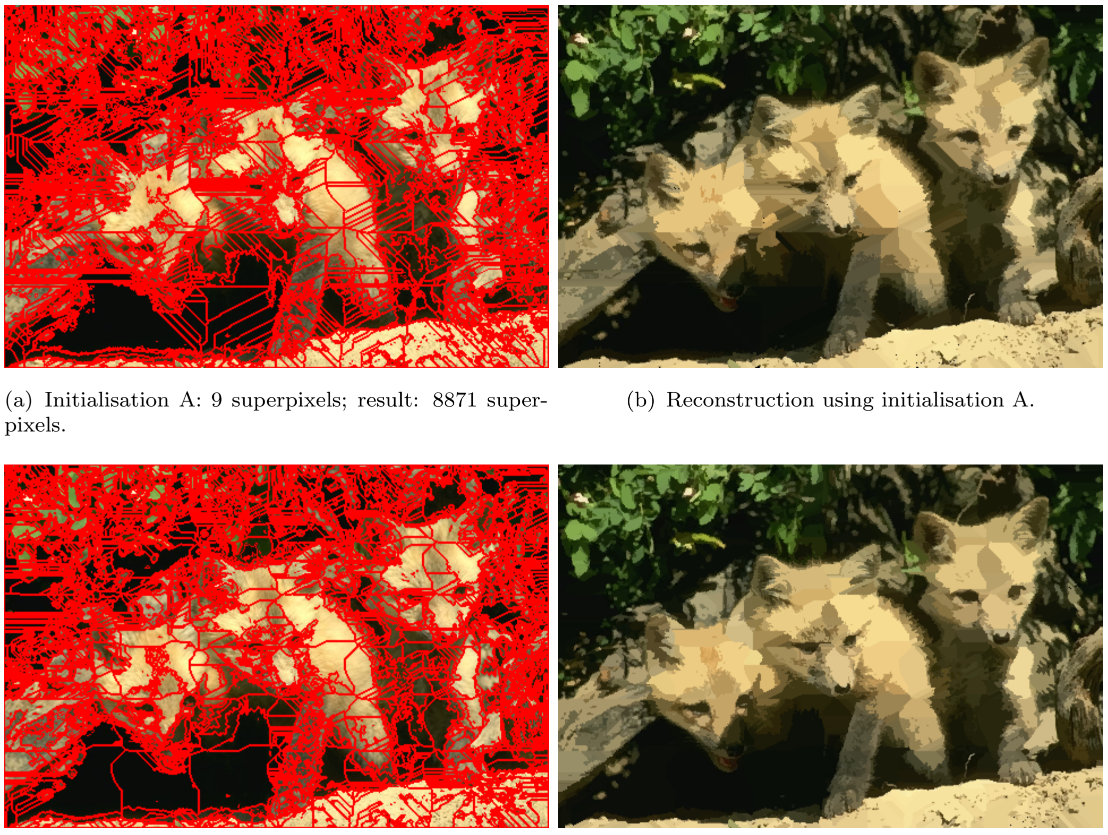
Explained variation can be considered a measure of undersegmentation. If the average colour of a superpixel does not accurately represent the pixels within it, that region is undersegmented.

Many analysis techniques, such as explained variation, favour more rather than less superpixels. While it is recognised that this is often not the main aim of superpixels, results can be improved if more superpixels are present to capture the higher levels of image variation. The extreme of this being of course the case where each superpixel represents a single pixel. Taking oversegmentation into account removes the emphasis on just creating more superpixels to capture more information. The emphasis is then on creating superpixels to capture more information only when required to do so by the image properties. As such, oversegmentation can be considered analogous to a measure of superpixel precision. As CD superpixels split on colour differences, oversegmentation can be measured by the Euclidean distance between the mean colour value of each connected superpixel averaged over all connections. If this value is low, then the average distance between superpixels is small, and therefore superpixels are less distinct in colour, implying that oversegmentation exists. This measure is given in Equation 2.17 where $(r, g, b)\epsilon[0, 1]$ represent the colour of connected superpixels i, j. C represents the sum of all superpixel connections and c represents a single connection.

$$\mathrm{d} = \frac{\sum\limits_{c} \sqrt{(r_i - r_j)^2 + (g_i - g_j)^2 + (b_i - b_j)^2}}{C} \tag{2.17}$$

The results generated in this chapter are compared against the algorithms in [Felzenszwalb and Huttenlocher, 2004](FH) and [Ren and Malik, 2003](N-Cuts) as these are well established techniques. As our new algorithm does not directly control the number of superpixels, all comparisons are achieved by using the output from CD superpixels to specify the equivalent parameters in the other algorithms. To assess the quality of our algorithm, results are generated on each image using varying levels of Gaussian smoothing. This allows a comparison to be drawn as the number of superpixels changes.

Finally, the compression ratio is defined as the ratio of pixels to superpixels. Plotting this ratio instead of the explicit superpixel numbers helps to put the result into perspective irrespective of the image size. A low compression ratio indicates



(a) Initialisation A: 9 superpixels; result: 8871 superpixels.

(b) Reconstruction using initialisation A.

(c) Initialisation B: 36 superpixels; result: 7248 superpixels.

(d) Reconstruction using initialisation B.

FIGURE 2.16: Illustrating the difference between two different initialisations arranged in an evenly spaced grid. Despite the difference in the initialisation the reconstruction is hardly affected.

a high number of generated superpixels.

As shown in Figure 2.16, even though the output can be directed by the initialisation, the reconstruction is largely unaffected despite the difference in resulting superpixels. In addition, all results are similarly affected by smoothing the image. For this reason, the initialisation will not be changed while testing the effects of smoothing. However, the invariance to initialisation is tested. This invariance is tested on a single image using random Gaussian perturbations of the grid pattern at varying levels of standard deviation.

The algorithm is also assessed when introducing increasing amounts of Gaussian noise.

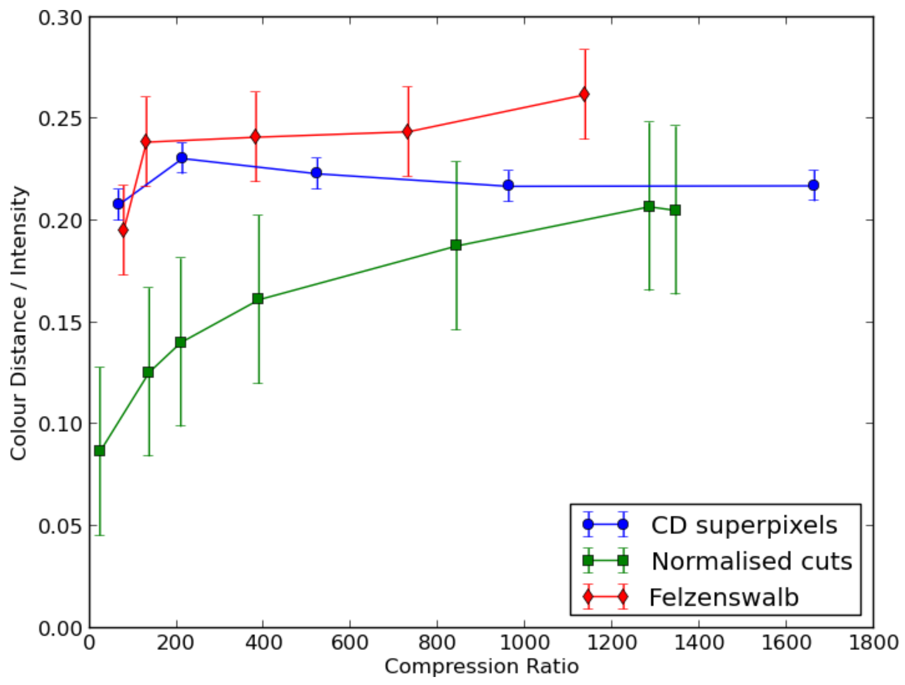


Figure 2.17: Showing the difference in colour between superpixel neighbours as a function of superpixel compression.

2.6.2 Results

Figure 2.17 shows the colour difference between neighbouring superpixels as a function of the number of superpixels in the image over the test set of images from BSDS. A lower value indicates that the colour difference is smaller, or in other words, that the regions are more likely to be oversegmented as two neighbouring superpixels could be represented as one. N-cuts performs badly on this test because as the number of superpixels increases, the oversegmentation increases. This is not unexpected as superpixels generated in this way are designed to be of similar size, meaning that large regions of one colour will contain the same number of superpixels as a much more complex region. CD superpixels, however, has an almost uniform response despite the number of superpixels. This means that as the number of superpixels increases and the representation of the image improves, the colour difference between superpixels remains constant. When specifying lower numbers of superpixels for N-Cuts and FH, the latter performs best. Again this could be due to the implicit nature of superpixels generated by Felzenswalb.

Figure 2.18 shows the relationship between oversegmentation and undersegmentation. N-cuts and FH both show, to varying degrees, that as the explained variation of the result increases (how well is the image reconstructed by superpixels), the difference in colour between those superpixels decreases, leading to oversegmentation.

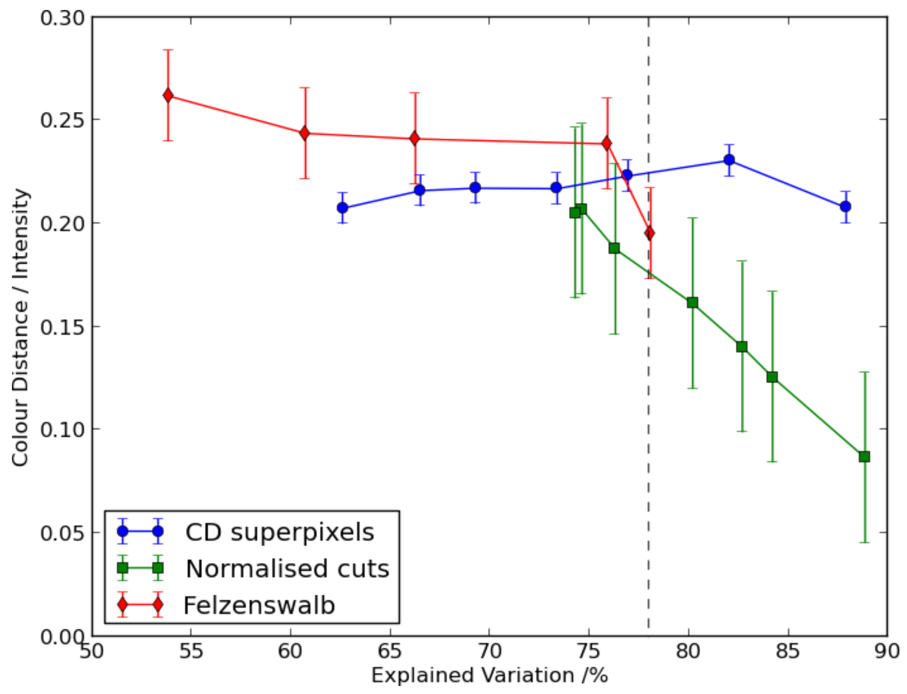


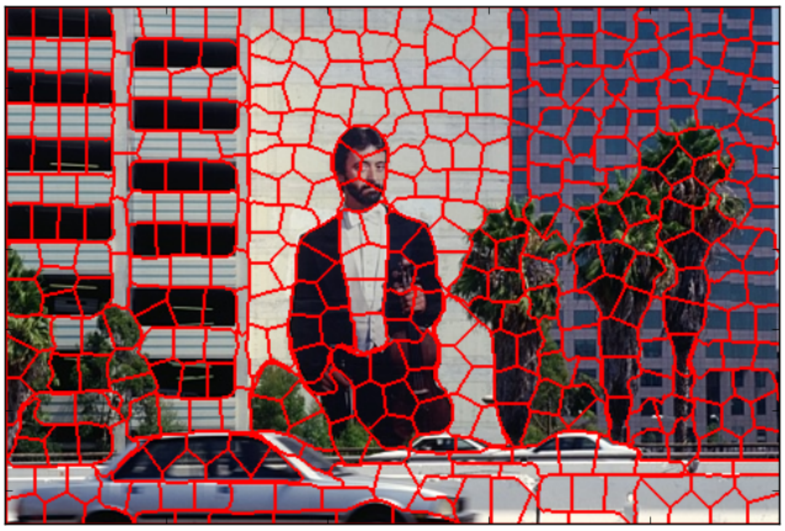
FIGURE 2.18: Showing the difference in colour between superpixel neighbours as a function of explained variation.

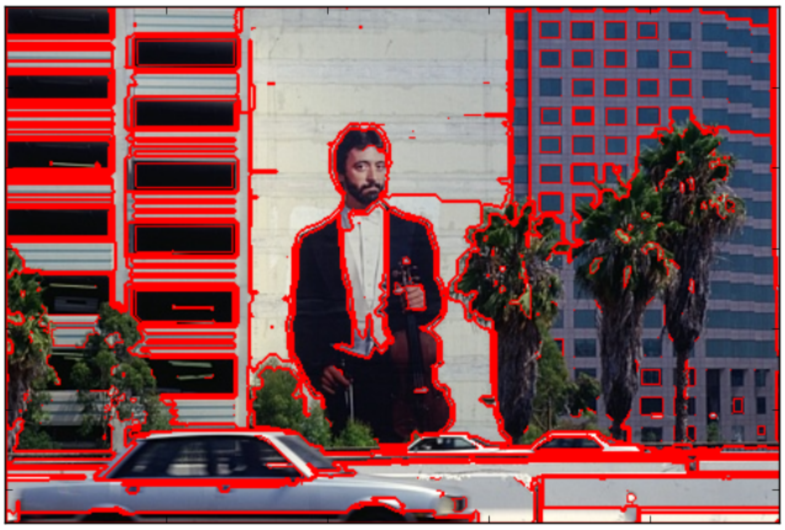
CD superpixels remains almost constant, irrespective of the quality of reconstruction. This means that increasing the quality of result for CD superpixels does not lead to large amounts of oversegmentation as would normally occur.

A sample image is taken from all three algorithms at the same Explained Variation value denoted by the dotted line in Figure 2.18. These three images are shown in Figure 2.19. N-Cuts clearly contains too many superpixels, particularly in the central area surrounding the person. FH does a better job in the centre of the image. The windows in the right hand tower are well detected, yet the trees contain significant undersegmentation, as does the person in the centre. CDS performs well, particularly in the trees and the person in the centre yet, like N-Cuts, does not detect the windows on the right.

Figure 2.20 illustrates three metrics: recall rate; explained variation; and mode label. In Figures 2.20(a) and 2.20(b), CD superpixels perform well, but only at high numbers of superpixels (low compression). Recall rate (Figure 2.20(c)) clearly shows N-cuts to perform best by almost 20% at low compression. Soon after, all three algorithms are largely equivalent until high compression at which point CD superpixels suffers.

The ability of CD superpixels to provide a constant measure of oversegmenta-


(a) N-Cuts


(b) FH

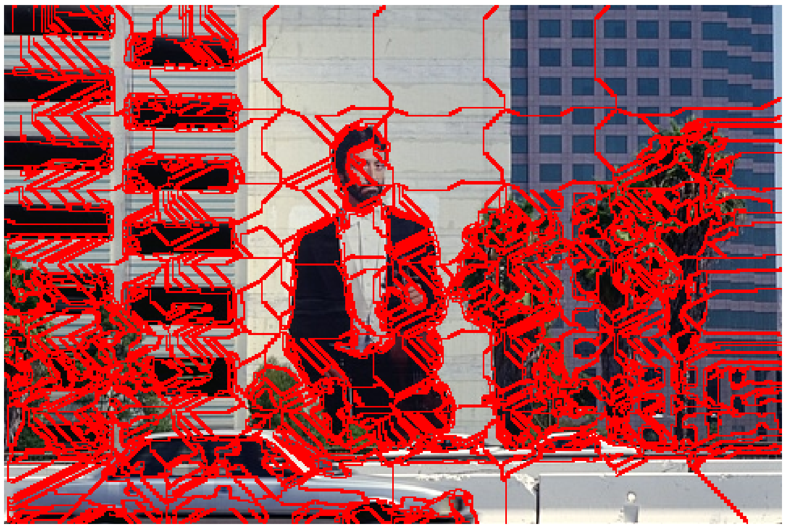

(c) CDS

FIGURE 2.19: Example images all shown at the same value for Explained Variation shown in Figure 2.18

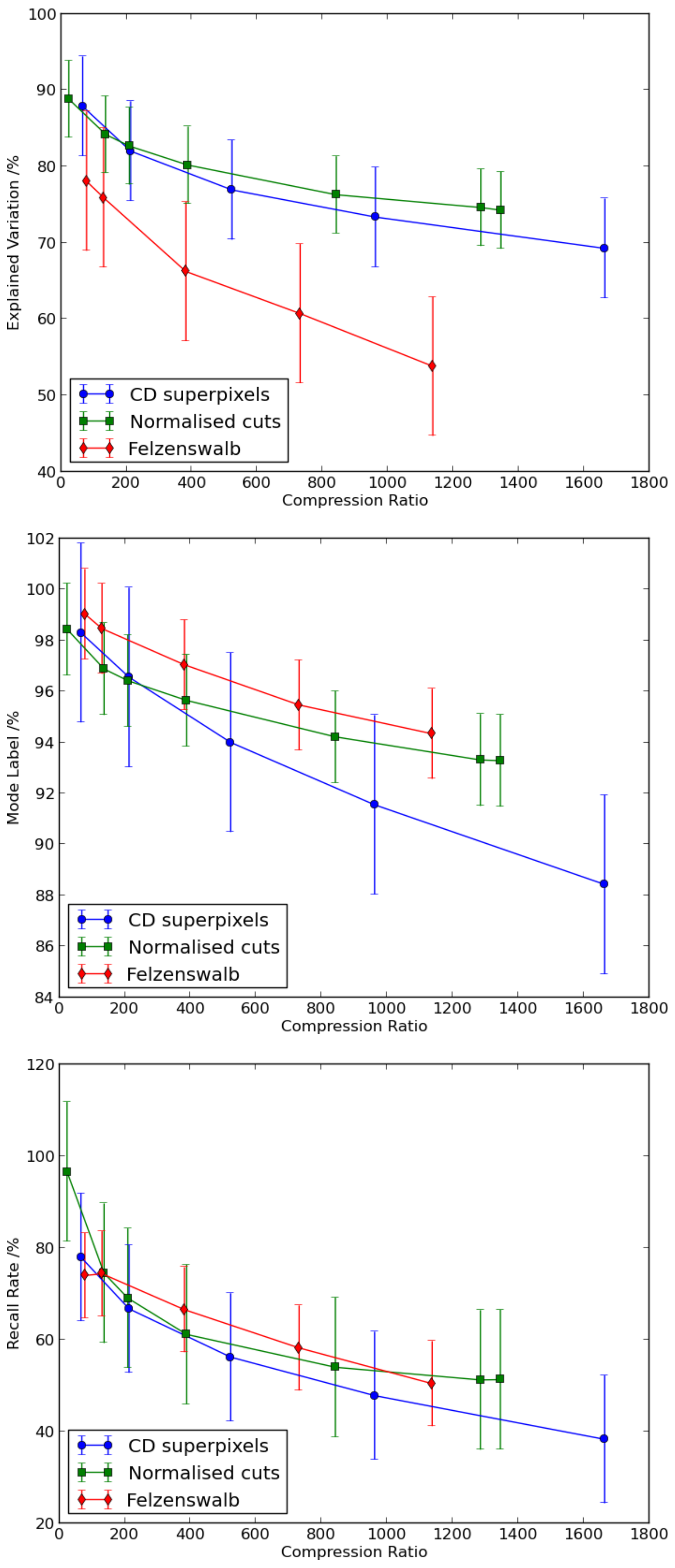


FIGURE 2.20: Showing how recall rate, mode label and explained variation vary as a function of compression

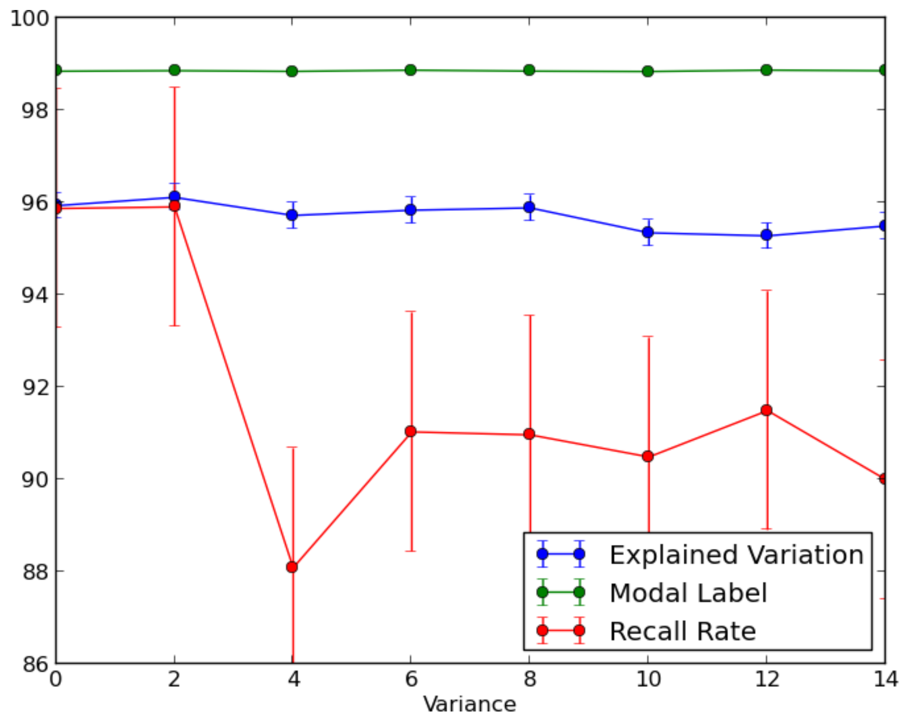


FIGURE 2.21: Showing how recall rate, mode label and explained variation all vary as the initialisation is perturbed by a Gaussian random variable of standard deviation σ.

tion throughout is an interesting property of the algorithm. It is thought that it can be explained by the use of ACWE as a splitting algorithm. As ACWE uses colour differences to divide, and almost all neighbouring superpixels have occurred through division, the constant difference in colour must be attributed to how ACWE separates a region. The benefit of this is improved stability when generating superpixels.

CD superpixels clearly suffer with regard to recall rate. As the control mechanism comes from smoothing the input image, the edges and detail in the image degrade with more smoothing. This reduces the likelihood of an image boundary matching a superpixel boundary. This problem occurs for all metrics plotted in Figure 2.20 however colour difference does not suffer for the same reason described above.

One further important point is that the human labelling used for recall rate and mode analysis is subjective. Labels are drawn on the assumption that they are equally important to the user. While this is partially accounted for by the averaging, some important image information is ignored. For this reason, superpixel quality should not be exclusively assessed by low-resolution labelling. The use of explained variation is intended to address this issue.

Figure 2.21 shows the results of perturbing the initialisation by a random Gaussian variable of increasing variance. Explained variation and modal label vary little, having standard deviations of 0.3 and 0.01 respectively. Recall rate is the only metric that varies by more than one percentile at 2.4. This is still a small variation and is attributed to one result only, at $\sigma = 2$. Colour difference is not assessed as the previous experiment has shown it is almost constant.

The results in Figure 2.21 show there is high stability in the algorithm. As CD superpixels are parameterised on image properties, the number of superpixels is forced to adhere to the image and consequently, there is little possibility of the superpixel arrangement deviating. This helps to resolve one major problem of superpixel algorithms: that they are unstable due to initialisation parameters.

Figure 2.23 shows the effect of noise on the algorithm. The noise was generated by using a Gaussian random variable of increasing variance centred on the value of each pixel. This will make the image tend more toward a completely random image where no structure is present. The variance is controlled between $\sigma = 10$ and $\sigma = 70$ such that the image is still at least partially visible.

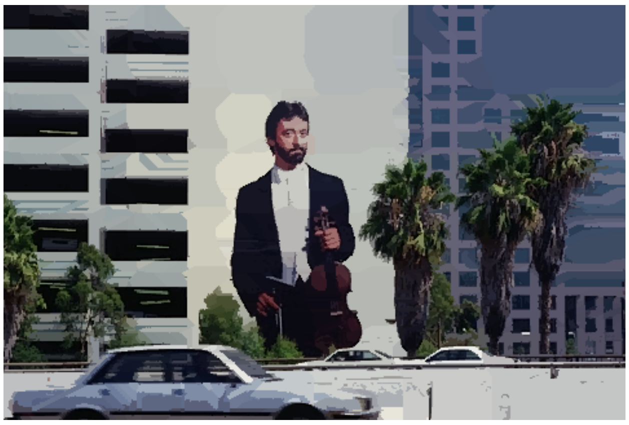
This shows that the reconstruction accuracy reduces as the image tends toward being completely random.

One would expect that as the image becomes less structured that it is more difficult for CDS to extract regions of uniformity. CDS has an inherent averaging process, which in most images has little effect. Using images where the colour changes frequently and unpredictably clearly affects the reconstruction quality. However, the explained variation is still 75% in the presence of significant noise. This can be seen in Figure 2.22 where the visual quality of the result appears to have been improved by the presence of noise. The reason for this is that there are more superpixels triggered by the presence of more variance within local areas. So the result is more visually pleasing, yet the oversegmentation has been increased.

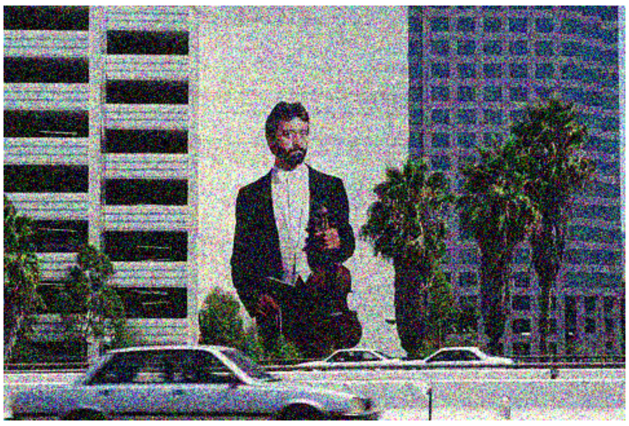
2.7 Conclusion

This chapter has developed and demonstrated an algorithm that can successfully and, more importantly, reliably reconstruct an image using superpixels. The results also show that the instability of previous superpixel algorithms has been reduced by parameterising superpixels not by number but by image complexity.

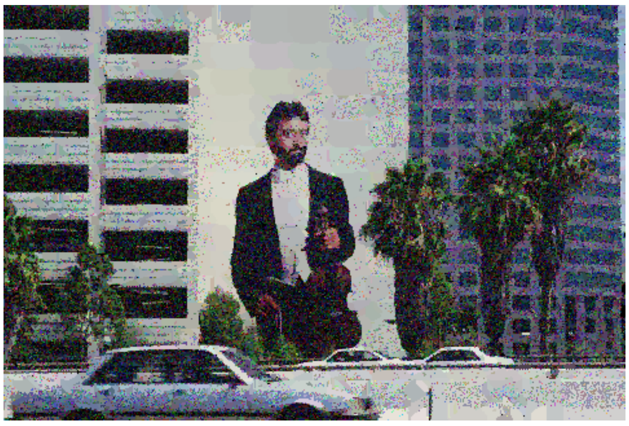
(a) Image at $\sigma^2 = 0$



(b) Reconstruction at $\sigma^2 = 0$



(c) Image at $\sigma^2 = 70$



(d) Reconstruction at $\sigma^2 = 70$

FIGURE 2.22: Examples of the effect of noise on an image and its reconstruction.

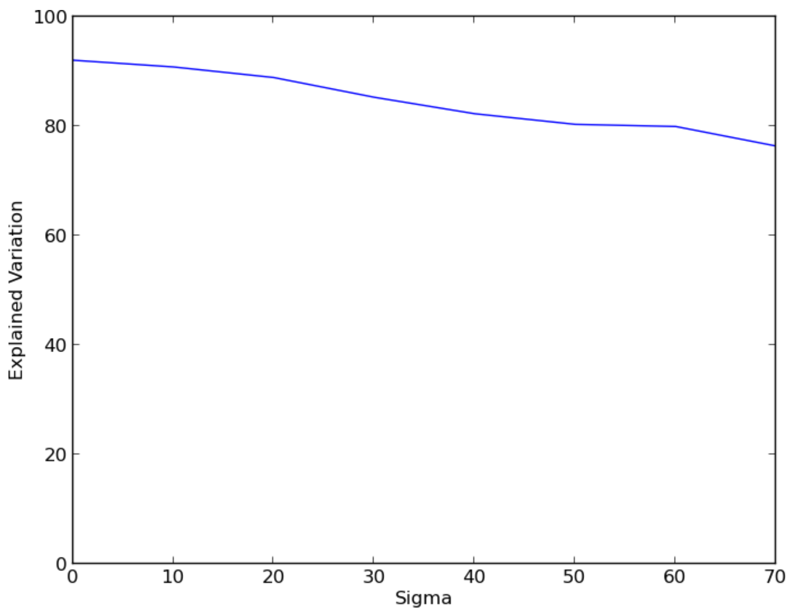


FIGURE 2.23: Quality of reconstruction as noise increases

This improves the invariance to initialisation as all metrics used vary by less than 3%.

The performance of Content-Driven Superpixels is dependant on image complexity. The more complex an image is the better the performance of CDS. This might seem counter-intuitive yet it is linked to the use of ACWE as a division method. As the colour information within the superpixel is averaged, bigger superpixels are more likely to contain larger colour differences. A smaller superpixel is more sensitive to small variation in colour as it makes a larger contribution to the average. This averaging also causes problems if two areas vary significantly yet their means are similar. This manifests itself when dealing with noisy images, as shown in Figure 2.23. Superpixels will not divide under this circumstance and this reduces the reconstruction quality.

In addition, this chapter developed the previously overlooked concept of explicitly measuring oversegmentation to better evaluate superpixels. It was subsequently shown that CDS oversegments less than other well-used algorithms.

In general, superpixels remain to have their uses truly explored. The ability to reduce image complexity whilst retaining high-level features is highly desirable in many areas of computer vision and the next few chapters explore some potential applications.

Chapter 3

Applications

3.1 Introduction

This chapter explores the use of superpixels in three established areas of computer vision:

- Segmentation;
- Compression;
- Scene Categorisation.

These areas are chosen because they are from very different areas of computer vision and are therefore intended to show the versatility of content-driven superpixel pre-processing.

3.2 Converting Superpixels into a Graph

A graph is defined as follows. Let the graph $G = \{V, E\}$ where V represents the vertices of the graph and E the edges. The vertices and edges are defined as the superpixels and their local connections, respectively.

This is illustrated in Figure 3.1. The vertex colour and location represent the mean and centre of mass of the associated superpixel. As superpixels generated using CDS are not uniform in size, the distance between each centre of mass is dependent on the size of each pair of superpixel neighbours.

Converting superpixels into this form enables local structure to be observed, as well as allowing dynamic re-structuring of the superpixels. For instance, removing a superpixel from the graph, and therefore the image, based on certain properties.

3.3 Segmentation using CDS

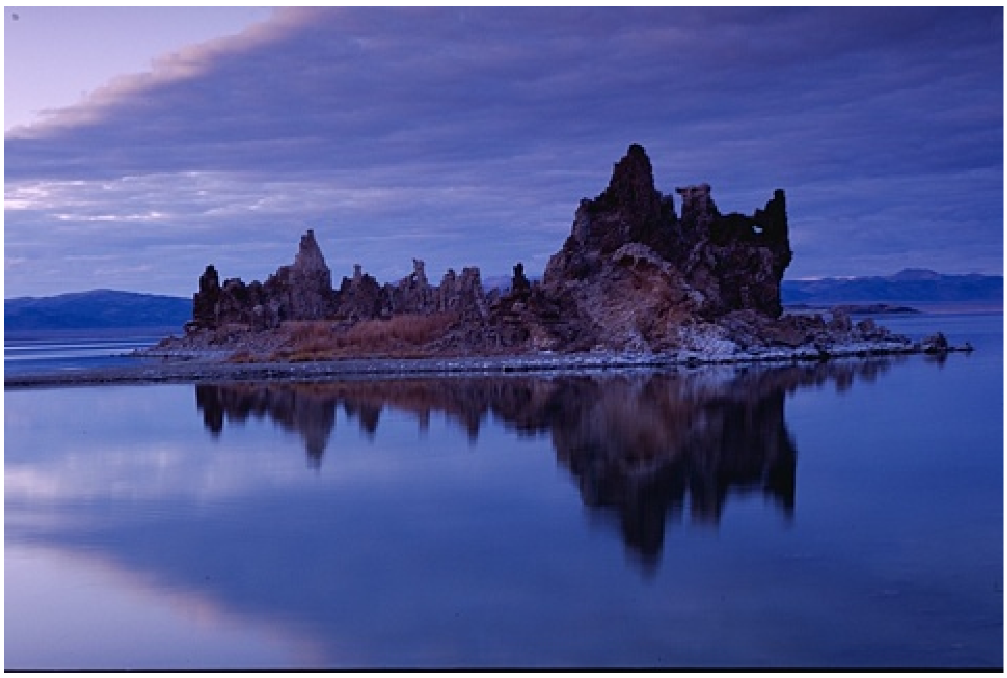
3.3.1 Introduction

The aim of segmentation is to determine image structures such as separating an image into its foreground and background components, for example locating a person. One method of segmentation is to use pairwise pixel comparison methods. This becomes prohibitively expensive on large images as the number of vertices increases [Ren and Malik, 2003]. Using superpixels, however, allows much larger images to be segmented in a given time frame.

CDS only produces edges between regions of different colour. This means that, for segmenting images, most of the information required is already embedded in the superpixel structure. [Rohkohl and Engel, 2007] used a similar superpixel approach, however they used N-Cuts superpixels which are designed to produce superpixels of the same size. By using CDS, the distance between superpixels is not uniform and this information can be exploited to improve segmentation results. Using this information gives significant improvement compared with using N-Cuts. An alternative approach is described in [Fulkerson et al., 2010] that uses histograms of local features to classify superpixels.

3.3.2 Method

As superpixels remove the traditional pixel neighbourhood, the graph defined in Section 3.2 is required to show the structure of an image. While being something of a necessity, it also facilitates the use of additional algorithms which could not be used with a pixel structure due to the computational cost of having one vertex per pixel.


(a) Image

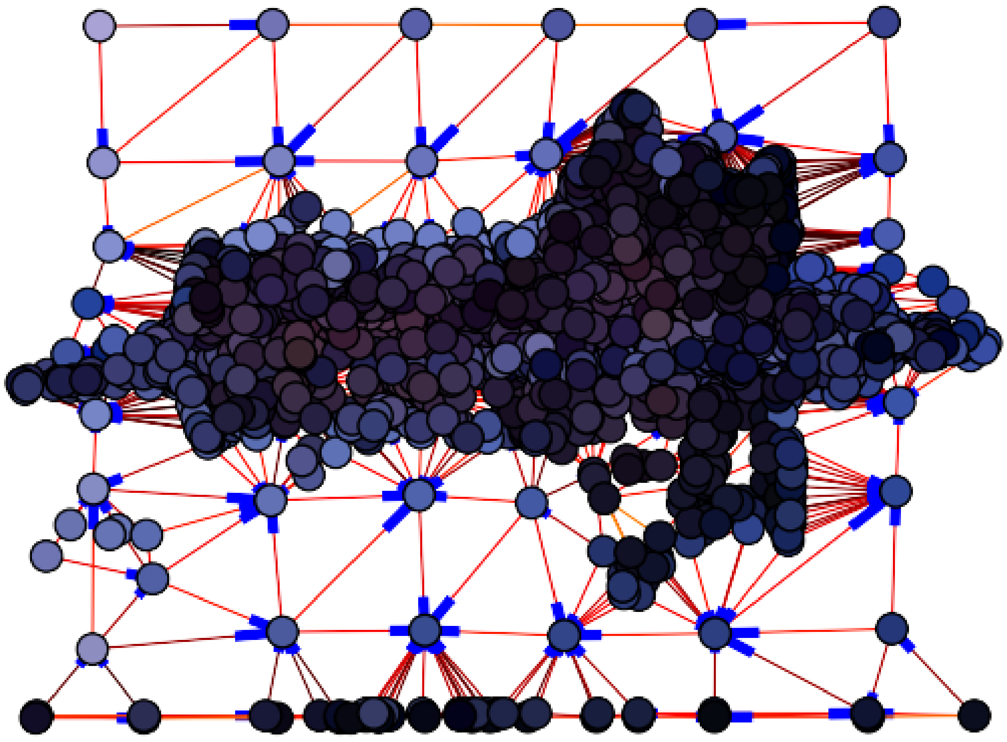

(b) The associated graph

FIGURE 3.1: Illustrating the superpixel graph

To segment the image using the graph, the graph edges are weighted based on the proximity of two superpixels and the colour separation between them. This is shown in Equation 3.1. The parameter λ is a control parameter that can 'switch off' either the spatial (x, y) or colour (r, g, b) components of the weighting. For these experiments the parameter was set to $\lambda = 0.5$ to give equal weight to both colour and spatial components.

$$\begin{aligned} W(i,j) &= \lambda e^{-\sqrt{(x_i-x_j)^2+(x_i-x_j)^2}} \\ &+ (1-\lambda)e^{-\sqrt{(r_i-r_j)^2+(g_i-g_j)^2+(b_i-b_j)^2}} \end{aligned} \tag{3.1}$$

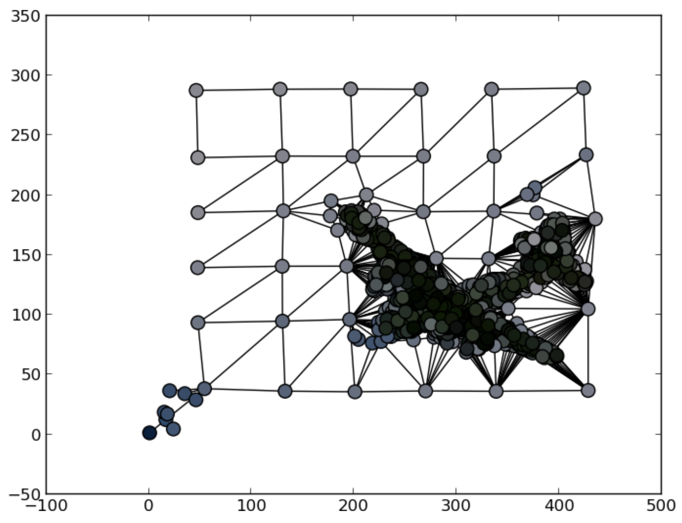
These weights can then be thresholded, where any weight below the threshold t will be treated as an edge to remove, as shown in Figure 3.2. The remaining regions form the segmentation. This is demonstrated in Figure 3.3 where each cut edge is drawn to represent the segmentation boundary. Clearly, $t = 0.25$ is too low as few pixels remain and $t = 0.5$ is too large as superpixel boundaries remain.

The algorithm is improved by using the principles of hysteresis thresholding [Canny, 1986]. This works by applying two thresholds to the graph. The lower threshold is applied to each edge first, then any graph edges that are connected to edges above the lower threshold are also cut if they do not exceed the higher threshold. This allows more edges to be present in the segmentation whilst also removing noise. Figure 3.4 illustrates this. Figure 3.4(a) shows the cut (in red) where the edges are lower than the threshold $t = 0.3$. By including the higher threshold in Figure 3.4(b), two additional edges are included, but this does not include every edge below the upper threshold. Note that the edge weights are not calculated for this example, they merely serve as an illustration.

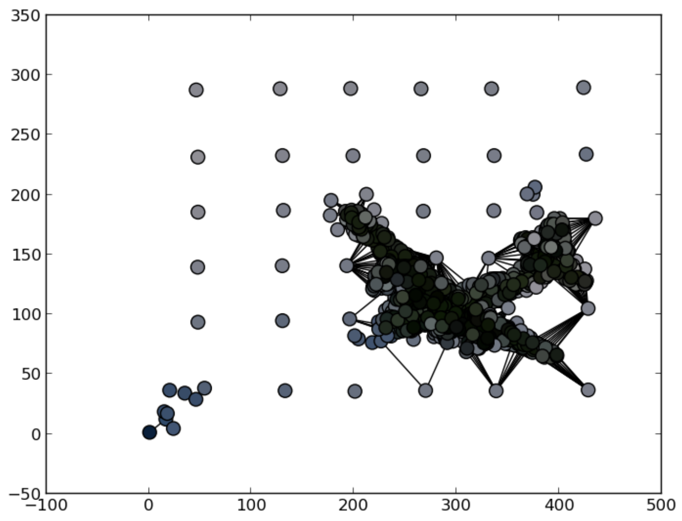
Figure 3.5 shows the benefit in using hysteresis. There is significantly less noise after applying thresholding.

3.3.3 Results

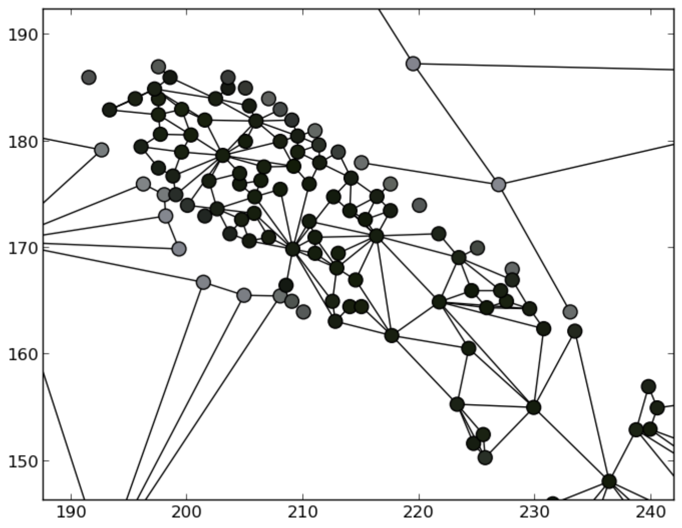
The database used is the BSDS database as it includes human segmentation collected over multiple subjects, shown in Figure 3.6. The analysis will focus on determining the co-existence of lines in the human segmentation and the superpixel segmentation, which is the recall rate of the manual segmentation. This



(a) All edges are present

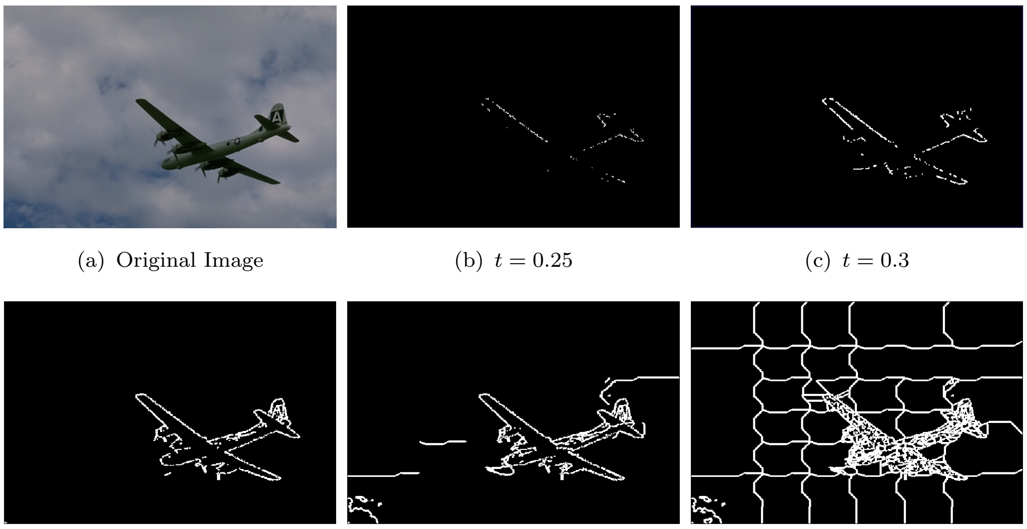


(b) Showing only the ones that are to be cut



(c) Zoom on the plane wing, showing the effective segmentation by removing the edges in Figure 3.2(b)

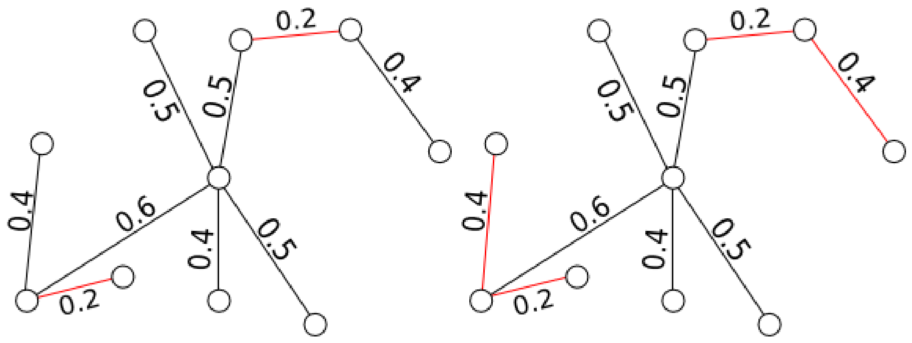
FIGURE 3.2: Illustrating the edges that are cut



(a) Original Image (b) $t = 0.25$ (c) $t = 0.3$

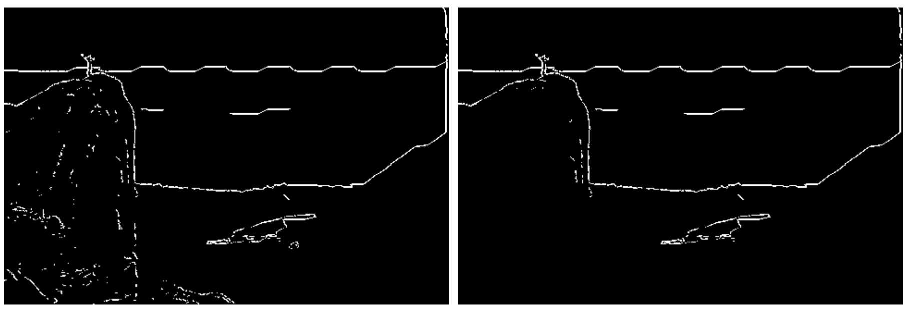
(d) $t = 0.4$ (e) $t = 0.45$ (f) $t = 0.5$

FIGURE 3.3: Showing the segmentation result with varying thresholds



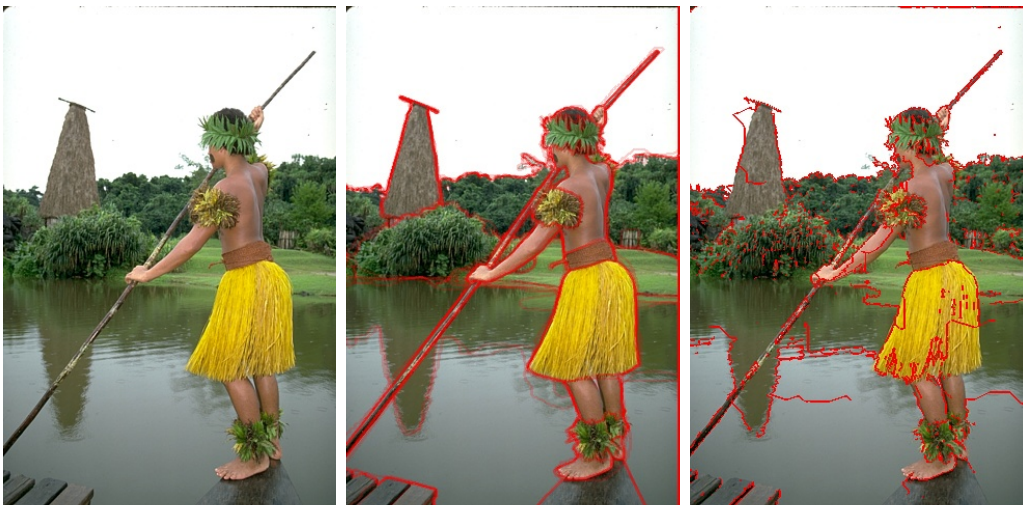
(a) $t = 0.3$ (b) $t_H = 0.45, t_L = 0.3$

FIGURE 3.4: Cutting the superpixel graph using hysteresis



(a) before (b) after

FIGURE 3.5: Showing the benefits of hysteresis



(a) Original (b) Human segmentation (c) CDS segmentation

FIGURE 3.6: Showing the test procedure. Figures 3.6(b) and 3.6(c) are compared.

test will determine if the information from CDS is better than just using pixel information for image segmentation, both in speed and quality. Other superpixel algorithms are also used for comparison.

The control parameter λ is varied to show the benefit of using a combination of colour and spatial information. As in [Ren and Malik, 2003], there is also a tolerance factor n included that determines if there is an edge detected within n pixels of the manual boundary. This shows how accurately the boundaries are being detected.

Figure 3.7 shows the results when considering only the colour component of the cutting criterion. Although CDS consistently out-performs N-Cuts, the quality is poor, as expected, though with greater threshold values the results improve slightly. Conversely, Figure 3.8 shows the results when considering only the spatial component of the cutting criterion. This graph shows that the spatial component alone is not suitable for segmentation. Despite having very low thresholds, most of the edges are cut and treated as contributing to the segmentation. Clearly this is of little use. As expected, N-Cuts detects no edges using this information alone. As N-Cuts does not vary spatially, there is little to suggest that it would have worked.

Figure 3.9 shows the effect of combining the two components. By reducing the effect of the spatial component and including the discrimination ability of the colour component the result is that we can achieve higher accuracy and a highly

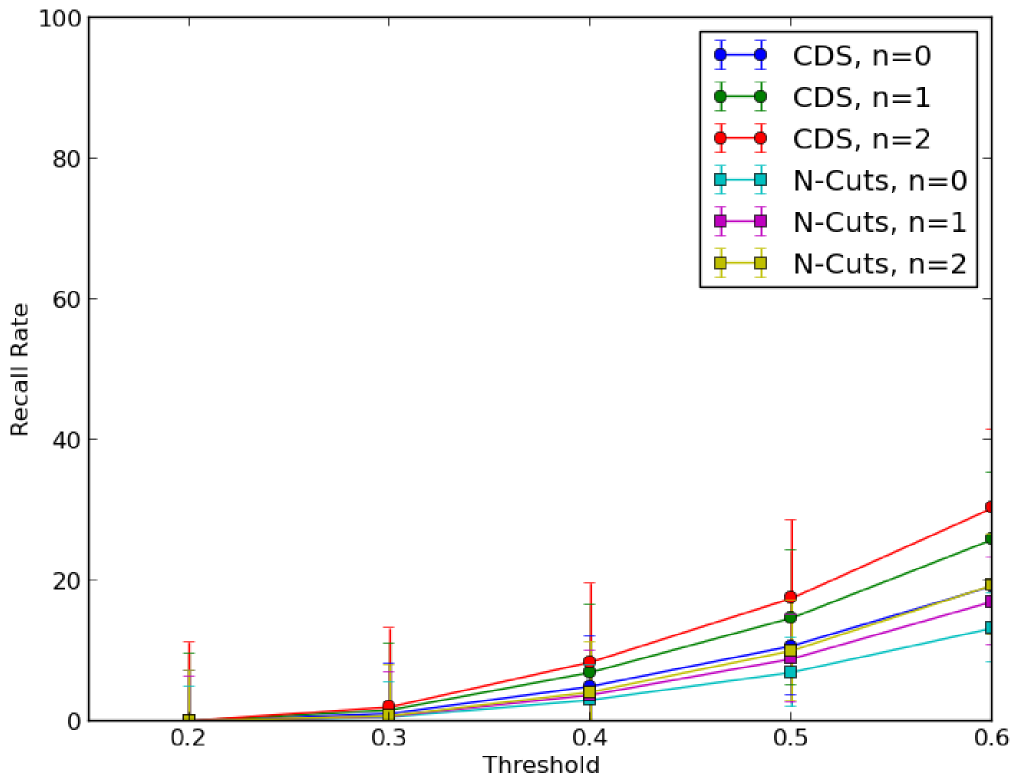


FIGURE 3.7: Comparing colour components of CDS with N-Cuts for varying thresholds, $\lambda = 0$

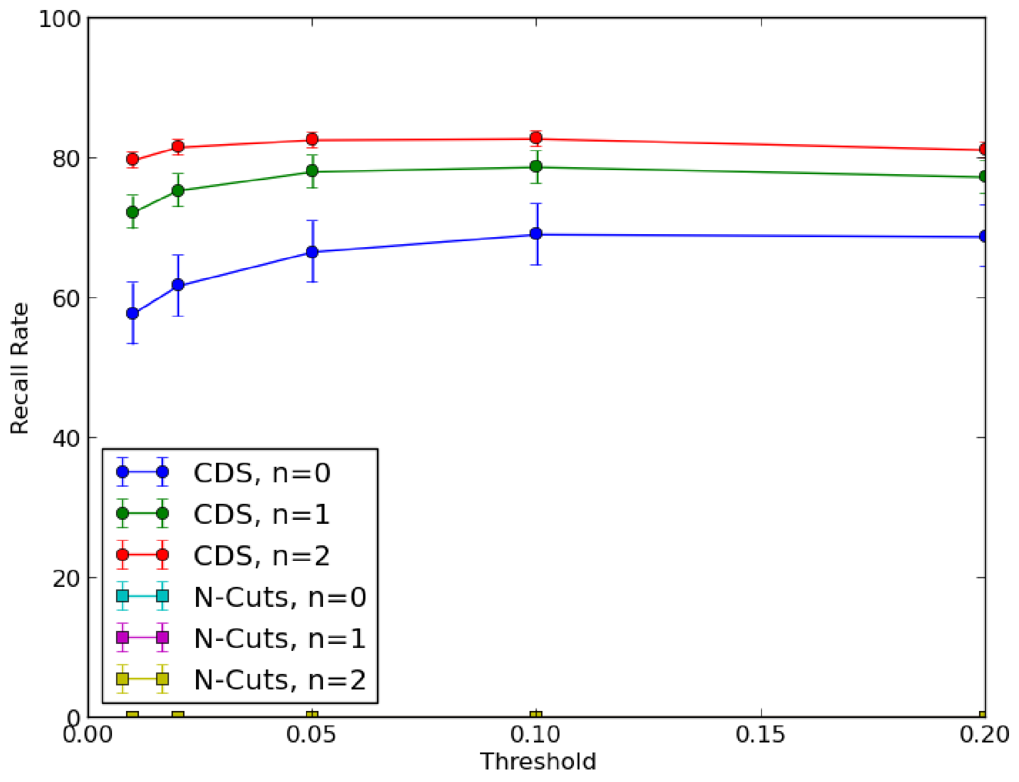


FIGURE 3.8: Comparing spatial components of CDS with N-Cuts for varying thresholds, $\lambda = 1$

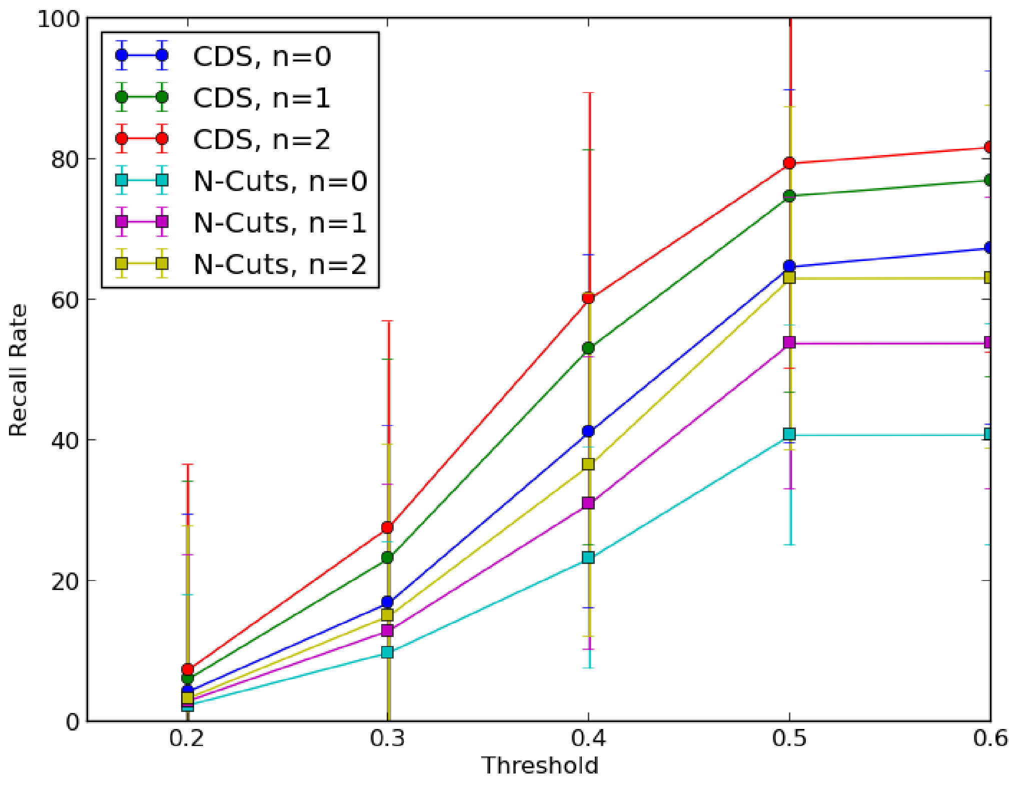


FIGURE 3.9: Comparing CDS with N-Cuts for varying thresholds, $\lambda = 0.5$

tunable algorithm. In addition, we can show improved performance over the whole dataset. Even when N-Cuts includes a greater tolerance, it performs worse than CDS with zero tolerance.

Figure 3.10 shows the results of a separate experiment performed on a graph where each node is an individual pixel, to show the baseline performance. Only colour information is used as including the spatial information gives no result. Having taken significantly longer to generate, the results on the pixel graph are still worse than CDS.

3.4 Image compression using CDS

3.4.1 Introduction

Superpixels and compression are naturally related, though there has been little study of their inter-relation. This section introduces the concept of superpixels in an image compression framework. As CDS can be applied to images without much loss of information, it is ideal in such an application. By creating superpixels, the

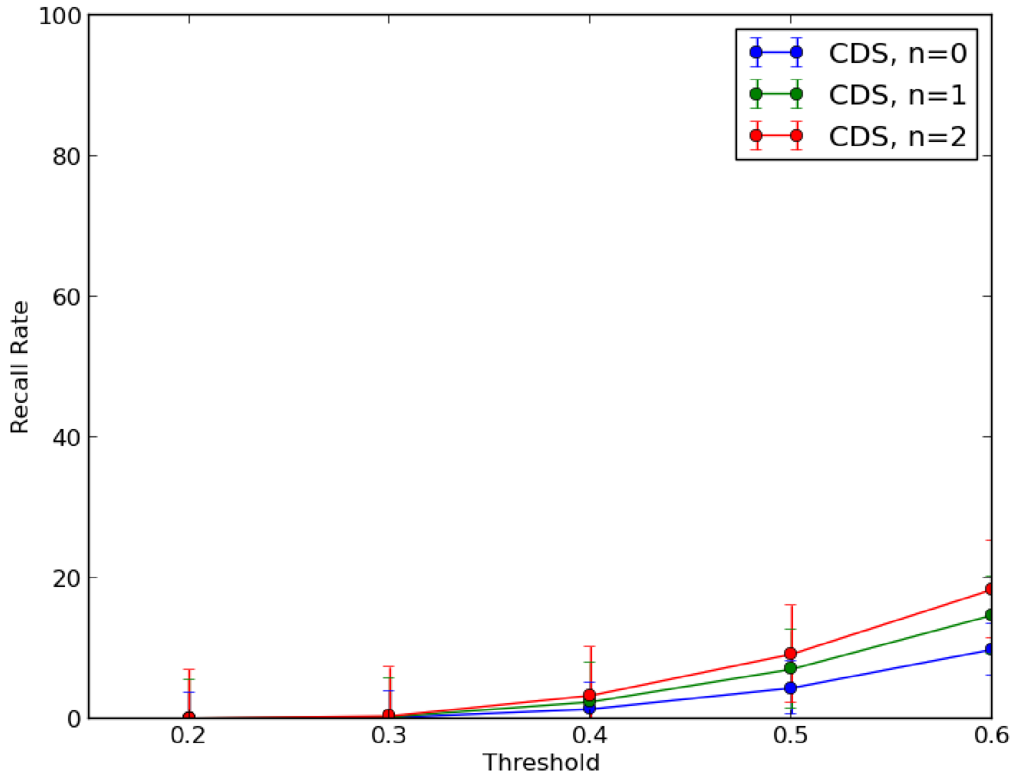


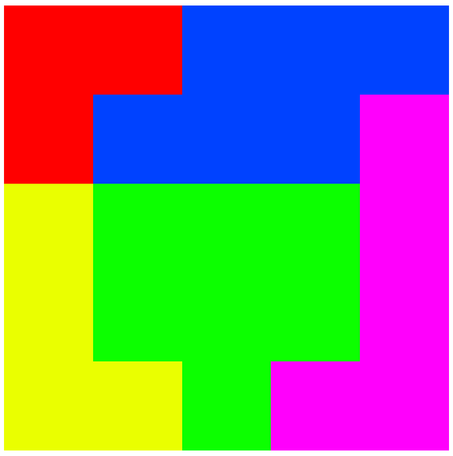
FIGURE 3.10: Showing the results on a pixel graph, $\lambda = 0.5$

number of base regions in the image can be significantly reduced. This compresses to over 50 times fewer regions to consider for even a moderately small image.

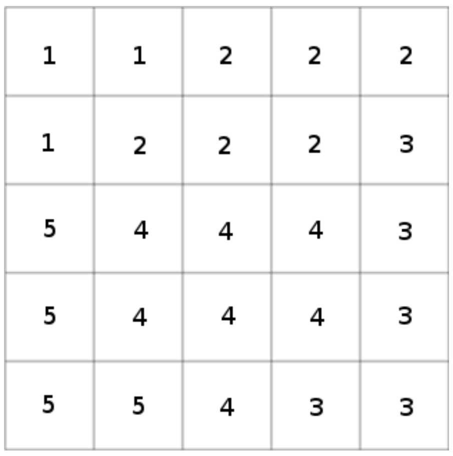
This simplified structure can be considered as analogous to image compression, where structure within an image is exploited to reduce the file size.

3.4.2 Method

Consider the 5x5 image shown in Figure 3.11. With no compression, this image requires 24 bits per pixel (bpp): eight for each of the three colour channels. This requires 600 bits in total. CDS could represent this as five superpixels, one assigned to each distinct colour. By referencing each pixel with their superpixel label and having an array of 24 bit colours, one for each superpixel, the image information can be easily reduced. The number of bits, b, required is shown in Equation 3.2, where S is the number of superpixels and P is the number of pixels. Taking the two extremes $S = 1$ and $S = 25$, the number of bits required is then $b = 49$ and $b = 725$ respectively, which is 1.96 bpp and 29 bpp. For the image shown (with five superpixels), this is 195 bits or 7bpp, a saving of 405 bits. As there is expansion of the file size as the number of superpixels increases, there exists a trade-off between

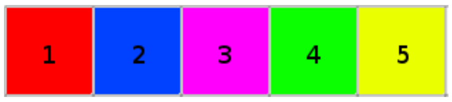


(a) Original image



1	1	2	2	2
1	2	2	2	3
5	4	4	4	3
5	4	4	4	3
5	5	4	3	3

(b) Superpixel labels



🟥 1	🟦 2	🟪 3	🟩 4	🟨 5

(c) Colour indices

FIGURE 3.11: Simple image used to illustrate the compression mechanism

the quality of the image reconstruction and the file size, to the point where, as in the example above, it could actually increase the amount of data required to render the image.

$$b = \begin{cases} \lceil \log_2 S \rceil P + 24S, S > 1 \\ P + 24S, S = 1 \end{cases} \tag{3.2}$$

3.4.3 Results

3.4.3.1 Face Database

This new compression algorithm is applied to the XM2VTS face database since these images are uncompressed and were acquired with the same sensor. The algorithm is tested on 400 images of size 720x576 pixels. The compression ratio is compared with the quality of the reconstruction of each image.

The average compression is at 54.9% of the image while retaining an average of 97.6% of the original information. Figure 3.12 shows the distribution of the results across 400 images. Despite the distribution of superpixels per image being between

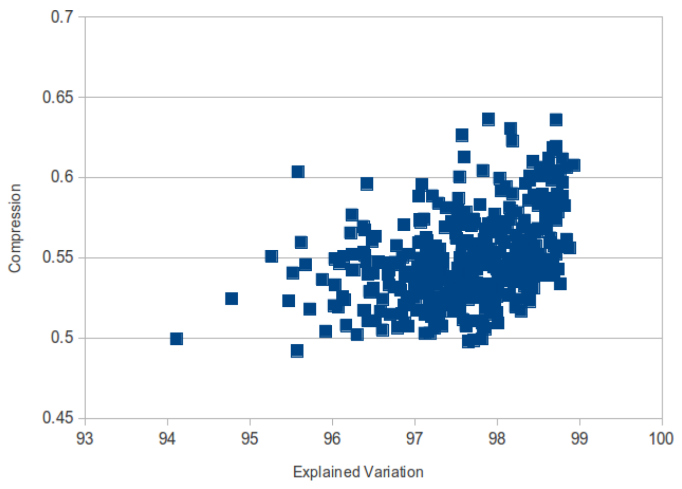


FIGURE 3.12: Showing the variation in compression with superpixel quality

400 and 1200, there is very little fluctuation in compression between images as most are between 50% and 55%; the compression has a standard deviation of 0.03.

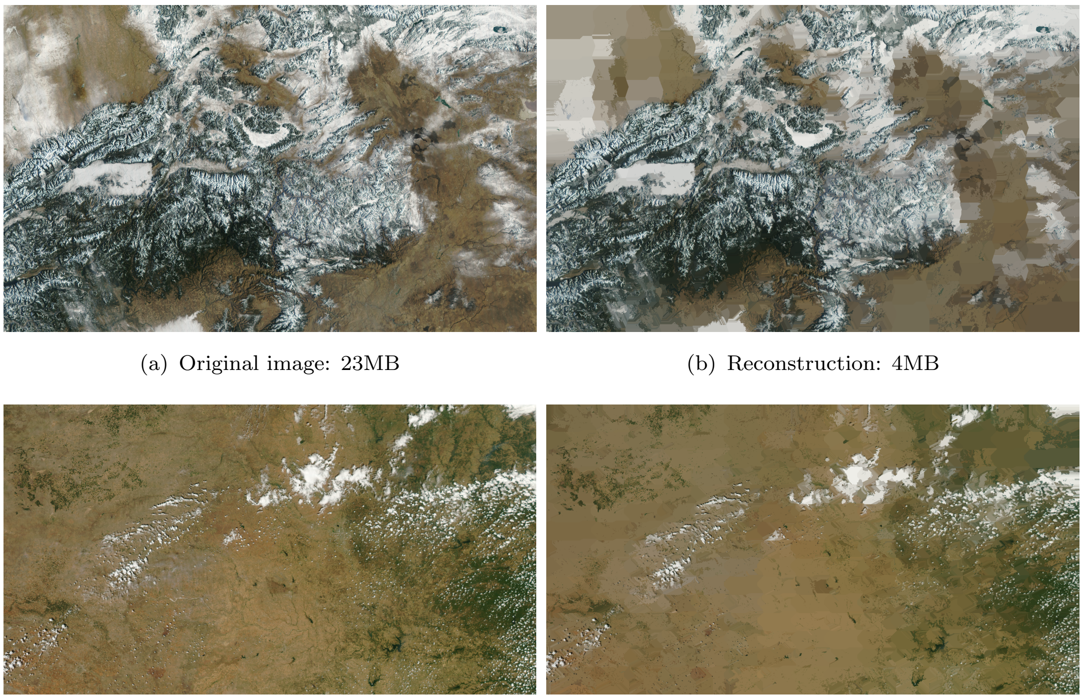
3.4.3.2 Remote Sensing

Remote sensing is the acquisition of information without direct interaction with the subject matter, the most notable example being aerial photography. These aerial photographs usually cover large areas of land and are acquired at resolutions which incur a high processing cost. By reducing the base unit of the image, these images can be used in more complex image processing algorithms without incurring this higher processing cost.

The two images shown in Figure 3.13 have an Explained Variation of 93.4% and 84.1% respectively, and both have a compression ratio of 0.84. Again this shows that more complex images (Figure 3.13(a)) have better reconstructions than seemingly less complex images (Figure 3.13(c)). The compressed images are significantly smaller than the original images; with file sizes reduced to 17% and 11% of their original size.

3.4.3.3 Image Scaling

The approach was also employed on images of the same scene derived from three different sensors: A webcam, a phone camera, and a digital SLR. All images shown in Figure 3.14 are scaled to the same resolution, 1000x750 pixels, resulting in different image qualities.



(a) Original image: 23MB (b) Reconstruction: 4MB

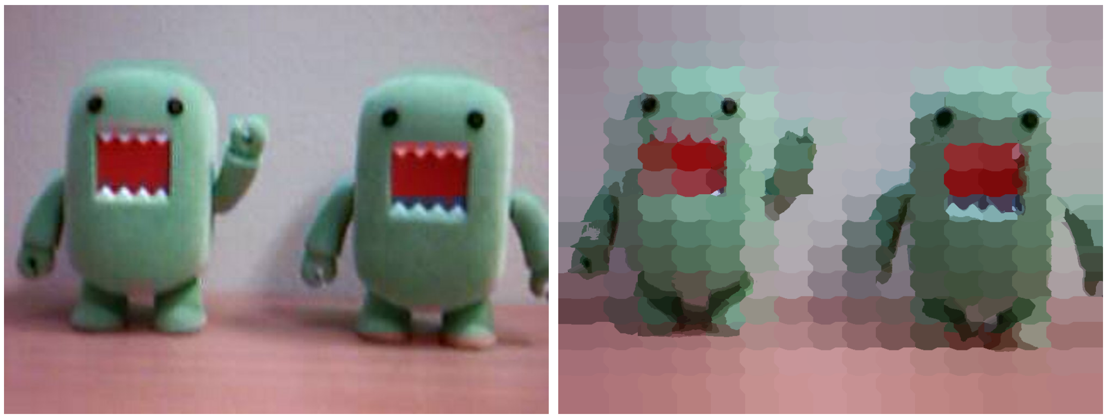
(c) Original image: 17MB (d) Reconstruction: 2MB

Figure 3.13: Results on aerial photography

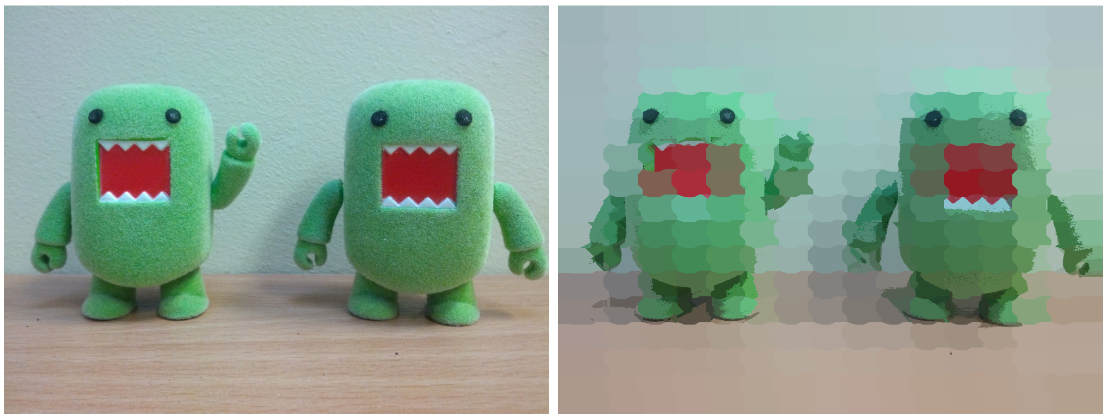
The results show that there are two factors affecting the fidelity of the compressed image. Naturally, the resolution of the original image is a significant factor. Also, the three sensors have very different colour schemes.

Each image has retained much of the structure present in the original images, such as the shapes in the eyes and even in the shadows. As such the compression appears largely unaffected by resolution, certainly between the mid and high resolution cameras.

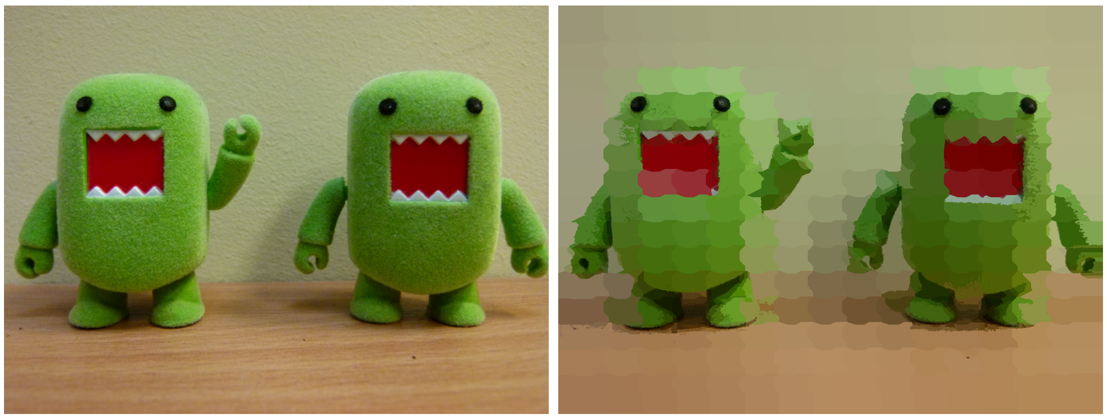
Using Table 3.1 it is difficult to draw any direct link between the image resolution and the quality of the reconstruction. The larger the superpixel size, the more likely it is that it mis-represents the image. Even when considering the ratio of pixels to superpixels, there is no link. It is possible that the cause of this is due to the colour schemes in the camera. As CDS divides on colour difference it could be that the difference in colour quality accounts for the inconsistency in quality.



(a) Webcam



(c) Phone



(e) SLR

FIGURE 3.14: Showing the same image taken with different cameras and the superpixel reconstructions.

		Superpixels		Pixels/Superpixel	
Camera	Resolution	Original	Scaled	Original	Scaled
Webcam	320x240	815	1239	94	605
Phone	2592x1944	19536	3035	258	247
SLR	4320x3240	61868	3938	226	190

Table 3.1: Results on the same scene taken at different resolutions, shown in Figure 3.14

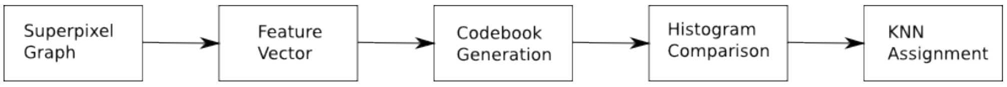


Figure 3.15: Block diagram describing algorithm

3.5 Scene Categorisation using CDS

3.5.1 Introduction

CDS aims to represent images in a more descriptive way than other superpixel algorithms, therefore it is possible that it is suitable for scene categorisation. Scene categorisation involves classifying an image based on whether or not it includes a specific object, for example a person, or is of a certain type, for instance a beach scene.

There are several stages involved in scene categorisation. The crux of it is to generate a set of features that are used to train a classifier. Images can then be categorised depending on which features are present in each image. In this instance, a Bag of Features model is used, which is described in Section 3.5.3. As superpixels have not yet been used in any machine learning applications, the features in this section have been designed for CDS.

3.5.2 Method

Figure 3.15 describes the steps necessary to perform scene categorisation. The first step is to transform the image into a superpixel graph as described in Section 3.2. This allows the structure of the superpixels to be used when generating features. The features are then generated using a bag of visual words. These features can be used to generate a codebook, which is calculated by clustering similar features from different images into a pre-defined number of codewords. The superpixels are then re-labelled as their corresponding codewords. By doing this it is possible to compare the codeword content of each image in the database, which allows the images to be classified in terms of scene content. A histogram is generated that describes the quantity of each codeword that exist in the image. These histograms are then compared with each other.

Using a leave-one-out cross validation test with K-Nearest Neighbours, each image is tested against all other images in the database to determine the closest K matches. The scene type associated with the median class of the K nearest matches is the resulting scene type for that image.

3.5.3 Bag of Features

The bag of features model (BOF) Fei-Fei and Perona [2005] is a modification of the bag of words model. The bag of words model is a way of recognising similar sentences that do not necessarily have words in the same order. By grouping the words into a common dictionary, the content of a set of documents can be analysed and therefore classified. This model is used as an analogy for classifying images based on their content. By generating feature vectors present within the image, these can be used as 'visual words' to compare the content of each image and therefore classify it based on the similarity to other images within the dataset.

3.5.3.1 Feature Vector

The feature vector produced here needs to take advantage of the benefits of using CDS. The benefit is the descriptive nature of the superpixels, for example the size of the superpixel and the superpixel density. These features are unique to CDS. The problem with bag of features is that it removes the local structure present in the image. Therefore a description of the local neighbourhood is included as part of the feature vector.

The features that are used are as follows:

- The number of neighbours, N;
- The average Euclidean distance to the neighbours, d;
- The size of the superpixel p;
- The colour of the superpixel μ;

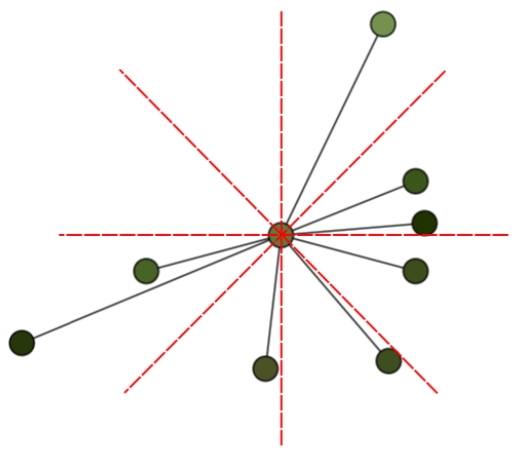


Figure 3.16: Illustrating the radial bins used in the feature vector.

- the average colour of the neighbours C_{μ}.

It is a requirement of most classifiers that the feature vectors are the same length, which is why the average distance and colour is used rather than the actual distance to each neighbour. In addition to the vector described above, two modifications are included to test improved performance.

By averaging all superpixel neighbours, most of the information is lost. For example two superpixel neighbours: a large red superpixel and a small blue superpixel, will give the same result as a small red and large blue combination. To mitigate this effect, some extra information can be introduced. Firstly, the variance of the information is calculated. Secondly, the colour information is converted to HSV so that the variation in brightness can be removed.

[Mikolajczyk and Schmid, 2005] evaluated a selection of feature descriptors. They find that their own descriptor GLOH (Gradient Localisation and Orientation Histogram), SIFT [Lowe, 2004] and shape context [Belongie et al., 2002] perform best as descriptors. These are all SIFT derivative descriptors and all include directional information. To include directional information in this feature vector, the neighbours are grouped into eight radial bins, shown in Figure 3.16. The information in each bin is still averaged but aims to improve the discriminative ability of the vector while still providing a constant vector length.

The final vector is therefore:

- The number of neighbours, N;
- The average Euclidean distance to the neighbours, d;

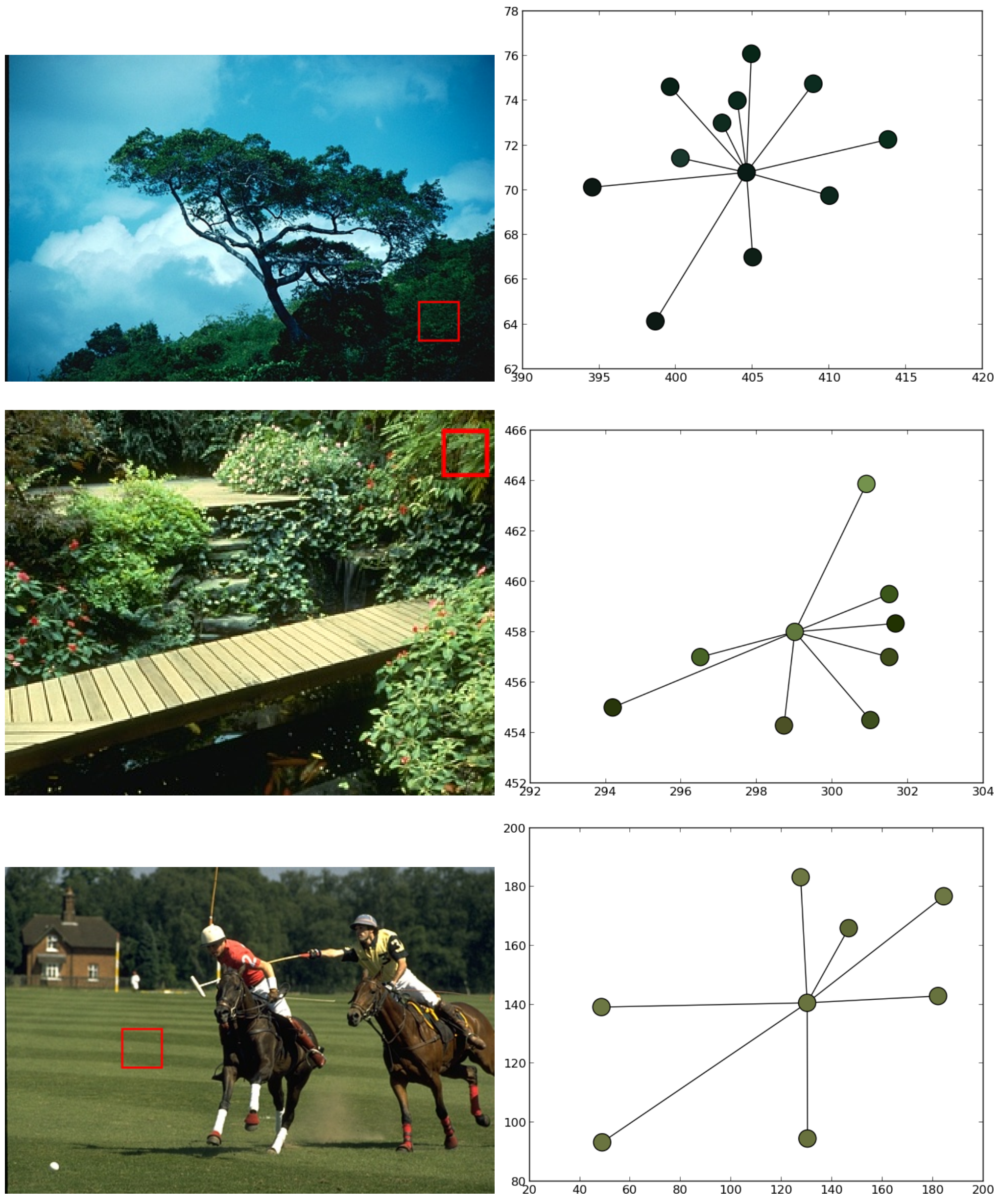


FIGURE 3.17: Showing the difference between superpixel neighbourhoods

- The size of the superpixel p;
- The colour of the superpixel C;
- the mean colour of the neighbours in each radial bin $C_{\mu\phi}$;
- the colour variance of the neighbours in each radial bin $C_{\sigma\phi}$.

Figure 3.17 shows three different graph structures from grass or plant-like regions. The central node is the superpixel for which the vector will be generated. Each edge and subsequent node represent a neighbour to that superpixel. The colour of each node represents the mean colour of each superpixel. In each case, the central node is taken from the marked red region in the image.

The first two are similar in structure, having similar distributions of superpixels, both in distance from central node and number. The third is different as the distances are much greater (as the superpixels are likely to be larger in this region). Despite this the number of nodes is similar and the colour is similar. The intention would be that two different codewords are generated here, one for the grass regions and one for the foliage regions of the first two images.

[Yang et al., 2007] evaluates BOF when using it for scene classification. The paper considers various weighting schemes, the effect of vocabulary size and feature selection methods. As a weighting scheme, the best method described is the 'term frequency - inverse document frequency' (tf-idf) [Sivic and Zisserman, 2003]. In their application it is used on documents however it could be used on visual words. Given in Equation 3.3, it is a product of how often a term appears in the document and the rarity of that word within the dataset. n_{id} is the frequency of word i within a document d and n_d is the total number of words in that document. N is the total number of documents and n_i is the total number of occurrences of word i across all documents.

[Yang et al., 2007] also suggests that normalisation should not be applied as this removes the information obtained from images with different complexities but similar colour distributions.

$$t_i = \frac{n_{id}}{n_d} \log \frac{N}{n_i} \tag{3.3}$$

3.5.3.2 Codebook Generation

The codebook contains a pre-determined number of codewords. Each codeword represents the centroid of a set of superpixel features. The codewords are generated using K-means clustering. K-means assigns each feature to a centroid. These centroids are iteratively updated to provide the best representation of the data. This is achieved by minimising the Euclidean distance to a set of superpixels.

These codewords are then used to describe an image in terms of the number of each codeword in the image. This turns each image into a single vector of fixed length.

3.5.4 Classification

The classification of the images is calculated using a similarity measure between the vectors, then classified using K-Nearest Neighbours. The similarity metric given in Equation 3.4 is the Euclidean distance between length N vectors i and j.

$$d^2 = \sum_{i=0}^{N} (f_i - f_j)^2 \tag{3.4}$$

K-nearest neighbours classifies a data point given the class of the closest k neighbours to the test point. Including more neighbours reduces noise but increases smoothing in the feature space.

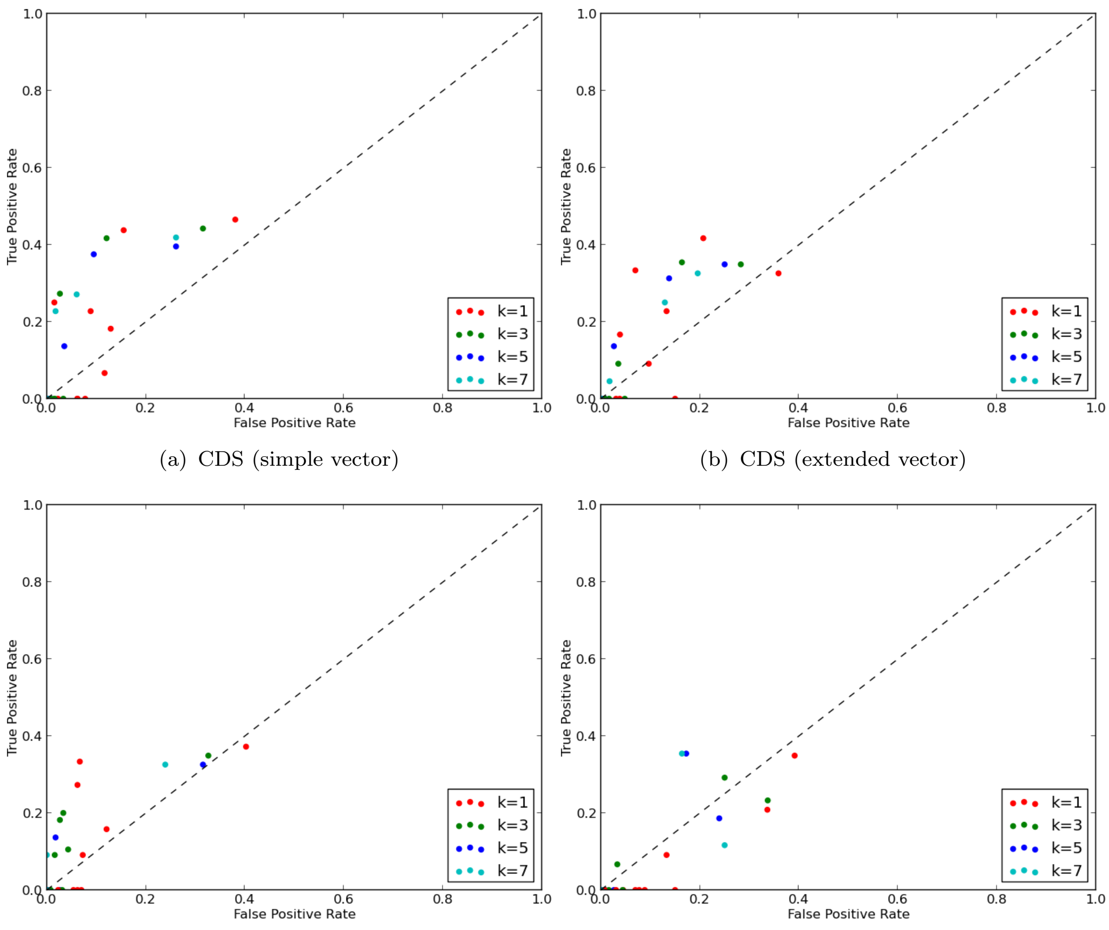
3.5.5 Results

Classification is determined by using a leave-one-out cross-validation test. The two new feature vectors presented in Subsection 3.5.3.1 are compared. The number of visual words that are the k-means centroids is also varied. The algorithm is compared with N-cuts, in order to show the benefits of content-aware superpixels. In addition it is compared with SIFT.

The results are computed over several different class labels. The PASCAL challenge database is used as it contains a label assigned to each image describing the content within it. A random set of approximately 150 images from the database was taken. The images were then subject to a binary classification to determine whether they matched a chosen label. For example, each image is tested to determine if it contained an aeroplane. A confusion matrix was computed showing whether the image was correctly classified based on the ground-truth labelling.

To visualise the results a Receiver-Operator characteristic (ROC) is used and shows results obtained from all class labels. This is a plot of the false positive rate (FPR) against true positive rate (TPR). The more accurate the result, the closer it will be to the top-left portion of the graph: low false and high true positives. The line drawn through $f(x) = x$ is an indicator to show the quality of the result; those points above the line show more true positives than false.

All of the graphs show some results that do not work at all, suggesting that there is not enough training data to successfully detect that type of scene.



(a) CDS (simple vector) (b) CDS (extended vector)

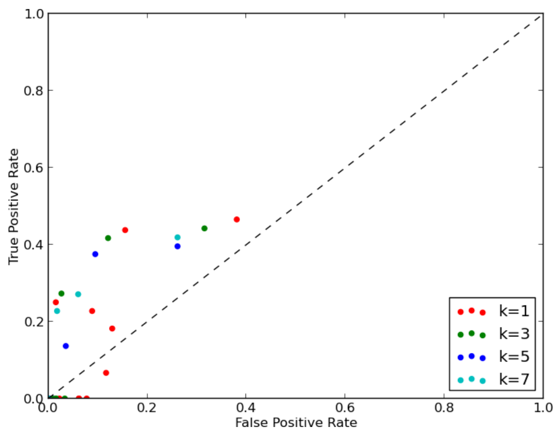
(c) N-Cuts (d) SIFT

FIGURE 3.18: ROC showing results on multiple images with 256 Words

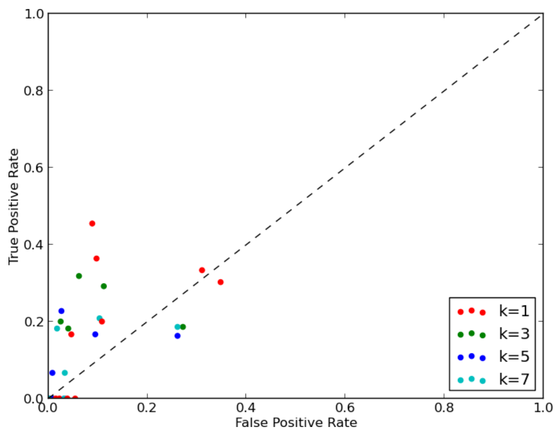
When using 256 words, as shown in Figure 3.18, SIFT has a large number of results below the line $f(x) = x$, indicating more false positives than true positives. The superpixel algorithms, N-Cuts and CDS, both show large numbers of results of good accuracy. None however are particularly convincing that the feature vector used is of any use in accurately determining scene information.

Increasing the number of visual words, shown in Figure 3.19 improves the accuracy of the results, yet none surpass a 50% TPR. The results with the highest TPR/FPR ratio exist in Figure 3.19(b), as there is a large cluster of results with a low FPR.

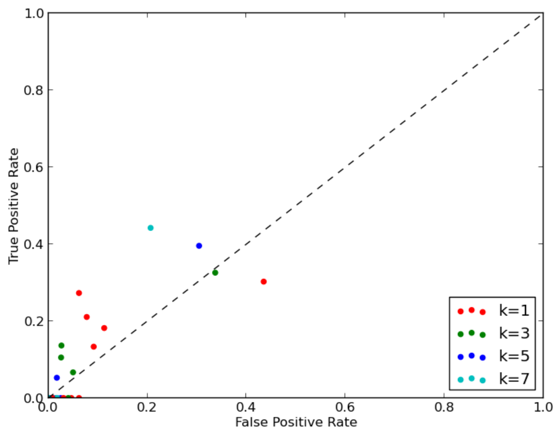
What is interesting is that as K increases, the TPR actually reduces. This could be interpreted as there being very little discriminative ability in the feature vector.



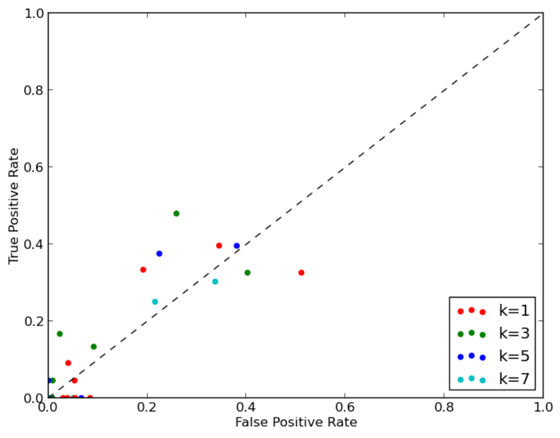
(a) CDS (simple vector)



(b) CDS (extended vector)



(c) N-Cuts



(d) SIFT

FIGURE 3.19: ROC showing results on multiple images with 1024 Words

3.6 Conclusion

Each of the methods developed in this chapter could be taken further in their own right yet here they are developed enough to show the potential of CDS in these fields.

This chapter has shown a general segmentation algorithm, using pairwise superpixel similarity, that is easy to apply. A simple equation is given to determine superpixel similarity based on distance and colour cues. The algorithm is non-iterative and is computable very quickly once the superpixel labels are known. This illustrates the ease of using the superpixel labels for quick application of further processing.

The combination of a spatial and colour description provided by CDS is beneficial and improves the result. This method of segmentation cannot be performed with pixels as it is prohibitively expensive for even moderately small images.

The segmentation algorithm currently does not include connectivity constraints. The algorithm would benefit from these constraints as it would stop segmentation labels from 'bleeding' out into the rest of the image.

Each of the experiments show that CDS is consistently better than N-Cuts, the performance advantage being derived by the content-driven nature of the approach.

In addition to segmentation, using the superpixels from pre-processing has the added benefit of compressing the data into a smaller format.

As a consequence of generating superpixels, the tests show that 50% compression is achieved. Theoretically it may not be advantageous to use CDS as a dedicated compression method as it does not always result in a smaller format. There are also subtle effects on performance where different colour schemes are present, as in Figure 3.14, and these effects require further study.

Clearly there has been much research on image compression and techniques are highly sophisticated and very effective. CDS is a totally new approach and could offer further advantage in compression. It is interesting to note that JPEG has poor performance on large areas of similar colour Viraktamath and Attimarad [2011], whereas CDS is designed with this property in mind.

The final application, Scene Categorisation, shows promise but requires further work to be more useful. There is however some slight advantage to using CDS as opposed to either N-Cuts or SIFT.

There are some surprises when adding additional information in that it does not clearly improve the system. Increasing the number of visual words has a positive effect, albeit a small one. Adding the additional directional information does not clearly improve the result either, which was expected given the quality of other directional feature vectors.

The potential problem is that the number of sample is insufficient to train the classifier and that with more time and more samples the results may improve.

Chapter 4

Localising Image Focus

4.1 Introduction

The ability to focus is implicit in image formation. In photography, there are passive and active approaches to achieve image autofocus wherein the image clarity depends on optical parameters. Passive autofocus approaches analyse local image contrast as part of a feedback mechanism driving the lens motor whereas active approaches aim to sense distance to derive focus capability. As such, we can enjoy clear photographs which can usually be acquired with the object of interest in sharp focus.

In contrast to the plethora of approaches for image autofocus, there are few approaches which can be applied to analyse an image to automatically determine the regions which are in sharp focus. One such approach, the Sum Modified Laplacian [Nayar and Nakagawa, 1994], was designed to analyse shape from focus using the relative differences in contrast at differing image resolutions. More recent works are concerned with the extraction of edge information [Tai and Brown, 2009] as the edges contain more high frequency information. Another method uses Gabor wavelets [Chen and Bovik, 2009] that are tuned to detect high frequency image components. Other methods [Liu et al., 2008, Kovacs and Sziranyi, 2007, Levin, 2007] attempt to model a blur kernel and use convolution to inverse the blurring process.

The methods chosen to compare with are the Sum Modified Laplacian as it can be consistently replicated. Also used are the approaches described in [Levin, 2007] and [Liu et al., 2008].

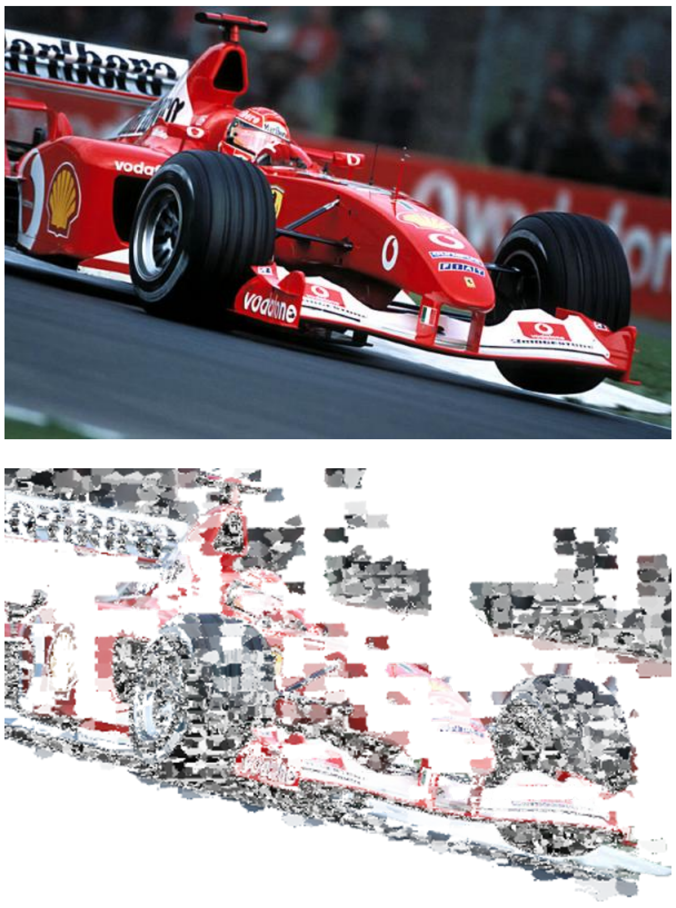


FIGURE 4.1: Determining focused areas of an image using superpixels

However, these methods require the tuning of several parameters and rely on feedback from human vision to determine if the result is 'correct'. As these rely on sharpness of edge information, they are sensitive to noise.

A new approach is introduced that can be applied to explicitly extract focal regions of a single image without requiring parameterisation. By exploiting the properties of the scale-space and CDS, it is possible to extract non-uniform regions in scale. The distribution of superpixels through scale can be used to infer where the image has been affected by smoothing and therefore where the image is in focus as illustrated in Figure 4.1.

With conventional images it is difficult to ascertain, other than with human vision, whether image focus has been correctly localised. To advance the analysis of focus performance, we use Light Field Photography (LFP) to validate our approach, as it provides a controlled environment with which to vary the focal plane in an image. The results can show precisely that as image focus varies, the extracted focus regions of the image vary consistently with that change.

4.2 Scale-Space

Developed by Witkin [Witkin, 1983], scale-space is a one-parameter family of derived images that successively smooths an image, removing more high-frequency features with each scale. Among other things, it has been used in detecting scale-invariant edges [Bergholm, 1987], as a basis for the popular SIFT and SURF operators, and also saliency [Kadir and Brady, 2001]. Edges are deemed to be more significant if they persist for several scales whereas saliency is more significant if it persists over few scales. To generate the scale-space, the new images need to be derived by convolving the image with a Gaussian filter, given in Equation 4.1, where t denotes the scale.

$$g(x, y, t) = \frac{1}{\sqrt{2\pi t}} e^{(-\frac{x^2+y^2}{2t})} \tag{4.1}$$

The choice of t is based on logarithmic sampling. To efficiently construct the scale-space,t is chosen such that the difference between scales is maximised without losing detail. Equation 4.2 provides a method of selecting t [Lindeberg, 1994]. τ is the transformation of the image as a function of the smoothing parameter t and A is a free parameter. This motivates the choice of sampling to be $t = 1, 4, 16, 64, 256$, which causes the logarithmic sampling to produce a linear increase in τ and therefore a linear difference between scales.

$$\tau(t) = A \log t \tag{4.2}$$

The scale-space is then collapsed into a single volume, where successive two-dimensional slices represent increasing levels of detail.

We can infer from the scale-space that if a spatial region is consistent over all scales then smoothing has had little effect. Therefore this region contains little high frequency information and is more likely to be out of focus.

4.3 Extending CDS into Three Dimensions

To allow CDS to be used on volumes it must be extended to a third dimension, where the third dimension is image scale. Fortunately, as CDS is a combination of spatial computer vision techniques, each sub-process can be separately transposed into 3D. The two main mechanisms: ‘Distance Transform’ and ‘Active Contours without Edges’ are ideally suited for 3D.

Superpixels that exist in the 3D space are referred to as supervoxels.

4.3.1 Distance Transform

The extension of the Distance Transform into 3D is centered around replacing the binary image described in Section 2.1.3.2 with a binary volume V, where the third dimension k is the scale of the original image.

The Distance Transform in 3D transforms a volume such that the volume displays the distance D of each voxel at location $(i, j, k)$to the nearest background location (x, y, z). This is given in Equation 4.3.

$$D = \min_{x,y,z:\, V(x,y,z)=False} \sqrt{(i-x)^2 + (j-y)^2 + (k-z)^2} \tag{4.3}$$

This growth occurs at each iteration t. The growth of the supervoxel S is given in Equation 4.4.

$$S^{<t+1>} = S^{<t>} \cup \{(x, y, z) \colon D(x, y, z) = 1\} \tag{4.4}$$

4.3.2 Active Contours Without Edges

Transforming ACWE into the third dimension is actually quite simple. The transformation is simply to consider each voxel within a supervoxel and calculating the force F that updates that voxel within a 3D distance function.

To use this algorithm with supervoxels it is necessary to define $\Omega(x, y, z)$ as a vector that contains a set of all the voxels within the supervoxel, as shown in Equation 4.5.

$$F = \int_\Omega \frac{1}{N} \sum_{i=1}^{N} |I_i(x, y, z) - u_i|^2 dxdydz - \int_\Omega \frac{1}{N} \sum_{i=1}^{N} |I_i(x, y, z) - v_i|^2 dxdydz \tag{4.5}$$

The segmentation criterion of either region u, v is given as the average of the means (u_i, v_i) of each of the N colour channels V_i of the volume V; shown in Equation 4.6. Supervoxel division occurs if there is a significant difference between any of the colour channels.

$$\begin{aligned} u_i &= \frac{\int_\Omega V_i(x,y,z)dxdydz}{\int_\Omega \Omega(x,y,z)dxdydz}, \forall \Omega_D(x,y,z) > 0 \\ v_i &= \frac{\int_\Omega V_i(x,y,z)dxdydz}{\int_\Omega \Omega(x,y,z)dxdydz}, \forall \Omega_D(x,y,z) \leq 0 \end{aligned} \tag{4.6}$$

ACWE still requires the separation of a supervoxel into two regions u, v, but these regions now occupy a 3D signed distance function $\Omega_D(x, y, z)$. The two regions C_u, C_v are given in Equation 4.7.

$$\begin{aligned} C_u &= \{(x,y,z) : \Omega'_D(x,y,z) > 0\} \\ C_v &= \{(x,y,z) : \Omega'_D(x,y,z) \leq 0\} \end{aligned} \tag{4.7}$$

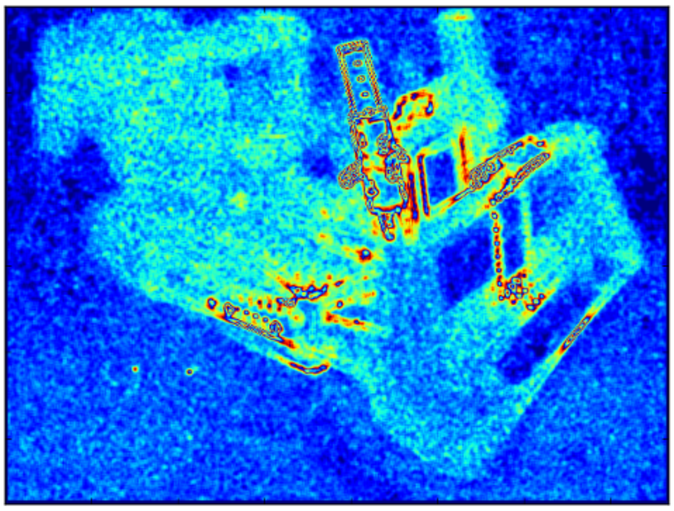
4.4 Sum Modified Laplacian

As a comparative approach, an established method of analysing focus is the Sum Modified Laplacian (SML). The focus is derived from the image I at levels spaced by a step Δs, shown in Equation 4.8.

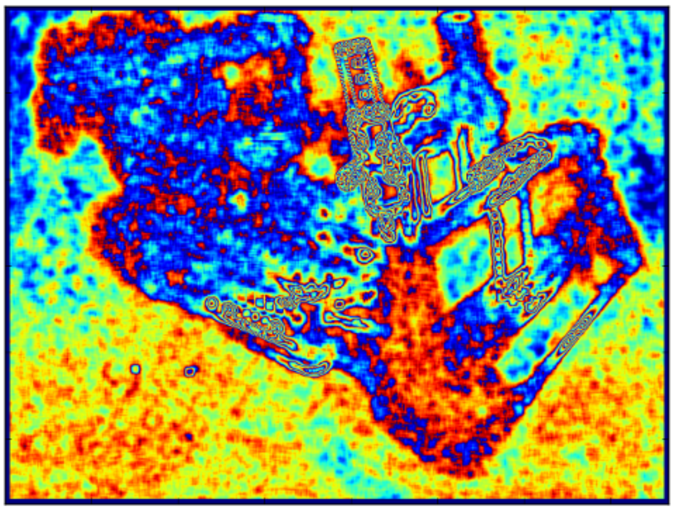
$$\begin{aligned} \mathrm{ML}(x,y) = &|2I(x,y) - I(x-\Delta s,y) - I(x+\Delta s,y)| \\ &+|2I(x,y) - I(x,y-\Delta s) - I(x,y+\Delta s)| \end{aligned} \tag{4.8}$$

The focus measure (Equation 4.9) at (i, j) is evaluated as the neighbourhood (of size N) sum of the modified Laplacian (Equation 4.8) which exceed a threshold T. The step size can be varied to locate different texture sizes.

$$F(i,j) = \sum_{x=i-N}^{i+N} \sum_{y=j-N}^{j+N} \mathrm{ML}(x,y) | \mathrm{ML}(x,y) \geq T \tag{4.9}$$



(a) $N = 2$



(b) $N = 4$

FIGURE 4.2: Showing the same image using different values for N in SML. The red areas depict distinctly different areas of focus in each image.

This is problematic as it will only select textures of a chosen size and will be affected by the size of the neighbourhood. This makes using the algorithm as a focus measure subject to human opinion and insight. The results of focus detection in Figure 4.2 show that the quality of the result relies on selection of appropriate parameter values, and the selection of those parameters relies on human visual analysis. This property is not a problem if one is comparing images generated using the same parameters.

In contrast, CDS can inherently locate scale-varying regions without parameterisation or supervision. It is also is region based, thereby selecting regions of interest which are more useful than individual pixels.

4.5 Applying CDS to Focus Detection

Firstly, images are converted into a set of 3D scale-space representations using the values of t defined in Section 4.2. By using scale-space for superpixels, the aim is to produce superpixels that have context over scale; scale-persistent superpixels are more likely to be stable whereas scale-varying superpixels are more likely to be in feature-rich areas of the image.

By grouping regions of scale-space, it is possible to gain information about the nature of that region of the image. The idea is that superpixels that exist in the low detail area of the scale-space are less likely to contribute the high-frequency content present in the focused region of the original volume. A set of supervoxels is initialised in the least-detailed layer of the scale-space. This is done such that as the supervoxels grow through more complex layers of the space, they increase in number. Initialising in the most complex layer would require more supervoxels than necessary to represent the least complex layer.

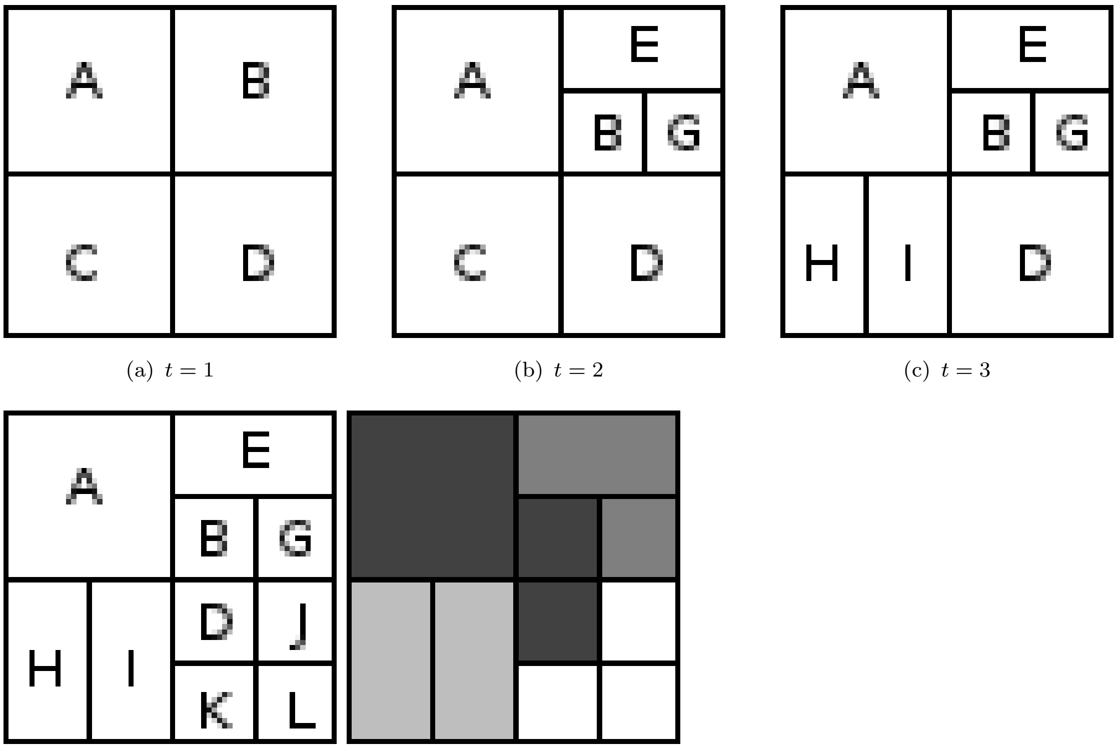
The focus is determined as the lowest detail layer T_{min} in which the supervoxel still exists. Therefore the supervoxels that exist in the first layer have the lowest focus value. Supervoxels that exist solely in the highest layer have the maximum focus value.

This is shown in Equation 4.10. The focus measure F for a supervoxel s is controlled by the first layer in which that supervoxel exists. $T_{\max}$ is the number of layers in the image.

$$F(s) = \frac{T_{\min}}{T_{\max}} \tag{4.10}$$

A hypothetical example is given in Figure 4.3, which shows four layers of the same volume, where each labelled region represents a supervoxel. Multiple layers can contain the same supervoxel, for example region A which exists for all layers, but the minimum layer is $t = 1$. Each subfigure is given with the layer t in the volume it represents. Figure 4.3(d) shows the least smoothed layer, i.e. the original image. It is the supervoxels in the original image that are assigned focus values.

As region A remains constant, no change in space or scale has been detected and can be considered out of focus. Regions *A,B,C,D* are therefore given a focus value of $F(s) = 0.25$. Next *E,G* have a focus of 0.5 as they first exist in layer $t = 2$, and regions *H,I* have a focus value of 0.75. Regions *J,K,L* are therefore the most



(a) $t = 1$ (b) $t = 2$ (c) $t = 3$

(d) $t = 4$ (e) *Result*

FIGURE 4.3: Illustrating how focus is determined

likely to be in focus, with $F(s) = 1$. Figure 4.3(e) shows this graphically, where brightness indicates a higher focus value. Each location in space shows the highest focus value at that point. For example in the case of regions *C,H,I*, even though they occupy the same spatial location, the focus values of H and I are given as those are the supervoxels that are actually present in the original image, and therefore the ones that are assigned a focus value.

4.6 Light Field Photography

4.6.1 Introduction

Light Field Photography (LFP) [Adelson and Wang, 1992, Levoy and Hanrahan, 1996, Wilburn et al., 2005, Ng et al., 2005] is continuing to gather interest. It is the process of re-focusing an image or shifting the perspective of the image.

LFP achieves this by gaining more information about the scene and using that information to infer hypothetical images created using ray-tracing. This additional information comes from an array of micro-lenses located between the main lens

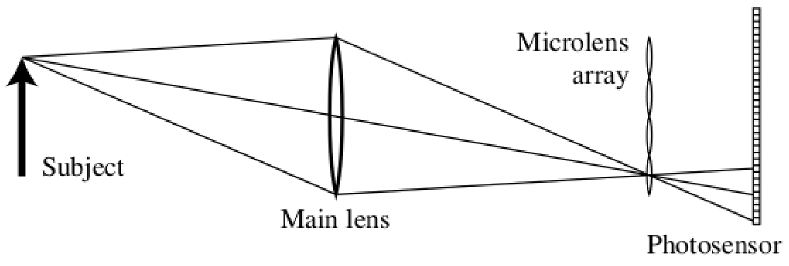


FIGURE 4.4: Illustration of the plenoptic camera, taken from [Ng et al., 2005]

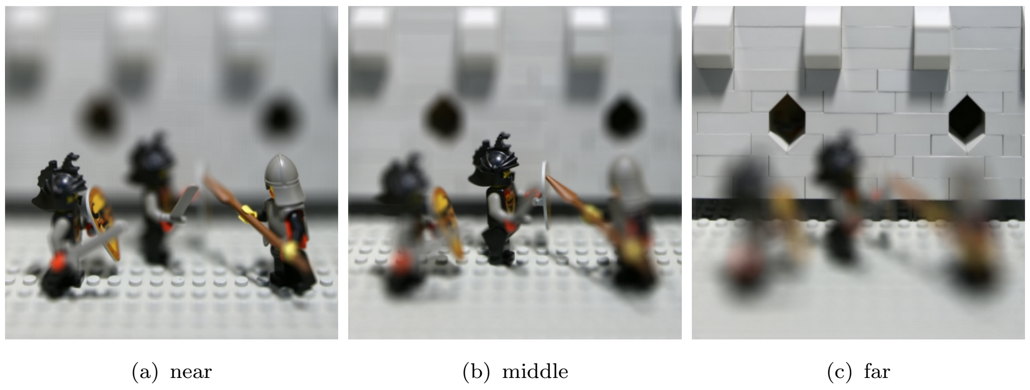
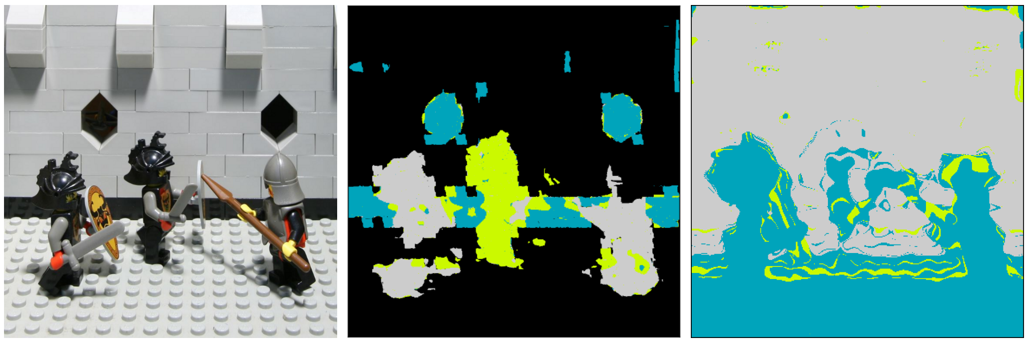


FIGURE 4.5: Illustrating the effect of change of focus on a light-field

and the sensors, shown in Figure 4.4. The ray of light is measured as it passes through each micro-lens creating a four-dimensional light field $L(s,t,u,v)$: two in the regular sensors and two created by the micro-lenses. This allows each ray travelling through the camera to be modelled and a source distance and position to be calculated. The lightfield can then create new images by artificially moving the distance of the image from the camera; changing the focal distance. A capture method using a single exposure has been developed commercially.

4.6.2 Generating the Test Images

This ability is used to generate controlled test images. A series of images are chosen from a light-field that focus on a different section of the scene, thereby allowing the efficacy of focus detection to be measured. An example series of images is given in Figure 4.5 which shows ability of LFP to focus on foreground and background objects.



(a) Reference image

(b) The labelled CDS focus response of each image

(c) The labelled SML response of each image

FIGURE 4.6: Result on the lego image

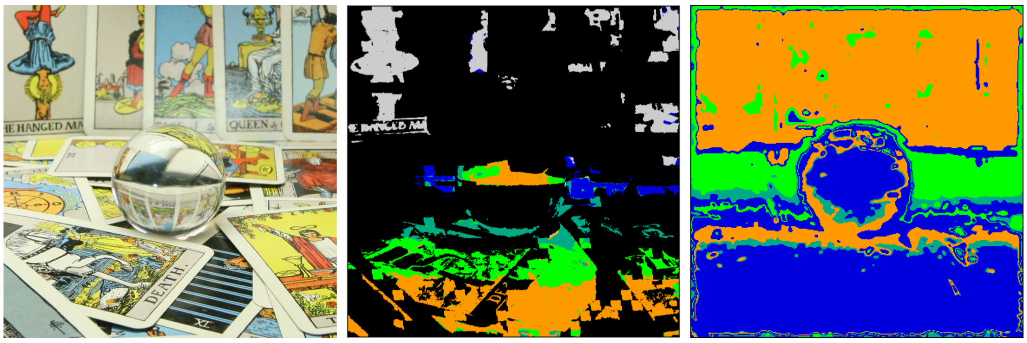
4.7 Results

Evaluation of the focus algorithm is achieved using controlled images derived from light fields which is then compared to SML, where $\Delta s = 1, T = 1, N = 3$ in order to give comparable results to CDS. There are two ways of analysing the quality of the results. Firstly, the 'focus response', where the supervoxels are drawn on the image as an alpha layer to show which parts are in focus. This response is then used in conjunction with the depth information of the light field to determine which depths of the image are extracted as 'in focus'. This can then be used to label the image with the corresponding depths. The images in this section are taken from light-field images available through the Stanford Computer Graphics Laboratory [1].

Figure 4.6 shows the result on the lego image in Figure 4.5. Figure 4.6(a) shows the image totally in focus for reference. Figure 4.6(b) shows the focus of each image as the focal depth changes. The coloured labels correspond to the focus of different images, and so here the change in focus through the image can be observed by areas of the image being occupied by distinct bands of colour. Grey corresponds to image 0, green to image 1 and blue to image 2. Black regions were not labelled as in focus in any image. CDS clearly shows the change in focus in the image, whereas SML incorrectly misses the central figure and mis-labels the background.

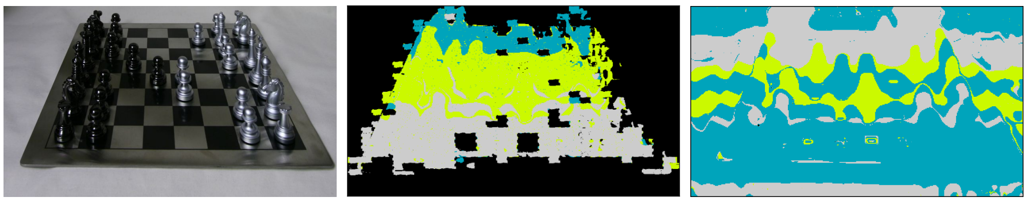
Figure 4.7 shows the response to a light-field image containing tarot cards. The CDS response is shown in Figure 4.7(b) where the objects at each depth belong to different image labels when the object was in focus. Orange corresponds to image

[1]http://lightfield.stanford.edu/



(a) Reference image (b) The labelled CDS focus response of each image (c) The labelled SML response of each image

FIGURE 4.7: Result on the image containing tarot cards



(a) Reference image (b) The labelled CDS focus response of each image (c) The labelled SML response of each image

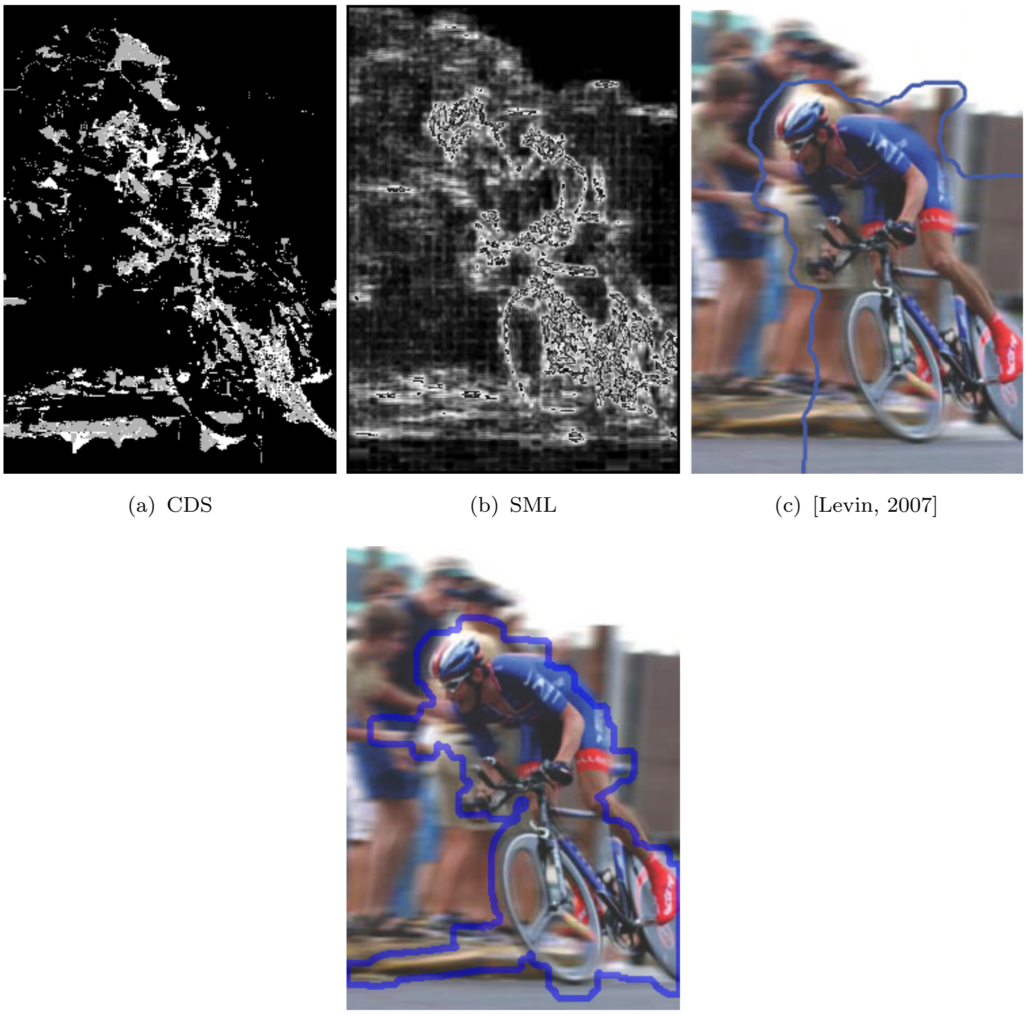
FIGURE 4.8: Result on the chess image

0; green to image 1; cyan to image 2; blue to image 3 and grey to image 4. Here, the SML response shows no clear distinction between the images, and the bands present in the CDS response are missing. There is also significantly more noise, as most of the pixels are labelled as being in focus, whereas CDS shows clear regions of no focus at all.

The image in Figure 4.8 again shows a clear transition from foreground to background as the focus of the image changes. In the CDS image there is once again a clear separation of each response, SML cannot correctly distinguish the focus as the focal depth changes.

Figure 4.9 compares the result from CDS with three other techniques. The brightness of the result denotes how focused that area is. CDS therefore detects the cyclist well. It is difficult to compare with other approaches as only the classification result is available and their techniques are not evaluated using LFP. However, we detect largely the same regions in all. There is some inclusion of the background, however this is at a lower focus value to the cyclist.

The CDS algorithm has also been applied to several sports images shown in Figure



(a) CDS (b) SML (c) [Levin, 2007]

(d) [Liu et al., 2008]

FIGURE 4.9: Comparing CDS with other techniques

4.10. The motorsport images are particularly suited to illustrating the ability to extract focused regions since the car has been extracted significantly more than the background. Note that neither technique extracts regions of uniform colour. This is because, inherently, contrast does not change in uniform regions.

There are notable differences between CDS and SML. Firstly, SML does a much better job of extracting the basketball image. However in the car images, SML detects areas of erroneous focus in the background that are attributed to strong edge information. As CDS does not rely on edge information to detect focus, these regions are not detected.

Table 4.1 compares the images as labelled in Figure 4.11 for both SML and CDS by calculating the fraction of the output that is contained within the ground truth.

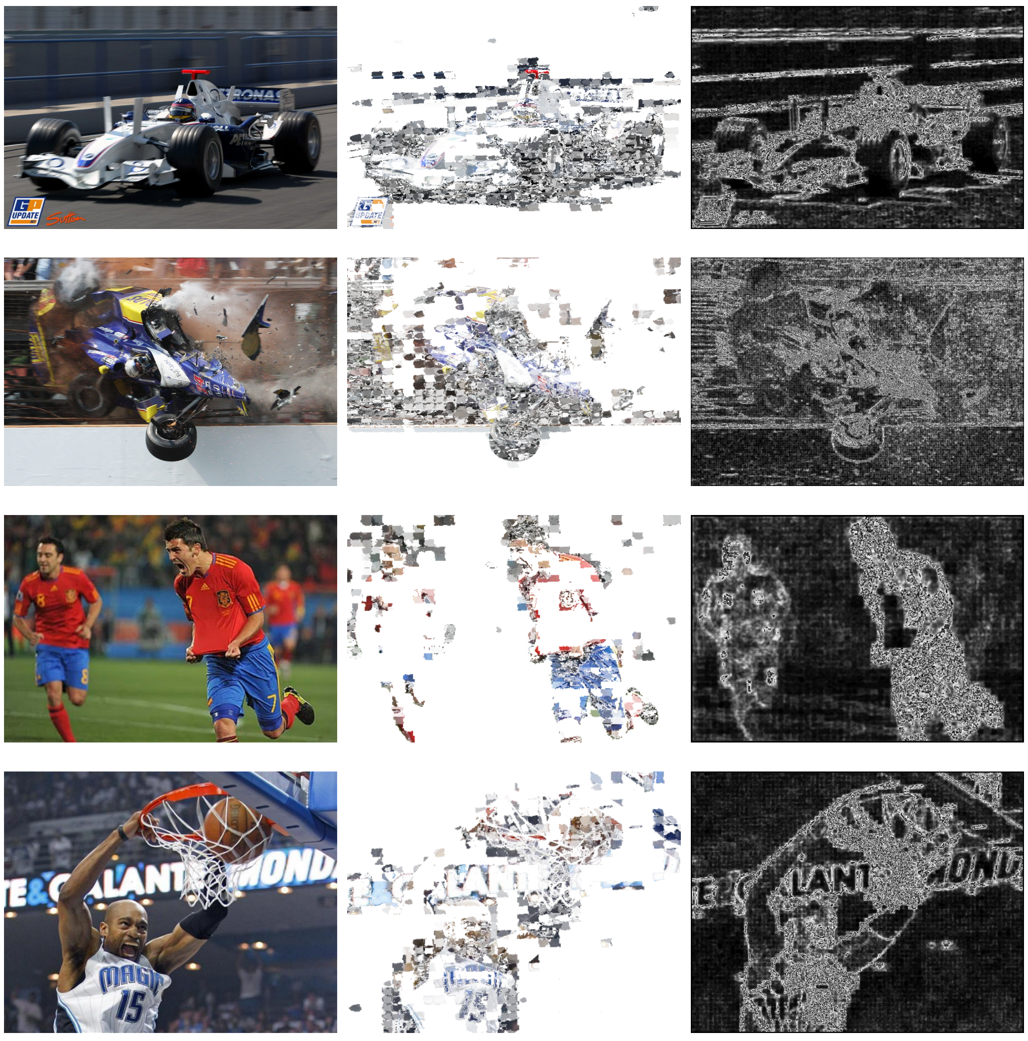
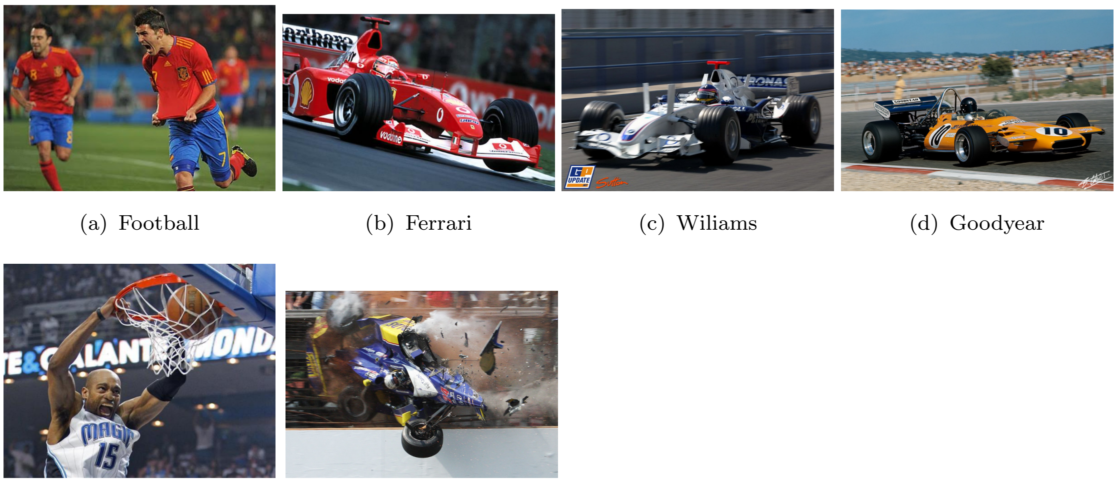

(a) Original (b) CDS (c) SML

FIGURE 4.10: Sports images

The ground truth was derived by averaging the response of five different human ‘labellers’. The ‘labellers’ were instructed to highlight the regions of the image which appeared to be in sharp focus. This implies that some uniform areas are manually labelled to be in sharp focus whereas these areas are detected by neither SML nor CDS. These results show that in most cases, CDS performs as well as SML, where SML has been manually tuned to give the best response. This shows that CDS is able to successfully determine the image focal regions automatically.

TABLE 4.1: Results on two images to show the percentage of the response that corresponds with a ground truth.

Image	CDS	SML
Football	0.49	0.43
Ferrari	0.6	0.74
Williams	0.85	0.89
Goodyear	0.96	0.97
Magic	0.59	0.89
Crash	0.79	0.74


(a) Football (b) Ferrari (c) Wiliams (d) Goodyear

(e) Magic (f) Crash

FIGURE 4.11: Test images used to compare SML and CDS

4.8 Conclusion

Any image can contain both focused and unfocused regions. There is no way to test the validity of a focus detection algorithm without an existing technique that can determine focus accurately. By using the depth information from the light field it is possible to show that the focal response corresponds to a specific image depth and that this depth changes consistently with image focus.

Essentially, CDS highlights the parts of the image that are in focus but also are more likely to contain high-frequency information. As CDS creates new superpixels on detecting image variation, there will be some constant colour areas of the image that do not change significantly with blurring. The result will be that these regions are not marked as in focus.

While other methods can extract points of focus within the image, there are several problems with these. The first is that they rely on successful selection of the algorithm parameters. As we have shown, the results on SML can vary by as much as 50% depending on the selection of adequate parameters. In addition, as these are edge based techniques, they also rely on the absence of noisy edges in the image. CDS negates this by considering regions within the image, as there is an inherent averaging within ACWE.

This chapter has developed the first application of superpixels in conjunction with scale-space. Applying CDS to the task of focus detection gives a result which has been shown to correspond accurately to the focal regions of the image. Crucially, it is unsupervised and as such gives an unbiased representation of the focus within an image, which can be demonstrated by using the unique properties of Light Field Photography.

Chapter 5

Further Work

During the course of the research, several ideas for other applications were considered. This chapter outlines those ideas that either could not be completed within the time frame or had insubstantial results at the time of writing.

5.1 Technique Basis

There are several further experiments to do in light of the results achieved in this thesis.

The first would be to place the division process with a new form of active contours: open active contours [Shemesh and Ben-Shahar, 2011]. These are lines that attempt to minimise the distance between two pre-prescribed boundary lines while adhering to image features. By treating the boundaries as two halves of a superpixel, it could be possible to use this algorithm as a replacement for ACWE and potentially create a single equation to control the superpixel. As illustrated in Figure 5.1, a superpixel could grow uninhibited by the image, only stopping on other superpixels, while also dividing using image content (shown in blue) using open active contours (show in red) inside the superpixel.

Another investigation would focus on enhancing the segmentation criteria. Currently it is concerned with colour but it could be adapted to deal with other features such as texture, making it useful in more applications, and perhaps more successful in scene categorisation.

In addition to this change to the algorithm, there is more work required to improve

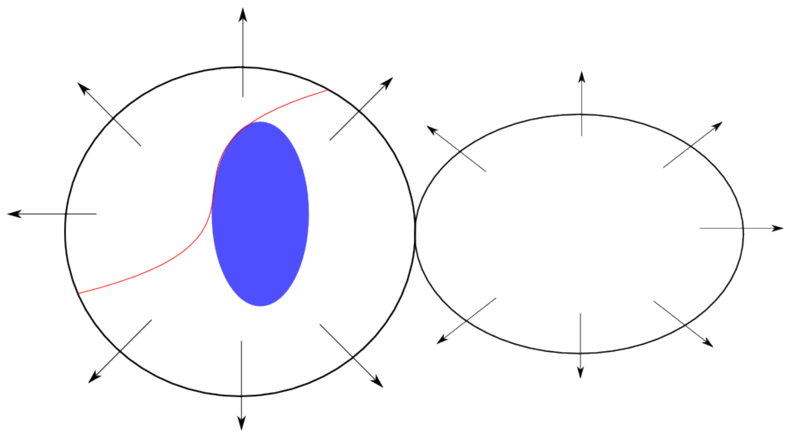


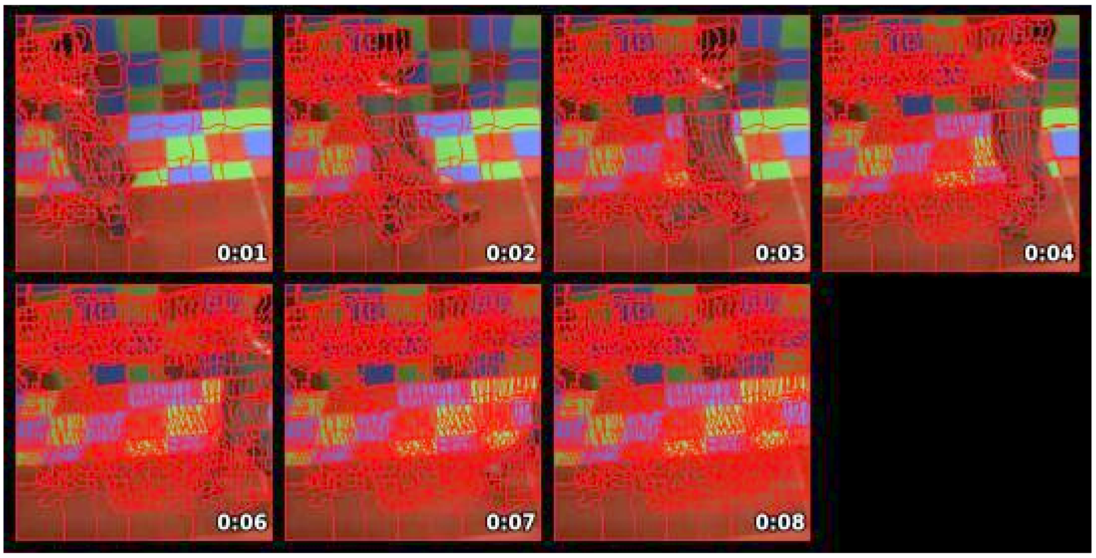
FIGURE 5.1: A stylised version of a unified control mechanism

the accuracy of the scene classification algorithm. This would be achieved either by improving the feature vector or the model used to categorise images. One problem could be the size of the database; using extra test images could be beneficial.

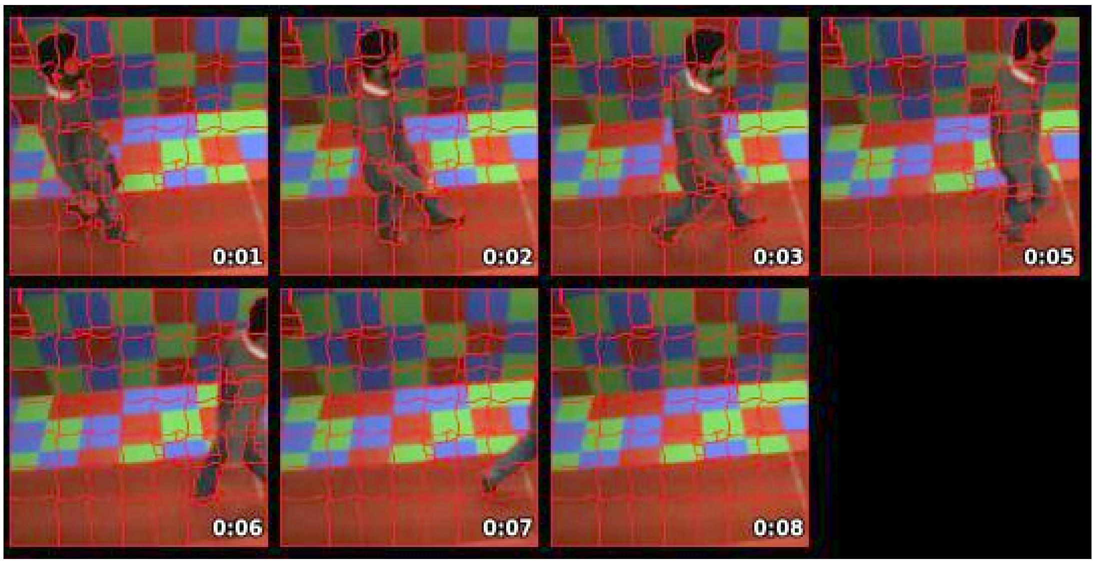
Finally, the initial compression algorithm could be extended. It shows promising results in its current form yet is not intended as a compression algorithm. There are some changes that would improve the quality of the compression. Principally, there is a colour index for each superpixel, yet there may be significant colour overlap. Reducing the way in which this information is stored could improve file sizes. In addition, in its current form the compression algorithm is a theoretical study and no investigation has been made on the best way to actually store the data. Additional encoding techniques could further improve file sizes.

5.2 Video processing and 3D Volumes

Video processing was considered, and actually served as the basis for computing the 3D scale-space representation required for focus detection. It would be inefficient to simply process each frame and two methods were explored. These methods are shown in Figure 5.2. The first shows the effect of processing the first frame, then for each subsequent frame attempting to divide any superpixels once more. This captured small changes between the frames but left the data incoherent by the final frame. The second method was similar, but relied on having a background image of the scene to act as the base superpixel representation. Each frame of the video has an additional superpixel divide applied, which again results in small changes being detected. These methods would be suitable in a constrained



(a) Subsequent frame processing



(b) Background pre-processing

Figure 5.2: Experiments on different types of video processing

environment such as the Southampton Biometric Tunnel [Seely et al., 2008],where the images were obtained, but not in an unknown scene.

In addition, by treating video frames as a volume, superpixels that persist through time could be established. This could allow a new method of object tracking to be developed.

5.3 Graph Matching

Graph matching is the process of determining if two graphs are similar or not. Exact graph matching is determining if there is a perfect match between the vertices of the graphs. This is termed isomorphism. In a superpixel approach it is extremely unlikely that exact graph matching can occur as the number of nodes in both graphs must match. The type of matching that should be considered is inexact graph matching, where some or most of the vertices match. This problem is typical of vision graph problems, as usually it is only a smaller section (subgraph) of the image that one is interested in. The topic of interest is therefore *subgraph isomorpism*: the task of locating a map between a subgraph of the image and a graphical model of a set of features. However, subgraph isomorphism is an NP-complete problem.

There is surprisingly little research in this field. The first citation is that of [Ullmann, 1976]. The best algorithm is considered to be [Cordella et al., 2004] finding application in several standard libraries such as *networkx*[1]. Perhaps the reason for such a lack of interest is the sheer complexity involved in matching graph nodes when a node exists for each image pixel. A node per superpixel is much more feasible.

Face recognition is one such area where graph matching could be of use for example [Wiskott et al., 1997] where an elastic graph is used.

[1]http://networkx.lanl.gov

Chapter 6

Conclusion

This thesis has developed a new superpixel algorithm that displays excellent visual reconstruction of the original image. In addition it achieves high stability across multiple random initialisations. This is achieved by producing superpixels of multiple sizes where the size corresponds to local image complexity; accomplished by growing superpixels and dividing them on image variation. Once the algorithm was developed, it was found that the existing analysis was not sufficient to take these properties into account. New measures of oversegmentation, along with an analysis similar to precision-recall, provides new insight into the optimum superpixel representation. In performing the analysis, it was discovered that CDS has properties that have eluded previous algorithms, such as invariance and stability. The way in which the superpixels are generated, by the image content, is believed to be attributed to these properties. It was found that CDS has the unusual property that less detailed images give worse results than more complex images.

The new superpixel algorithm and its novel properties facilitate new applications for superpixel pre-processing. Three applications were studied. The first, image segmentation, considered pairwise superpixel comparison by treating the superpixels as a weighted graph that could be cut into segments. The second, image compression, developed a new compression algorithm and tested it across 400 images of the same size. The results show that despite varying superpixel numbers, the compression was reliable and displayed little variation across the image set. The last, scene classification, developed multiple feature vectors that could be used as image descriptors. These were found to perform better in a Bag of Features model than N-Cuts superpixels and SIFT descriptors.

The new properties also allowed for the generation of a new algorithm to detect localised regions of image focus. This work included several novel aspects. The first being that this is the only use of superpixels in such an application. The second being that this is the first use of a principled analysis to evaluate image focus. It was possible to show that by adjusting the focal plane of a Light-Field image that the superpixel regions corresponded directly to that focal plane.

This thesis has developed advancements in the fields of superpixel generation; application; and analysis. It motivates further study in each of these areas with the hope of further benefit to the field of computer vision.

References

R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. PAMI*, 2012.

E. H. Adelson and J. Y. A. Wang. Single lens stereo with a plenoptic camera. *IEEE Trans. PAMI*, 14(2):99–106, 1992.

A. Ayvaci and S. Soatto. Motion segmentation with occlusions on the superpixel graph. In *ICCV Workshops, 2009 IEEE 12th Int.Conf.*, pages 727–734, 2009.

S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. PAMI*, 24(4):509–522, 2002.

F. Bergholm. Edge Focusing. *IEEE Trans. PAMI*, 9(6):726–741, 1987.

G. Borgefors. Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing*, 34(3):344–371, 1986.

J. Canny. A Computational Approach to Edge Detection. *IEEE Trans. PAMI*, PAMI-8 (6):679–698, 1986.

V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *IJCV*, 22(1):61–79, 1997.

T Chan, BY Sandberg, and L Vese. Active Contours without Edges for Vector-Valued Images. *Visual Communication and Image Representation*, 11(2):130, 2000.

T. F. Chan and L. A. Vese. Active Contours Without Edges. *IEEE Trans. Image Processing*, 10(2), 2001.

M.J Chen and A. C. Bovik. No-reference image blur assessment using multiscale gradient. In *Int. Workshop, QoMEx* , pages 70–74, 2009.

L. D. Cohen, I. Cohen, and P. I. X. Ceremade. A finite element method applied to new active contour models and 3D reconstruction from cross sections. In *Computer Vision* , pages 587–591, 1990.

D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. PAMI*, 24(5):603–619, 2002.

L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub) graph isomorphism algorithm for matching large graphs. *IEEE Trans. PAMI*, 26(10):1367–1372, 2004.

L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *CVPR*, pages 524–531, 2005.

P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2):167–181, 2004.

B. Fulkerson, A. Vedaldi, and S. Soatto. Class segmentation and object localization with superpixel neighborhoods. In *Computer Vision*, pages 670–677, 2010.

D. Hoiem, A. A. Efros, and M. Hebert. Geometric context from a single image. In *IEEE ICCV*, 2005a.

D. Hoiem, A. A. Efros, and M. Hebert. Automatic photo pop-up. *ACM Trans. Graphics*, 24(3):584, 2005b.

T Kadir and M Brady. Saliency, Scale and Image Description. *IJCV*, 45(2):83–105, 2001.

M. Kass, A. Witkin, and D. Terzopoulos. snakes - active contour models. *IJCV*, 1(4): 321–331, 1987.

J. Kaufhold, R. Collins, A. Hoogs, and P. Rondot. Recognition and Segmentation of Scene Content using Region-Based Classification. In *ICPR*, pages 755–760, 2006.

L. Kovacs and T. Sziranyi. Focus area extraction by blind deconvolution for defining regions of interest. *IEEE Trans. PAMI*, 29(6):1080–1085, 2007.

A. Levin. Blind motion deblurring using image statistics. *Advances in Neural Information Processing Systems*, 19:841, 2007.

A. Levinshtein, A. Stere, K. N. Kutulakos, D. J. Fleet, S. J. Dickinson, and K. Siddiqi. TurboPixels: Fast Superpixels Using Geometric Flows. *IEEE Trans. PAMI*, 31(12): 2290–2297, 2009.

M. Levoy and P. Hanrahan. Light field rendering. In *Proc. CGIT*, pages 31–42, 1996.

T. Lindeberg. Scale-space theory: A basic tool for analyzing structures at different scales. *Journal of Applied Statistics*, 21(1):225–270, 1994.

M-Y. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa. Entropy rate superpixel segmentation. In *CVPR*, 2011.

R. Liu, Z. Li, and J. Jia. Image partial blur detection and classification. In *CVPR*, pages 1–8, 2008.

D.G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *Computer Vision*, 60(2):91–110, 2004.

R.J. Lowe and M.S. Nixon. Evolving Content-Driven Superpixels for Accurate Image Representation. In *ISVC2011*, pages 192–201, 2011.

R.J. Lowe and M.S. Nixon. Detecting Focal Regions using Superpixels. In *VISAPP*, 2013.

D. Martin, C. Fowlkes, D. Tal, and J. Malik. A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. In *Computer Vision*, pages 416–423, 2001.

F. Meyer and S. Beucher. Morphological segmentation. *Journal of visual communication and image representation*, 1(1):21–46, 1990.

K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Trans. PAMI*, 27(10):1615–1630, 2005.

A P. Moore, S. J. D. Prince, J. Warrell, U. Mohammed, and G. Jones. Superpixel lattices. In *CVPR*, pages 998–1005. IEEE, 2008.

A. P. Moore, S. J. D. Prince, and J. Warrell. "Lattice Cut"-Constructing superpixels using layer constraints. In *CVPR*, pages 2117–2124. IEEE, 2010.

D. Mumford and J. Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Comm. Pure Appl. Math*, 42(5):577–685, 1989.

S. K. Nayar and Y. Nakagawa. Shape from focus. *IEEE Trans. PAMI*, 16(8):824–831, 1994.

R. Ng, M. Levoy, M. Brédif, G. Duval, M. Horowitz, and P. Hanrahan. Light field photography with a hand-held plenoptic camera. *CSTR*, 2, 2005.

S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *Journal of computational physics*, 79 (1):12–49, 1988.

C. Rasmussen. Superpixel analysis for object detection and tracking with application to UAV imagery. In *Int. Conf. Advances in Visual Computing*, pages 46–55, 2007.

X. Ren and J. Malik. Learning a classification model for segmentation. In *Computer Vision*, pages 10–17, 2003.

X. Ren, A. C. Berg, and J. Malik. Recovering human body configurations using pairwise constraints between parts. In *Proc. ICCV*, 2005.

C. Rohkohl and K. Engel. Efficient image segmentation using pairwise pixel similarities. In *Pattern Recognition*, pages 254–63, 2007.

R. D. Seely, S. Samangooei, M. Lee, J. N. Carter, and M. S. Nixon. The University of Southampton Multi-Biometric Tunnel and introducing a novel 3D gait dataset. In *BTAS*, pages 1–6, 2008.

L. G. Shapiro and G. C. Stockman. *Computer Vision.* Prentice Hall, 2001.

M. Shemesh and O. Ben-Shahar. Free boundary conditions active contours with applications for vision. *Trans. ISVC*, pages 180–191, 2011.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. PAMI*, 22 (8):888–905, 2000.

J. Sivic and A. Zisserman. Video Google: a text retrieval approach to object matching in videos. In *Computer Vision*, pages 1470–1477, 2003.

Y.W Tai and M. S. Brown. Single image defocus map estimation using local contrast prior. In *ICIP*, pages 1797–1800, 2009.

T. Tuytelaars and K. Mikolajczyk. Local Invariant Feature Detectors: A Survey. *Foundations and Trends in Computer Graphics and Vision*, 3(3):177, 2007.

J. R. Ullmann. An Algorithm for Subgraph Isomorphism. *J. ACM*, 23(1):31–42, 1976.

A. Van den Hengel, A. Dick, T. Thormählen, B. Ward, and P. H. S. Torr. VideoTrace. *ACM Trans. Graph.*, 26(3):86, 2007.

A. Vedaldi and S. Soatto. Quick shift and kernel methods for mode seeking. *ECCV*, pages 705–718, 2008.

S. V. Viraktamath and G. V. Attimarad. Performance analysis of JPEG algorithm. In *ICSCCN*, pages 629–633, 2011.

D. Wang, N. M. Kwok, X. Jia, and G. Fang. A Cellular Automata approach for superpixel segmentation. In *CISP*, 2011.

J. Wang and X. Wang. VCells: Simple and Efficient Superpixels Using Edge-Weighted Centroidal Voronoi Tessellations. *IEEE Trans. PAMI*, 34(6):1241–1247, 2012.

B. Wilburn, N. Joshi, V. Vaish, E. V. Talvala, E. Antunez, A. Barth, A. Adams, M. Horowitz, and M. Levoy. High performance imaging using large camera arrays. *ACM Trans. Graph.*, 24(3):765–776, 2005.

L. Wiskott, J. M. Fellous, N. Kuiger, and C. von der Malsburg. Face recognition by elastic bunch graph matching. *IEEE Trans. PAMI*, 19(7):775–779, 1997.

A.P. Witkin. Scale-space filtering. *Intl. Joint Conf. Art. Intell.*, 2:1019–1022, 1983.

C. Xu and J. L. Prince. Gradient vector flow: A new external force for snakes. In *CVPR*, 1997.

C. Xu and Jerry L. Prince. Snakes, Shapes, and Gradient Vector Flow. *IEEE Trans. Image Processing*, 7(3), 1998.

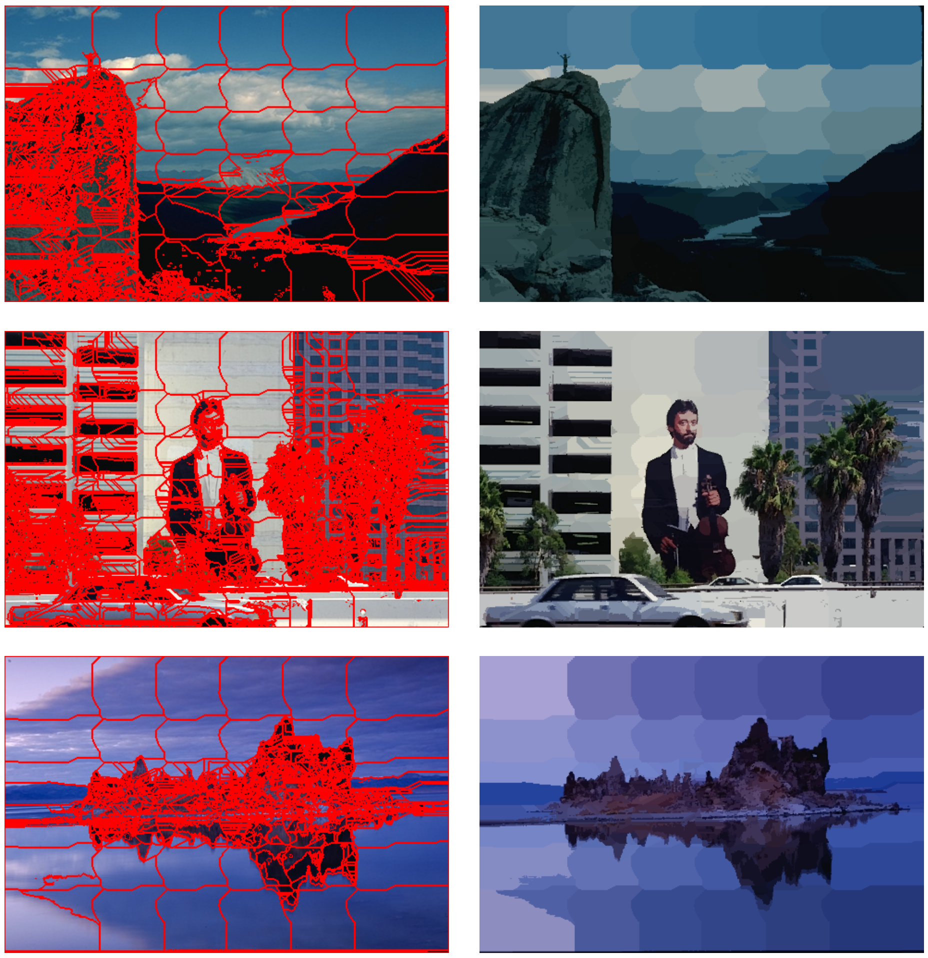
J Yang, Y.G. Jiang, A.G. Hauptmann, and C.W Ngo. Evaluating bag-of-visual-words representations in scene classification. In *Proc. Int. Workshop on multimedia information retrieval*, pages 197–206. ACM, 2007.

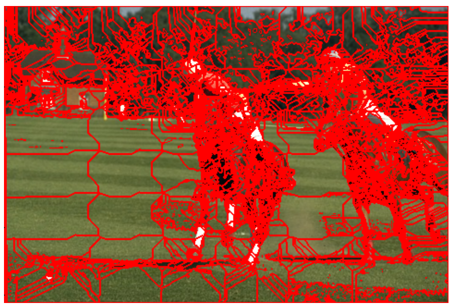
Y. Zhang, R. Hartley, J. Mashford, and S. Burn. Superpixels via pseudo-boolean optimization. In *ICCV*, pages 1387–1394, 2011.

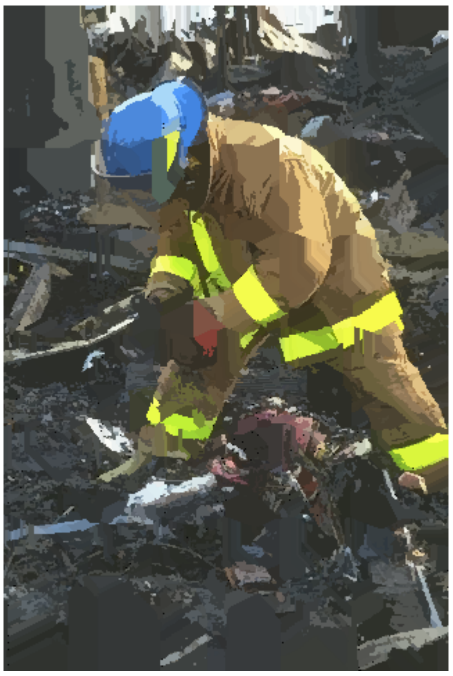
Appendix A

Additional Results

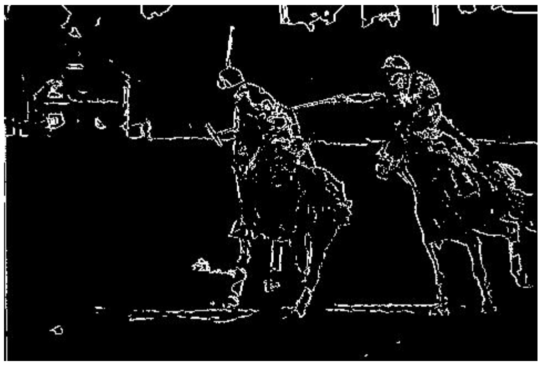
A.1 CDS Reconstruction

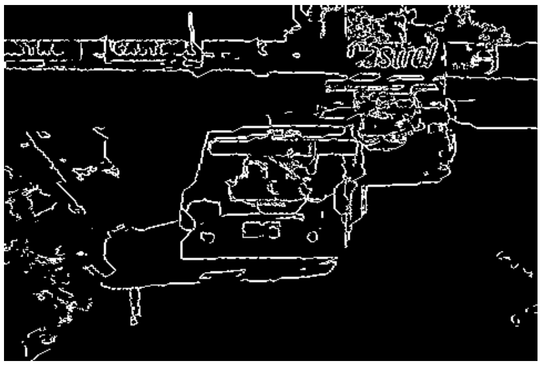


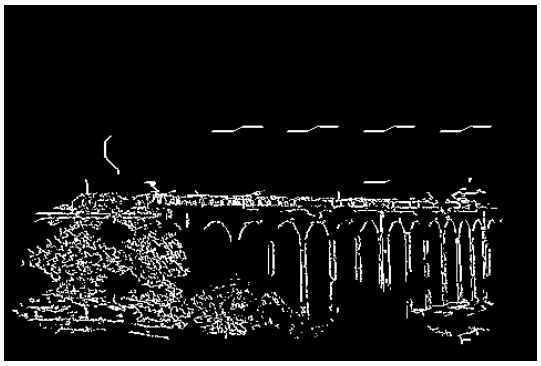


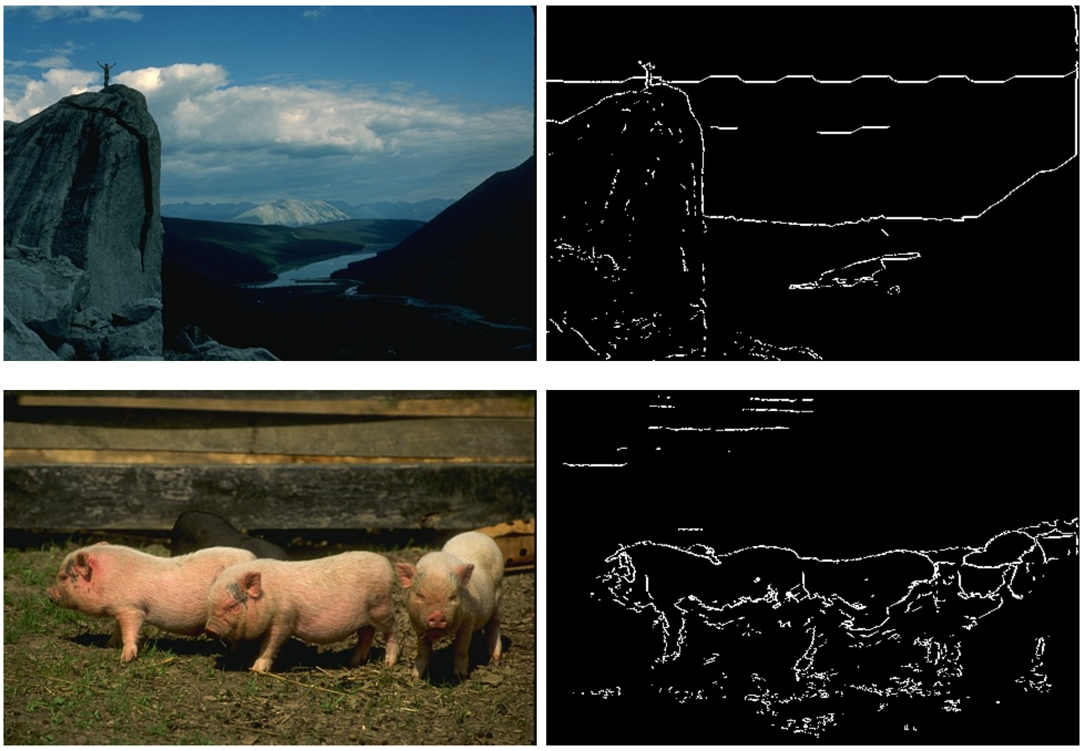


A.2 Segmentation Results









Appendix B

Description of the Code

B.1 Data Structures

There are two classes used to control CDS: The superpixel class and the container class that contains all the superpixels.

Each superpixel class contains two data elements. One contains a binary image that is True for any pixel that is contained within the superpixel and False otherwise. The other contains an offset to the first pixel that contains any data, reducing space required by the binary image. Also contained within the superpixel class are the grow and divide methods. If a superpixel grows outside of the size of its binary image the image is extended to fit the new pixels. If this is in the direction of the origin the offset value is adjusted accordingly. When a superpixel divides it creates a new superpixel within the method and returns this object to be stored within the container object.

The container class has a list of all the superpixels and has methods that allow every superpixel to grow by computing the image showing all the superpixels, which prevents overlap. It also has a method to allow every superpixel to divide simultaneously. These methods are threaded allowing multiple cores to take on a proportion of the processing. This class also contains methods to draw all the superpixel edges or the mean colour of the superpixels, in other words the reconstruction of the image.

B.2 Pseudocode

```
input: location of file: filepath, number of seeds: seeds
output: array containing superpixel labels: superpixels

im=open(filepath)
superpixels=zeros(im.height,im.width)
previous=np.ones(im.height,im.width)

label=1
for i in range(1,seeds):
  ipos=i*im.height/float(seeds)
  for j in range(1,seeds):
    jpos = j*im.width/float(seeds)
    superpixels[ipos,jpos]=label
    label+=1

while (previous!=superpixels):
  previous=superpixels

  transform, neighbours = perform distance transform on image

  for superpixelLabel in range(1,superpixels.max()):

    get superpixel array where superpixels==superpixelLabel

    for i in range(superpixel.height):
      ioff=i*superpixel.width
      for j in range(superpixel.width):
        if 1==superpixel[i,j]:
          if 1==transform[ioff+j]:
            superpixels at neighbour[ioff+j]=superpixelLabel

  #ACWE

    phi = compute ACWE using superpixel array

    if phi.max()>0 and phi.min()<0:

      newLabel=superpixels.max()+1

      for i in range(superpixel.height):
        ioff=i*superpixel.width
        for j in xrange(superpixel.width):
          if 1==superpixel[i,j]:
            if 0>=phi[ioff+j]:
              superpixels[i,j]=newLabel
              #if phi>0 it still belongs to same label

return superpixels
```

```
input: 1D image: image, image height: height, image width: width

output: 1D array showing distance of each pixel to background: dtNBE, 1D array showing nearest backgrou


dtNBE=np.ones_like(image)
dtNBE*=255
nNBE=np.ones_like(image)
```

```
nNBE*=255


colourToAvoid=255
ONE=41
SQRT2=58
sqrt5=92
MAX=255

for j in range(0,height):
  joff=j*width;
  for i in range(0,width):
    if (image[joff+i]!=colourToAvoid):
      dtNBE[joff+i]=0
      nNBE[joff+i]=joff+i
    else:
      if (i==0):
        dtNBE[joff+i]=MAX
        nNBE[joff+i]=-1
      else:
        if (dtNBE[i-1+joff]<MAX):
          dtNBE[joff+i]=dtNBE[i-1+joff]+ONE
          nNBE[joff+i]=nNBE[i-1+joff]
        else:
          dtNBE[joff+i]=MAX
          nNBE[joff+i]=-1

      if (j!=0):
        if ((dtNBE[joff+i-width]+ONE)<dtNBE[joff+i]):
          dtNBE[joff+i]=dtNBE[i+joff-width]+ONE
          nNBE[joff+i]=nNBE[joff+i-width]
        if ((i!=0)&((dtNBE[i-1+joff-width]+SQRT2)<dtNBE[joff+i])):
          dtNBE[joff+i] = dtNBE[i-1+joff-width]+SQRT2
          nNBE[joff+i] = nNBE[i-1+joff-width]
        if ((i!=width-1)&((dtNBE[i+1+joff-width]+SQRT2)<dtNBE[i+joff])):
          dtNBE[i+joff] = dtNBE[i+1+joff-width]+SQRT2
          nNBE[i+joff] = nNBE[i+1+joff-width]

for j in range(height-1,0,-1):
  joff=j*width;
  for i in range(width-1,0,-1):
    if (image[i+joff]==colourToAvoid):
      if (i!=width-1):
        if ((dtNBE[i+1+joff]+ONE)<dtNBE[i+joff]):
          dtNBE[i+joff]=dtNBE[i+1+joff]+ONE
          nNBE[i+joff]=nNBE[i+1+joff]
      if (j!=height-1):
        if ((dtNBE[i+joff+width]+ONE)<dtNBE[i+joff]):
          dtNBE[i+joff]=dtNBE[i+joff+width]+ONE
          nNBE[i+joff]=nNBE[i+joff+width]
        if ((i!=0)&((dtNBE[i-1+joff+width]+SQRT2)<dtNBE[i+joff])):
          dtNBE[i+joff]=dtNBE[i-1+joff+width]+SQRT2
          nNBE[i+joff]=nNBE[i-1+joff+width]
        if ((i!=width-1)&((dtNBE[i+1+joff+width]+SQRT2)<dtNBE[i+joff])):
          dtNBE[i+joff]=dtNBE[i+1+joff+width]+SQRT2
          nNBE[i+joff]=nNBE[i+1+joff+width]

return dtNBE,nNBE
```

```
Algorithm 3: ACWE

Input: 1D image: im, superpixel locations: locations, number of colours: ndim
Output: Distance function: phi

F = zeros_like(locations)
phi=zeros_like(locations)
oldphi=ones_like(locations)

while (oldphi!=phi):

  oldphi=phi

  ucount=0
  vcount=0
  umean=zeros(3)
  vmean=zeros(3)

  for i in range(0,len(im)) step ndim:
    if (1==locations[i/ndim]):
      if (phi[i/ndim]<=0):
        for j in range(0,ndim):
          umean[j]+=im[i+j]
        ucount+=1
      else:
        for j in range(0,ndim):
          vmean[j]+=im[i+j]
        vcount+=1

  for j in range(0,ndim):
    umean[j]=umean[j]/float(ucount+0.0000001)
    vmean[j]=vmean[j]/float(vcount+0.0000001)

  for i in range(0,len(im)):
    F[i/ndim]+= (im[i]-umean[i%ndim])**2 /float(ndim)
    F[i/ndim]-= (im[i]-vmean[i%ndim])**2 /float(ndim)

    phi[i/ndim]+=F[i/ndim]

return phi
```