Chapter 5

Contour

The previous chapter focussed on developing an edge detector that could define boundaries or contours at each pixel for the purpose of contour extraction. In this chapter the edge points from the Gaussian thinned Asymmetry edge detector are linked together to form whole contours. As shown in Figure 1.4, contour extraction in this research is designed to be used for higher-level feature extraction such as region identification. Much of the hard work in contour extraction has been addressed with the edge detector of the previous chapter and all that remains is to use the edge responses to form whole contours.

The challenge in contour extraction is to extract whole independent contours. That is, the contours extracted should not be split unnecessarily but should also not be joined with other contours. The edge features of the previous chapter aid the contour extraction process in that multiple orientation responses are provided at each edge point. Multiple edge responses provide two advantages. The first is that the exact edge orientation can be determined by interpolating between adjacent orientation responses allowing more accurate orientation comparisons between edge points. The second is that multiple edges that cross the same point can also be represented allowing contours to co-terminate or cross the same point independently. In this chapter a new contour extraction technique is presented based on the local processing edge linking approach [69] that takes advantages of the edge features of the previous chapter.

Before regions can be identified, additional geometric features must be extracted. In this chapter vertex extraction is briefly discussed and a new neurophysiologically-based vertex detector is presented which is also based on the Asymmetry edge detector.

The second half of the chapter is dedicated to contour-based image similarity techniques. Existing techniques are discussed and are found to be not suitable for comparing whole image contours. Two new contour matching techniques are presented and their performance is compared with the Hausdorff distance [117]. Finally, a new combined colour and contour representation is presented that is more compact than the other representations but provides comparable results.
Section 5.1 presents existing techniques for extracting contours. Section 5.2 identifies the requirements of contour extraction. Section 5.3 reviews existing techniques for identifying edge points. Section 5.4 presents a new technique for identifying edge points whilst preserving the true orientation of the edge. Section 5.5 presents and evaluates a new approach for linking multi-orientation edge points. Section 5.6 discusses vertex extraction and presents a new vertex extraction technique based on the Asymmetry detector. Section 5.7 discusses the problem of using contours as a retrieval feature. Section 5.8 investigates using the Hausdorff distance measure as a contour-based image similarity metric. Section 5.9 presents a new technique for determining image similarity based on contour by creating and comparing contour summaries. Section 5.10 presents a new approach for representing and comparing contours using histograms. Section 5.11 presents the results of combining both colour and contour histograms. Section 5.12 presents conclusions drawn from the findings of this chapter.

5.1 Contour Extraction

Reviews of contour extraction such as Gonzalez and Woods [69] generally begin with a quick description of the local processing approach followed by a detailed analysis of the Hough transform. In this section we will also describe both techniques but show that the local processing approach is more flexible than the Hough transform but needs more work before successful contours can be extracted.

5.1.1 Local Processing

The local processing method of linking edge points into contours involves analysing a small neighbourhood of pixels and linking neighbouring points that have similar orientations to the central pixel. Gonzalez and Woods [69] identify two properties for joining edge pixels into a contour: (1) the strength of the response of the gradient operator, and (2) the direction of the gradient.

The first property determines that two edges are similar if the magnitude of the gradient response is similar. If \( \nabla f(x', y') \) is the magnitude of the gradient at neighbouring point \((x', y')\) and \((x, y)\) is the centre of the neighbourhood then the neighbour is part of the contour if

\[
|\nabla f(x, y) - \nabla f(x', y')| \leq T
\]

(5.1)

where \( T \) is the predefined magnitude difference threshold.

Using the second property, two edges are considered similar if the difference in their angles is less than a predefined threshold \( A \):

\[
|\alpha(x, y) - \alpha(x', y')| < A
\]

(5.2)

There are two limitations with the approach presented in Gonzalez and Woods [69]. Firstly, the approach assumes that there is only one gradient direction per pixel when in fact two or more
contours may cross each other at the same pixel. Secondly, there is no consideration of the position of the pixel in the neighbourhood and the relative directions of gradients. The local processing approach is suitable for contour extraction, but more work can be done to extend the technique to support Asymmetry edge detector responses and produce contours suitable for video retrieval.

5.1.2 Hough Transform

More research appears to have been performed with investigating the Hough transform [118, 119, 120, 69] compared with the local processing approach due to the motivation for performing pattern recognition rather than contour representation. The local processing approach can be described as an approach that can extract arbitrary contour shapes whereas the Hough transform extracts contours that conform to predefined shape functions.

The Hough transform begins with a shape function to be detected in the image, such as a line:

\[ y = ax + b \] (5.3)

The shape function will have parameters, such as \( a \) and \( b \) in this case. The parameters form a parameter space that can be laid out in multiple dimensions. A straight line has two parameters and therefore all possible lines can be described by a point in two dimensions in the parameter space.

Every pair of edge pixels in the edge image are substituted into the shape function to determine the parameters of the shape that passes through both edge pixels. The parameter space becomes a histogram and the bin that represents the parameters of the line is incremented. After processing, the value of each bin represents the number of edge points that contributed to that particular shape. Thresholding can be used to determine significant shapes and the resulting parameter points can be used to reconstruct the edge image with only the significant shapes.

The gradient-offset line equation of Equation 5.3 is generally not used because the gradient \( a \) approaches infinity as the line approaches \( 90^\circ \) making uniform histogram construction difficult. The gradient problem can be avoided by using polar co-ordinates:

\[ x \cos \theta + y \sin \theta = \rho \] (5.4)

\( \rho \) will be no greater than half the diagonal of the original image and \( \theta \) will range from \(-90^\circ \) to \( 90^\circ \). An example of the Hough transform into polar co-ordinates of an image containing lines of various angles and positions is shown in Figure 5.1. Four dense clusters are formed in the parameter space representing the four different line equations present in the original image.

Other shape functions such as the circle can be used which result in a three dimensional parameter space due to three parameters in the shape function:

\[ (x - a)^2 + (y - b)^2 = c^2 \] (5.5)
The primary limitation with the Hough transform is that it searches for predefined shapes. Any shape that is not, for example, a perfectly straight line or circle, will be misrepresented. In addition, the parameter space can increase to multiple dimensions even for relatively simple shapes that would easily be extracted using local processing techniques. Therefore, the Hough transform is not suitable for producing an accurate representation of contours suitable for content-based video retrieval.

5.2 Contour Extraction Requirements

As seen in the last section current contour extraction techniques have their limitations. Before a new technique can be developed we need to determine the requirements of a contour extraction technique in the context of image and video retrieval. We have identified the following three requirements for contour extraction. Firstly, relatively arbitrary contours must be representable. This is important because most natural contours do not follow a simple analytic formulation or vector description, such as a straight line or an arc. Secondly, the edge responses of the previous chapter are very precise and unambiguous, therefore the contours extracted should reflect the same level of precision and non-ambiguity. Thirdly, even though the contours may be of arbitrary shape they must not contain sharp edges, which is an indication of two contours joining. Based on these three requirements the local processing approach of the previous section is much more suitable for video retrieval than the Hough transform. The following sections take the local processing approach and expand and refine it to produce the contours that are required.

5.3 Identifying Edge Points

The first stage of the local processing approach is to identify edges that occur at each pixel in the edge image. In the local processing approach described in [69] it is assumed that there is only one edge orientation per pixel and therefore the only criteria for identifying the presence of an edge
point is whether the magnitude is greater than a predetermined threshold (Equation 5.1). The orientation of the edge point simply becomes the angle of the gradient vector, where the gradient vector is composed of the individual gradients along the $x$ and $y$ axes:

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

(5.6)

from vector analysis the angle of the gradient is:

$$\alpha(x, y) = \tan^{-1} \left( \frac{G_y}{G_x} \right)$$

(5.7)

However, this approach is not suitable for multi-orientation edge responses such as those provided by the technique presented in the previous chapter. The first reason is that the orientation responses are not ‘thinned,’ that is, multiple adjacent orientations may have magnitudes greater than the predetermined magnitude threshold. However, the orientation responses can’t simply be thinned because the orientation responses adjacent to the peak orientation response are required so that the true orientation of the edge can be interpolated. So the process of selecting the peak orientations and computing the true orientation of the edge must occur simultaneously.

### 5.4 New Contour Extraction Approach

The process for extracting an edge must first begin by finding the peak orientation responses at a pixel. A peak orientation response is identified when its response is greater than the predetermined threshold and its adjacent responses have a lower magnitude than the peak response. We have found that ensuring that all orientation responses $45^\circ$ in either direction (+/-3 orientation steps) have a lower magnitude than the central orientation reduces the effect of noise on the results. This process of ‘thinning’ along the orientation curve is similar to orientation competition that occurs in the visual cortex. However, it also means that edges crossing at the same point must differ by $45^\circ$ to be detected.

#### 5.4.1 True Orientation

Once a peak orientation has been detected the true orientation of the edge is determined. Since there are only a discrete number of oriented edge detectors, the true orientation of the edge must be interpolated from the adjacent responses. Figure 5.2 (a)-(d) shows four scenarios that need to be considered when calculating the true orientation.

The first scenario (Figure 5.2 (a)) is simple to evaluate with the peak orientation being the orientation of the edge (15°) as both adjacent orientations are zero. The second scenario (Figure 5.2 (b)) is also simple to evaluate as both adjacent orientation responses are of the same magnitude therefore the true orientation of the edge is the bisector between both orientations, 22.5°. The
third scenario (Figure 5.2 (c)) is more complex as the true orientation of the edge is a proportional
distance between the 15° and 30° orientations. The small response at 0° also indicates that the
true orientation of the edge may be slightly closer to 15° than just the 15° and 30° responses may
indicate. Figure 5.2 (d) shows that not only the peak orientation nor only two orientations but all
three orientations must be considered when calculating the true orientation.

The true orientation of an edge will either lie directly on an edge detector orientation or between
two edge detector orientations. So at most, only two values are required to interpolate the true
orientation. Therefore, the three orientation responses must be reduced to two. Looking at Figure
5.2 (d) we can subtract one of the smaller responses from the other two responses to produce the
result in Figure 5.2 (h). Now if the peak orientation and one of the smaller responses of Figure
5.2 (h) is used to interpolate the true orientation then the result will be 15° because both other
responses are now zero. Figure 5.2 (g) also more accurately indicates that the true orientation is
closer to 15° than the two orientation responses indicate in Figure 5.2 (c), whilst Figures 5.2 (e)
and (f) remained unchanged as they should be. The algorithm for computing the true orientation
response for a peak orientation is shown in Algorithm 2. With the complete process for extracting
edge points shown in Algorithm 3.

After computing the true orientation a complete edge point can be described in terms of location
and direction. The approach described above improves upon the standard local processing approach
with the following features:
Algorithm 2 Calculate the true orientation for a peak orientation response $o_i$.

$$r_3 \leftarrow \min(o_{i-1}, o_{i+1})$$

if $o_{i-1} < o_{i+1}$ then

$$r_2 \leftarrow o_{i+1}$$
$$d \leftarrow 1$$

else

$$r_2 \leftarrow o_{i-1}$$
$$d \leftarrow -1$$

end if

Subtract the minimum:

$$r_1 \leftarrow o_i - r_3$$
$$r_2 \leftarrow r_2 - r_3$$

$$\theta \leftarrow \left( i + d \frac{r_2}{r_1 + r_2} \right) \frac{\pi}{N} \ (N \text{ refers to the total number of orientation responses})$$

Algorithm 3 Extracting all edge points from an edge image (a pixel may contain multiple edge points of different orientations).

for all $i$: pixels in edge image, $i$ represents pixel location do

for all $o_j$: orientation responses at $i \geq$ SEED THRESHOLD, $j$ represents orientation index do

An orientation can only create a new edge point if it is the largest within its neighbours

if $|o_j| \geq |o_{j-a}| : -3 \leq a \geq 3, a \neq 0$ then

$$\theta \leftarrow \text{true orientation of } o_j \ (\text{See Algorithm 2})$$

Create a new edge point $p$ at location $i$ with orientation $\theta$

end if

end for

end for
5.5 New Edge Linking Approach

Once all of the edge points in the image have been determined they can be linked. The local processing approach [69] only links points that have a similar magnitude and direction. This approach is limited for a number of reasons. Firstly, an edge may be formed between a foreground object and a patterned background. The varying background colour may cause a variation in the magnitude of the edge response along the contour even though there are no breaks in the edge. Therefore, we have removed the first criteria of having a similar magnitude and only require that the magnitude of each edge response be above the predetermined seed threshold (12), which, by this stage, all edge points will satisfy. Secondly, even though it is important that a pair of linked edge points have similar orientation, this criteria is too flexible and may result in incorrect edges being linked. Consider Figure 5.3 for example. If only similar orientation is considered, points (a) and (b) would be incorrectly linked to each other. The additional criteria of relative location with respect to orientation needs to be considered.

The angle of the relative location of a neighbour to the edge point being considered is simple to compute. The angle begins at 0° at location (1, 0) and increases in 45° increments for each neighbour in the counter-clockwise direction. This relative location is only relevant with respect to the orientation of the centre edge point. We have found that allowing a 45° difference (this is the location threshold, $A_L$) between the edge point’s orientation and the angle of relative location is
suitable for edge linking as it allows a contour to deviate one pixel to the left or right when moving in the direction of the contour. Figure 5.4 shows the angle that is formed between the orientation of the edge point and the angle of neighbouring relative locations.

The next step is to determine which edge orientation at the pixel location to link to, if any. Firstly, neighbouring edge points are only considered whose difference in orientation from the central edge point is less than 30° (this is the orientation threshold, \(A_O\)). Secondly, the link strength for each edge point must be greater than a predetermined threshold, which is the same as the seed threshold. The link strength modulates the neighbouring edge point’s strength based on the difference between the orientations of the neighbouring edge point and the central edge point. A Gaussian function with a bandwidth of 45° is used to modulate the strengths of the neighbours:

\[
l = ne^{-\frac{(\theta_n - \theta_c)^2}{b^2}}, \quad b = 45°
\]  

where \(l\) is the resulting link strength, \(n\) is the strength of the neighbour edge point, \(\theta_n\) is the orientation of the neighbouring edge point, \(\theta_c\) is the orientation of the central edge point, and \(b\) is the bandwidth of the Gaussian function. If there is no difference between the two orientations then the neighbouring edge point’s strength will not be affected. If the difference is 45° then the neighbouring edge point’s strength will be diminished by almost two thirds. Since the link threshold is the same as the seed threshold a neighbouring edge point’s strength must be quite large to overcome the modulating effect of a 45° difference in orientation.

Once a neighbour has been identified for linking it must be determined whether the edge point is already part of an existing contour. If the neighbouring edge point is already part of an existing contour then the question arises as to why the central edge point wasn’t considered linkable to it when that contour was being followed. The answer is because the link strength is based on the neighbour’s strength, \(n\), so depending on which direction the contour is being trace, either from central edge point to neighbour, or from neighbour to central edge point, a different link strength will be determined. This is a reasonable side-effect because a strong edge may not consider a weaker edge worthy of being linked because of the difference in orientation. However, a weaker edge may
have been linked to another edge because of the similarity in orientation and now the weaker edge is part of a greater contour and is able to link the stronger edge to itself. This is a basic form of medium-level perceptual grouping similar to that which occurs in the visual cortex.

If the neighbouring point already exists in another contour then the two contours become linked between the two edge points (but still remain independent contours). If the point doesn’t exist then it is simply added to the existing contour being traced. The complete edge linking algorithm is described in Algorithms 4 and 5.

**Algorithm 4** Follow contour starting at edge point \( p \)

\[ \theta_p \leftarrow \text{orientation of edge point } p \]

for all \( n_i \): neighbouring pixels of \( p \) in \( 3 \times 3 \) neighbourhood do

\[ \theta_i \leftarrow \text{orientation of edge point } n_i \]

\[ \phi_i \leftarrow \text{the angle of the vector } p \rightarrow n_i \]

if \( |\theta_p - \phi_i| \leq A_L \text{ AND } |\theta_p - \theta_i| \leq A_O \) then

Find the strongest edge at location \( i \) that can be linked to

Begin by computing the strength of the link for each edge

for all \( p_j \): edge points at location \( i \), \( j \) represents edge index do

Link strength \( l_j \) depends on the strength of \( p_j \) and the difference in orientation between \( p_j \) and \( p \)

\[ l_j \leftarrow p_j e^{\frac{(\theta_p - \theta_j)^2}{A_L^2}} \]

end for

\[ l_{max} \leftarrow \max(l) \]

\( p_{link} \leftarrow \text{edge point represented by } l_{max} \)

if \( l_{max} \geq \text{LINK THRESHOLD} \) then

if \( p_{link} \) is not already part of an existing contour then

Add \( p_{link} \) to contour

Continue following contour with edge point \( p_{link} \) by recursively calling this algorithm

else

Link \( p_{link} \) to this contour

end if

end if

end for

5.5.1 Edge Linking Experiments

The new edge linking approach presented in this chapter was compared with the conventional local processing approach [69]. Edge responses from the Asymmetry detector of the previous chapter were used as input for both edge linking techniques. The Sobel edge detector was also used to
Algorithm 5 Find all contours in an edge image

for all \( i \): pixels in edge image, \( i \) represents pixel location do

for all \( p_j \): edges at location \( i \), \( j \) represents edge index do

if \( p_j \) isn’t part of an existing contour (i.e. point at same position with difference in orientation \( \leq \pi/6 \)) then

Create a new contour \( c \) with edge point \( p_j \)

Follow contour \( c \) starting from \( p_j \) (Algorithm 4)

Add contour \( c \) to contour list

end if

end for

end for

compute the edge gradient which is conventionally used in the local processing approach described in Section 5.3, however the Sobel edge gradient was not used as input to the new edge linking approach as the new approach requires multi-orientation input.

A threshold of 12 was applied to the Asymmetry edge responses to reduce noise whereas a threshold of 64 was applied to the Sobel edge responses to compensate for the broader range and thicker responses of the Sobel edge detector. A maximum angular linking deviation of only 20° was used for linking edges from the Sobel responses because larger values allowed too many spurious links. A maximum angular deviation of 30° was used for the Asymmetry responses because they were more tightly tuned.

The conventional local processing approach assumes only one edge orientation per pixel therefore only one orientation was determined for each pixel from the multi-orientation Asymmetry edge responses by selecting the orientation with the largest response. The new edge linking approach allows multiple oriented edges at each pixel and therefore the orientations for the new edge linking approach were computed using the true orientation algorithm described in Section 5.4.1.

As outlined in Section 5.2 the requirements of a good edge linking algorithm are contours that do not contain sharp edges but may contain small variations in orientation throughout the contour. Edge linking techniques are difficult to evaluate quantitatively as there are many edge pixel scenarios. However, the new edge linking approach was significantly better than the conventional approach therefore a subjective analysis was more than adequate. The edge linking algorithms were evaluated using the standard plane image (Figure 5.5 (a)) and representative contours were analysed to determine the algorithm’s strengths and weaknesses.

5.5.2 Edge Linking Results

Figure 5.5 shows the edge images used as input for the edge linking algorithms. Figure 5.6 (a) shows the total contours extracted by the local processing approach when applied to the Sobel edge images and Figure 5.6 (b) shows the total contours extracted by the edge linking approach.
Figure 5.5: Input images for edge linking experiments: (a) Plane test image; (b) Sobel output; and (c) new edge detection output.

When applied to the new edge images, Figure 5.6 (b) contains more smaller contours than (a) because of the lower threshold used. Figures 5.6 (c) to (h) show the same contour extracted using both algorithms.

Figure 5.6 (c) shows a contour extracted using the local processing approach that forms part of a road on the airport tarmac. As can be seen in the original image the road is bounded by two horizontal contours. At no point along the road do they join. However, the local processing approach is not able to extract the two contours separately and also includes part of the tail wing in the contour. In contrast, the same contour extracted using the new edge linking approach (Figure 5.6 (d)) is one straight line that does not include the other bounding contour or any elements of the tail wing. In addition, the advantages of the new edge detection technique can also be seen with the contour only being one pixel thick as opposed to the two pixel thick Sobel line.

The second contour analysed (Figures 5.6 (e) and (f)) was difficult to extract accurately for both approaches with part of the tarmac contour joining with the top of the plane. However, the new edge linking approach was able to extract the full contour of the top of the plane up to the tail. The conventional local processing approach was thrown by a change in orientation caused by a nearby contour causing the contour to finish only part way down the top of the aircraft.

The final contour analysed is shown in Figures 5.6 (g) and (h). The new edge linking approach
Figure 5.6: Local processing edge linking results are contained in the left column and the new edge linking approach’s results are contained in the right column.
Figure 5.7: (a) The conventional local processing approach is applied to the new edge detector output and a contour is selected. (b) The same contour from the new edge linking algorithm.

is able to successfully extract the contour that goes down the middle of the aircraft whereas the conventional approach extracts many other contours of the aircraft including its wing and shadow.

These results show quite clearly that the local processing edge linking approach and Sobel edge detector combination perform quite poorly with natural images. However, its poor performance could be explained solely by the Sobel edge detector. Therefore the conventional local processing approach was also applied to the Asymmetry edge responses. A contour from both approaches is shown in Figure 5.7. In the background, in grey, it can be seen that both approaches extract roughly the same total contours as they use the same seed threshold (12). However, the contour selected to be analysed (in black) has not been extracted correctly by the conventional local processing approach. The road contour connects with and includes some of the tail contour. In contrast, the new edge linking approach successfully extracts the contour without containing any edge points from the tail of the aircraft.

In an effort to give the reader a broader idea of the kinds of contours extracted by the new edge linking algorithm Figure 5.8 shows a number of images that display varying length contours that were extracted by the new edge linking process.

5.5.3 Edge Linking Discussion

There are a number of reasons for the new edge linking approach’s successful extraction of contours from the plane image. Firstly, unlike the Sobel edge detector, the Asymmetry edge detector is designed specifically to satisfy the requirements of contour extraction including edge linking. As a result the Sobel edge detector performs much more poorly than the Asymmetry edge detector for edge linking. Secondly, because there are multiple orientation inputs the true orientation of each point may be computed without interference from other orientations at the same edge point. In contrast the gradient angle method must combine all edge stimulus into one angle approximation. Thirdly, the new edge linking approach considers the relative location of pixels with respect to the
Figure 5.8: Contours extracted from the Plane image using the new edge linking approach. Each image displays only the contours with lengths greater than (a) 1, (b) 2, (c) 3, (d) 5, (e) 10, (f) 20, (g) 50, and (h) 100 pixels.
orientation of each edge point reducing the number of spurious links such as the one in Figure 5.7 (a). Finally, the new edge linking approach modulates the strength of a neighbour point by the relative difference in orientation to allow strong edges more flexibility in orientation but less flexibility in orientation for weaker edges, again reducing spurious edge links. The results in Figure 5.6 and 5.7 show that the new edge linking approach is clearly better than the conventional local processing approach.

5.6 Vertices

The primary purpose of extracting contours within the image decomposition process presented in Figure 1.4 is to identify regions. Before regions can be extracted, contours must be grouped into region boundaries. Human vision is able to form regions from incomplete boundaries by filling in missing boundaries generating illusory contours [97]. Filling in illusory contours is a non-trivial process and requires edge and contour information surrounding the missing boundary. Biederman [96] found that shape recognition was pre-attentive and therefore must occur early on in human vision processing. Biederman [96] also conducted experiments that showed that vertices in line drawings were more important than the lines between vertices for shape recognition. Therefore when the lines were missing between vertices the brain was still able to join the vertices with an illusory contour to perform shape recognition. The ability for the brain to perform object recognition solely on vertices places a great deal of weight on vertices. Biederman [96] explained that features such as vertices, which are two or more lines terminating at a common point, are non-accidental in that they rarely occur in nature without describing some important characteristic of a distinguishable object. Biederman [96] used three vertices that occur in two and three dimensions: ‘L,’ ‘Y,’ and ‘Arrow’ vertices, however the T-vertex is also useful for determining occlusion where a straight continuous line causes another to terminate (see Figure 5.9).

Vertices can be used to reinforce incomplete contours or to construct non-existent contours. Grossberg et al. [25] demonstrated that illusory contours can be constructed by detecting contour-
ends and linking parallel contour-end detectors. Hubel and Wiesel [10] have found evidence in the visual cortex for contour-end detectors. Grossberg et al. [94] propose that dipole cells link the contour-end responses together forming the illusory contour. In this section we present a technique for identifying vertices also based on the concept of detecting contour-ends but using the Asymmetry edge detector.

5.6.1 New Contour-end Detection Technique

Contour-ends are detected by end-stopped cells in the visual cortex which are tuned to two edges oriented 90° apart in a T formation [10]. Neurophysiological research has found that the vertical length of the detector is tightly tuned like a simple cell while the top of the T is less tightly tuned allowing for corners of a wide range of angles [93]. To achieve a similar result an end-stopped filter has been designed using multiple inputs for the top of the T and one input for the centre (Figure 5.10 (a)). Similar filters to the Asymmetry edge detector are used. A narrower Asymmetry filter is used to allow larger responses close to contour-ends. Also the filter is offset by 3 pixels to achieve the T shape. Human vision uses 18 orientations with 10° between each oriented cell [10]. The Asymmetry edge detector uses 12 orientations (15° apart) which aren’t sensitive to the direction of contour. Contour-end detection on the other hand must describe the angle of the contour-end in 360° requiring 24 orientations. The end-stopped filter uses 14 filters to make up the top of the T and one filter for the centre.

The end-stopped responses are determined by first producing the 24 oriented edge responses for each point using the same Asymmetry edge detector as Equation 4.14 but with modified parameters:

\[ E_A = |C| - t|A| \]  

\[ (5.9) \]
where \( C \) is the response of the Canny edge detector (see Equation 4.9) with \( \theta = n \frac{\pi}{24}, n = (0 \rightarrow 23) \), \( s_y = 1, s_x = 3, \) and \( t_x = 3 \) to translate each filter away from the centre of the contour end, \( A \) is the response from the asymmetry detector with \( s_y = 1.33, s_x = 1 \) to provide a more compact filter, \( t_y = 3 \) to centre the asymmetry detector over the Canny edge detector, and \( t = 2 \) to provide a tight tuning curve.

The end-stopped detector of Figure 5.10 (a) is formed by sampling Asymmetry edge detector responses from selective orientations. The maximum response from the 14 filters at the top of the T is determined and the minimum response between the top of the T and the centre is used as the strength of the end-stopped response:

\[
E_C = \min \left[ E_A(\theta), \max_{i=3}^9 \left[ E_A \left( \theta \pm \frac{i \pi}{12} \right) \right] \right]
\]

(5.10)

where \( E_A \) is the Asymmetry edge detector response from Equation 5.9 and \( \theta \) is the orientation of the vertical bar in Figure 5.10 (a).

The Asymmetry edge detector responses are tightly tuned in orientation and position alleviating false responses which can occur with other contour-end detectors along the length of an edge [12]. The range of angles detectable by the end-stopped detector can be controlled by the number of filters occupying the top of the T. Because the orientation sensitivity is narrowly tuned, edge filters can be used at \( 45^\circ \) from the primary filter allowing acute corners to be detected.

### 5.6.2 New End-stopped Thinning Approach

When thinning end-stopped responses it is important to consider multiple orientations to ensure that contour-ends which will form a single vertex remain together. This is achieved by generating an aggregate image of the component orientations. The aggregate image is blurred using a Gaussian filter with a 2 pixel radius. The Gaussian filter ensures that contour-ends of the same vertex will be thinned to the same point. The contour-ends are thinned by thinning the blurred aggregate. Any points that are set to zero in the aggregate are also set to zero in the multiple orientation responses. Thinning is performed by setting the pixel to zero if it isn’t the largest value in its \( 3 \times 3 \) neighbourhood. Orientation competition is performed to ensure that a contour-end response occurs in only one orientation. A response is set to zero if either of its two adjacent orientations has a larger value.

### 5.6.3 New Vertex Extraction Technique

Vertices are created for points which contain more than one end-stopped response greater than a threshold (set to 16). The angles and strengths for each edge of the vertex are set to the orientations and strengths of the end-stopped responses.

Results for vertex extraction on the Plane image are shown in Figure 5.10 (b). The end-stopped detector allows vertices with acute corners to be detected allowing vertices with many edges to be
easily represented.

The vertex edges are then linked to the extracted contours. For each vertex edge, every contour point is analysed to see whether the vertex edge should be linked to it. It is possible that a vertex exists on the middle of a contour because of limitations with the contour following algorithm, therefore all points must be checked, not just points at the end of contours. The closest contour point in distance and angle is assigned to each vertex edge if it falls within the minimum distance and orientation difference. After the closest point has been determine a check is performed to see whether there is a contour-end which is closer to the vertex by distance but may not have been as close in orientation. If there is, then the point is replaced with the contour-end which is more appropriate to be linked to a vertex edge.

Even though the contour-vertex grouping performed well for simple geometric images more work needs to be done to allow the robust extraction of regions from natural images using the contour-vertex approach. Therefore vertices have not been used as features in the remainder of this research but show promise for higher-level object extraction.

5.7 Contour Matching

The first half of this chapter has dealt with the extraction of contours which is part of the feature extraction phase of content-based retrieval, but before they can be used they must be represented in an efficient form suitable for performing similarity queries. The type of representation used depends on the method of determining the similarity between image contours.

There are not a great deal of techniques that exist for determining the similarity between contours in content-based retrieval systems. Existing systems either compare independent edge points or use sophisticated region-based queries. For example, the ART MUSEUM system [46] uses global and local distributions of edge features for determining contour similarities, however edge distribution is a form of texture representation and does not consider the links between edge points. Likewise, many existing content-based retrieval systems [16, 27, 4] support the querying of texture but do not support the querying of linked edge points in the form of contours. These systems usually support other shape-based query methods based on regions, however the regions are extracted through image segmentation techniques as opposed to being formed from the linking of edge points. In addition, the shape-based query methods are generally designed for the user to specify a subset of objects in the query image that must be found in the database. As a result shape-based queries are generally not designed for whole image comparisons.

Scarloff and Pentland [85] devised a technique called modal matching for comparing two shapes that is invariant to various deformations. Modal matching is a similar approach to affine-invariant Fourier descriptors [84] but both techniques are designed for comparing object silhouettes as opposed to natural images that consist of many objects.
The Hausdorff distance has been used for comparing edge points in images [117]. The Hausdorff distance is used to determine the spatial similarity between two sets of points. Given two sets of points, $A$ and $B$, the Hausdorff distance is defined as:

$$H(A, B) = \max(h(A, B), h(B, A))$$ (5.11)

where

$$h(A, B) = \max_{a \in A} \min_{b \in B} ||a - b||,$$ (5.12)

and $|| \cdot ||$ is some underlying norm on the points of $A$ and $B$ such as the Euclidean distance [117].

Applied globally to an image, the Hausdorff distance ignores any links between edges. However, if applied locally to groups of related edge points the Hausdorff distance can be used for recognising model objects in scenes [117] and for object tracking in video sequences [121]. Since our requirement is to determine overall image similarity the Hausdorff distance would need to be computed between every contour in each image resulting in $NM$ computations per pair of images where $N$ and $M$ are the number of contours in each image. Even though the Hausdorff distance can compare sets of edge points such as those in contours it does not consider the links between edge points or any higher level features of contours such as orientation, length, or curvature.

What is required is a technique similar to the Hausdorff distance that uses contours as the primitives rather than independent points. Before presenting a new technique for matching contours, results from applying the Hausdorff distance to the test image database are presented and used as a benchmark for comparing the new contour matching techniques.

### 5.8 Hausdorff Distance Experiments

By itself the Hausdorff distance does not provide a measure of image similarity but simply the similarity between two contours. Huttenlocher et al. [117] extended the technique to support finding objects in images by translating the model points across the image points. For whole image matching there is no concept of translation to find a match but instead the images are overlaid and the contours compared. Each contour can be considered as a ‘model’ where the closest matching contour is being found in the other image. This is the approach we have taken for computing the Hausdorff distance between two images. The Hausdorff distances between each contour in the query image and its closest matching contour in the database image are summed. This approach is not commutative since images may have different numbers of contours in different spatial arrangements resulting in a different summed distance depending on which image is considered the query image. To make the approach commutative the reverse summed distance is also computed from the database image to the query image and both summed distances are added together. Finally, the result is normalised by the total number of contours in both images:

$$H(A, B) = \frac{\sum_{a \in A} \min(H(a, b) \forall b \in B) + \sum_{b \in B} \min(H(b, a) \forall a \in A)}{N_A + N_B}$$ (5.13)
Figure 5.11: Results of three image queries using the Car, Wedding, and Bush images and the Hausdorff distance between image contours. The query images are displayed in the left column followed by the 10 most similar images.

where $A$ is the query image and $B$ is the database image, $a$ and $b$ are contours in images $A$ and $B$ respectively, and $N_A$ and $N_B$ are the number of contours in images $A$ and $B$ respectively.

The test image database used for evaluating colour histograms in Chapter 3 was also used to evaluate the Hausdorff distance. Contours were extracted from each image using the technique presented in this chapter. The same three test images used in Chapter 3 were used as query images and the first ten images returned in ranking order of closeness of Hausdorff distance were recorded.

5.8.1 Hausdorff Distance Results

The results of the Hausdorff distance experiments are shown in Figure 5.11. The Hausdorff distance performed poorly with the Car image only returning one other car image and it was the last image returned. The other 9 images returned contain a lot of texture and support the fact that the Hausdorff distance only measures a concept of spatial ‘intersection’ as opposed to similarity in contour shape resulting in images with dense contours or texture being considered more similar because there are more points to ‘intersect’ with. It also worth noting that the similarity values returned by the Hausdorff distance measure do not contain a lot of variability for significantly different images. In fact the only variation in the similarities of the three image queries was for the first image returned for the Wedding image. The Hausdorff distance performed better with the Wedding image however only four wedding images were returned (there are enough wedding images in the database to fill the top ten results) and only two were in the top two results. The Hausdorff distance performed well with the Bush image returning bush images in the top 7 results, however, these results could also be explained by the Hausdorff distance’s tendency to measure contour point similarity as opposed to contour shape similarity.
5.8.2 Hausdorff Distance Discussion

The results show that the Hausdorff distance does not perform well for two out of the three test images. The closeness in similarity values of the returned images indicates that the Hausdorff distance has trouble distinguishing the similarity between various images. One of the problems with the Hausdorff distance is that it requires every edge point to be stored in the database. For the test image database it is not unusual for an image to contain 1000 contours each with 5 or more edge points per contour. Assuming two bytes are required to store each edge point and each contour has on average 10 edge points then 20 KB are required to represent an image. The second and most significant problem facing the Hausdorff distance as a contour matching technique is the processing requirements. The Java implementation running on an 800 MHz PC took 15 seconds to compare two images. An image database with 1000 images would take over 4 hours to perform one query.

Based on the storage and computational requirements as well as the poor querying results the Hausdorff distance is not suitable for contour matching. In the next section a new technique for comparing contours is presented that focuses on improving the querying results of the Hausdorff distance by incorporating shape features of contours rather than treating edge points independently.

5.9 New Contour Similarity Technique

The Contour Similarity technique takes the approach of comparing every contour in one image with every contour in another image. It differs from existing techniques such as the Hausdorff distance [117] and comparisons of edge distributions because the links between edges are implicitly used in the contour comparisons. Where the Hausdorff distance only compares spatial location of independent edge points the Contour Similarity technique also compares the orientation, curvature, and length of contours.

Before contours can be compared the location, orientation, curvature, and length of each contour must be determined. As noted above the Hausdorff distance requires every edge point to be stored in the database for comparison however since Contour Similarity operates at the contour level only the extracted features of each contour need to be represented. The following section describes how these features are extracted before describing the comparison algorithm.

5.9.1 Contour Representation

Contours have been represented in the literature through a variety of techniques which have been discussed in Section 2.4.1. These techniques include tangential angles, Fourier descriptors [84], and eigenvectors [85]. Tangential angles represent the change in curvature of uniform distances whilst Fourier descriptors and eigenvectors represent the various spatial frequencies in the varying distance of the shape’s outline from the centroid. Neither technique explicitly represents perceptual features
of contours and therefore comparison techniques can not be designed to use perceptual features. For the Contour Similarity technique we have chosen four perceptual features for describing and comparing contours:

- Centroid position \((x \text{ and } y)\)
- Length
- Prevailing orientation
- Curvature

We call the process of extracting these features *Contour Summarisation* and have found that it reduces the storage requirements of summarised contours to 10% of the raw contour data. The first two features are simple to extract. The centroid position is simply an \(x\) and \(y\) value that represents the mean \(x\) and \(y\) positions of every edge point in the contour. The length is simply the total number of edge points in the contour. The prevailing orientation and curvature are more difficult to extract and are described in the next two subsections.

**Prevailing Orientation** The prevailing orientation is the overall orientation of the contour and is extracted by averaging the orientations of the individual edge points. This process is slightly more difficult than it first appears since edge point orientations only range from \(0^\circ\) to \(180^\circ\). For example, the average of two edge points with orientations \(10^\circ\) and \(170^\circ\) isn’t \(90^\circ\) it is \(0^\circ\). This problem only arises when there are orientations from both \(90^\circ\) quadrants. So the first step is to find the average orientation of each quadrant. If all points lie in only one quadrant then the prevailing orientation of the contour is simply the average of all orientations. However, if there are points from both quadrants the two values need to be combined. If the difference between the averages of the two quadrants is less than or equal to \(90^\circ\) then the two average values can be added proportioned by the number of points that contributed to each quadrant. But if the averages differ by more than \(90^\circ\) then the first quadrant’s average orientation must be shifted by \(180^\circ\) before the two values are combined. This may cause the final prevailing orientation to exceed \(180^\circ\) and will need to be shifted back if necessary. The algorithm for calculating the prevailing orientation is shown in Algorithm 6.

**Curvature** Contour curvature is a description of how much a contour deviates from a straight line. We could calculate how far the contour deviates from a straight line or the area formed between the curve and the straight line but the simplest method is to calculate how much the orientations of the edge points deviate from the prevailing orientation. The average absolute difference between each orientation and the prevailing orientation is calculated for the entire contour. The edge linking algorithm will only link two points if their orientations are within \(\frac{\pi}{12}\) radians therefore the largest curvature occurs when a contour consists of edge points that have orientations that increment
Algorithm 6 Prevailing orientation.

\[ O_1 \leftarrow 0 \{ \text{Orientations of edge points in the first quadrant} \, 0^\circ \to 90^\circ \} \]
\[ O_2 \leftarrow 0 \{ \text{Orientations of edge points in the second quadrant} \, 90^\circ \to 180^\circ \} \]
\[ L_1 \leftarrow 0 \{ \text{Number edge points in the first quadrant} \} \]
\[ L_2 \leftarrow 0 \{ \text{Number edge points in the second quadrant} \} \]

\textbf{for all points in contour do} \hspace{1cm}
\begin{align*}
&\text{if point.orientation } < \frac{\pi}{2} \text{ then} \\
&\hspace{1cm} O_1 \leftarrow O_1 + \text{point.orientation} \\
&\hspace{1cm} L_1 \leftarrow L_1 + 1 \\
&\text{else} \\
&\hspace{1cm} O_2 \leftarrow O_2 + \text{point.orientation} \\
&\hspace{1cm} L_2 \leftarrow L_2 + 1 \\
&\text{end if} \\
\end{align*}
\textbf{end for}

\[ O_1 \leftarrow O_1 / L_1 \]
\[ O_2 \leftarrow O_2 / L_2 \]
\textbf{if} \, L_1 > 0 \, \text{AND} \, L_2 > 0 \, \text{then} \hspace{1cm}
\begin{align*}
&\text{if} \, O_2 - O_1 > \frac{\pi}{2} \text{ then} \\
&\hspace{1cm} O_1 \leftarrow O_1 + \pi \\
&\text{end if} \\
&\text{prevailingOrientation } \leftarrow (O_1 L_1 + O_2 L_2) / (L_1 + L_2) \\
\end{align*}
\textbf{else if} \, L_1 > 0 \, \text{then} \\
prevailingOrientation \leftarrow O_1 \\
\textbf{else if} \, L_2 > 0 \, \text{then} \\
prevailingOrientation \leftarrow O_2 \\
\textbf{end if} \\
\textbf{if} \, \text{prevailingOrientation } \geq \pi \, \text{then} \\
prevailingOrientation \leftarrow \text{prevailingOrientation} - \pi \\
\textbf{end if}
in $\frac{\pi}{12}$ increments. The result is circle or a semicircle. The average orientation deviation along a semicircle from the prevailing orientation is $\frac{\pi}{4}$ which is the largest possible curvature value and is used to normalise curvature values. We have found experimentally that contours with a normalised curvature above 0.25 can be considered curved lines.

### 5.9.2 Contour Similarity Algorithm

The Contour Similarity approach compares every contour in one image with every contour in another image. The basic algorithm is as follows:

1. Each contour in the query image is compared against every contour in a database image to find the contour with closest similarity. The closest similarity values are added to the running total of similarity.
2. Step 1 is run again but in the other direction from database image to query image.
3. The two totals are added together to form the total similarity.
4. The total similarity is normalised by the total number of contour points (not contours) in both query and database images.

The first step requires that a similarity value is computed for each contour pair in the two images. Individual contour similarity is unidirectional from small contour to large contour. Therefore not all contour pairs will be compared in step 1 but will be after step 2 which repeats step 1 in the opposite direction.

Contour similarity is the product of the similarity values computed for the four contour summary features: length, curvature, orientation, and position.

$$C_s = l_s \times c_s \times \theta_s \times p_s$$  \hspace{1cm} (5.14)

The component similarities are described in the following subsections.

**Length Similarity**  The length similarity calculation allows an effectual colinearity grouping to be achieved. As mentioned above similarity is one directional, from the shorter contour to the longer. What we want to allow for is shorter contours to be considered similar to longer contours as opposed to being considered very different. The purpose of this is to allow the similarities of multiple shorter contours that line up against one longer contour (Figure 5.12 (a)) to be aggregated to effectively give the same result as if the shorter contours had been grouped into one longer contour and the two long contours compared. This is achieved by making the length similarity simply the length of the shorter contour which will be from the query image as the comparison is one direction:

$$l_s = l_Q$$  \hspace{1cm} (5.15)

133
Curvature Similarity  Curvature similarity is the absolute difference between curvatures subtracted from 1:

\[ c_s = 1 - |c_Q - c_D| \]  \hspace{1cm} (5.16)

Orientation Similarity  The orientation similarity is calculated by first computing the orientation distance. The orientation distance is the absolute difference between the two prevailing orientations of the contours:

\[ \theta_d = |\theta_Q - \theta_D| \]  \hspace{1cm} (5.17)

The orientation distance may be larger than \( \pi/2 \) which is not possible with a circular range of \( \pi \), so if it is larger then it is subtracted from \( \pi \). The orientation similarity is calculated by normalising the resulting difference by \( \pi/2 \) and subtracting from 1:

\[ \theta_s = 1 - \frac{\theta_d}{\pi/2} \]  \hspace{1cm} (5.18)

Position Similarity  It would be easy to think that the position similarity is simply the Euclidean distance between the two contour centroids. However, the position similarity is the most difficult to compute as it must not interfere with the colinearity grouping effect that allows smaller contours to make up a larger contour. For example, in Figure 5.12 (a) contours A, B, and C are colinear with contour D, however the Euclidean distance from their centroids indicates that A and C are less similar than B, when in fact they all make up an equal contribution to the similarity to D based on the colinearity grouping effect. Therefore the Euclidean distance of the centroids can not be used to find the similarity between colinear contours. Instead, one aspect that remains constant is the perpendicular distance between the contours.

Since only the contour summaries are stored, the perpendicular distance must be computed from the centroid, prevailing orientation, and length of each contour. Firstly, two line segments

Figure 5.12: (a) Contours A, B, and C from the query image are colinear with contour D from the database image. (b) Line segments reconstructed from only the contour summarisation information. (c) Line segments rotated so that the longer line segment is parallel with the x axis.
must be constructed that are centred at the centroid of the contour and extend half the length from the centroid in opposite directions with the prevailing orientation of the contour (Figure 5.12 (b)). The following equations compute both points of each line segment from the query \((Q)\) and database \((D)\) images:

\[
\begin{align*}
x_{Q1} &= x_Q - \frac{l_Q}{2} \cos \theta_Q \\
y_{Q1} &= y_Q - \frac{l_Q}{2} \sin \theta_Q \\
x_{Q2} &= x_Q + \frac{l_Q}{2} \cos \theta_Q \\
y_{Q2} &= y_Q + \frac{l_Q}{2} \sin \theta_Q \\
x_{D1} &= x_D - \frac{l_D}{2} \cos \theta_D \\
y_{D1} &= y_D - \frac{l_D}{2} \sin \theta_D \\
x_{D2} &= x_D + \frac{l_D}{2} \cos \theta_D \\
y_{D2} &= y_D + \frac{l_D}{2} \sin \theta_D
\end{align*}
\]

The next step is to rotate both line segments around the centroid of the longer contour \((D)\) as in Figure 5.12 (c). This is achieved by first shifting all points so that they are relative to the longer contour’s centroid:

\[
\begin{align*}
x_{Q1} &= x_{Q1} - x_D \\
y_{Q1} &= y_{Q1} - y_D \\
x_{Q2} &= x_{Q2} - x_D \\
y_{Q2} &= y_{Q2} - y_D \\
x_{D1} &= x_{D1} - x_D \\
y_{D1} &= y_{D1} - y_D \\
x_{D2} &= x_{D2} - x_D \\
y_{D2} &= y_{D2} - y_D
\end{align*}
\]

Next all four points must be rotated by the negative prevailing orientation of the longer contour:

\[
\begin{align*}
x_{Q1} &= x_{Q1} \cos -\theta_D - y_{Q1} \sin -\theta_D \\
y_{Q1} &= x_{Q1} \sin -\theta_D + y_{Q1} \cos -\theta_D \\
x_{Q2} &= x_{Q2} \cos -\theta_D - y_{Q2} \sin -\theta_D \\
y_{Q2} &= x_{Q2} \sin -\theta_D + y_{Q2} \cos -\theta_D \\
x_{D1} &= x_{D1} \cos -\theta_D - y_{D1} \sin -\theta_D \\
y_{D1} &= x_{D1} \sin -\theta_D + y_{D1} \cos -\theta_D
\end{align*}
\]
The calculations for the longer contour can be greatly simplified. Instead of computing its rotated line segment and then inverse rotating back to the x axis, the longer contour can simply be reconstructed oriented along the x axis:

\[
x_{D1} = -\frac{D}{2} \\
y_{D1} = 0 \\
x_{D2} = \frac{D}{2} \\
y_{D2} = 0
\]

Figure 5.12 (c) shows the y distance between the two line segments. However, there is no guarantee that line segment A will be parallel to line segment D therefore the y distance is taken as the midpoint between the y positions of each end of line segment A:

\[
\Delta y = \frac{y_{Q1} + y_{Q2}}{2}
\] (5.19)

As was noted before the x distance can not simply be the distance between the two line segment centroids as that does not take into account overlap. For example, if line segment D was half the length than it is then there would be no difference in the distance between centroids yet there is a great difference in overlap. Therefore, the best way to determine the horizontal distance of line segment A from D is to measure the amount of extension from the end of D. The following equations measure the extensions on both sides of line segment D:

\[
e_{\text{low}} = \max(x_{Q1}, x_{Q2}) - \max(x_{D1}, x_{D2}) \\
e_{\text{high}} = \min(x_{D1}, x_{D2}) - \min(x_{Q1}, x_{Q2})
\]

Since we know that A is shorter than D, A can only extend along one side of D. The side of the extension is the largest of \(e_{\text{low}}\) and \(e_{\text{high}}\) and becomes the horizontal distance, \(\Delta x\):

\[
\Delta x = \max(e_{\text{low}}, e_{\text{high}})
\] (5.20)

If A does not extend over the end of D on either side \(\Delta x\) will be negative. In this case \(\Delta x\) should be set to zero. The overall distance is the Euclidean combination of \(\Delta x\) and \(\Delta y\):

\[
p_d = \sqrt{\Delta x^2 + \Delta y^2}
\] (5.21)

The position distance must be normalised to a value between zero and one which is done using the diagonal image size. Since the diagonal image size is large compared to many contour
extension lengths a linear normalisation would not allow small differences between contours to be easily distinguished so a two step non-linear normalisation is performed consisting of two linear normalisations. The first range of values maps to the 0 → 0.5 domain whilst the second range of values maps to the 0.5 → 1 domain. The first range of values is from 0 to half the length of the longer contour and the second range of values is from half the length of the longer contour to infinity. The first range normalisation is represented by the following equation:

\[ p_d = 0.5 \frac{2p_d}{l_D} \]  

(5.22)

Whilst the second range normalisation is represented by the following equation:

\[ p_d = 0.5 + 0.5 \frac{p_d - \frac{l_d}{2}}{\sqrt{W^2 + H^2}} \]  

(5.23)

Where \( W \) and \( H \) represent the image width and height. The result of the normalisation is that contours that are not too similar positionally can have a similarity measure that is not too impacted by the lack of positional similarity, providing a mild positional independence for contours that are not close together. The final positional similarity is:

\[ p_s = 1 - p_d \]  

(5.24)

5.9.3 Contour Similarity Experiments and Results

The contour similarity experiments used the same database and query images as the Hausdorff distance experiments. The same three search images of Car, Wedding, and Bush were also used.

The results for the Contour Similarity experiments are shown in Figure 5.13 with the label ‘Contour Similarity’. For the Car query image the contour similarity approach successfully returned both car images as the top two results which is clearly better than the Hausdorff distance. Of note is that the Contour Similarity approach reverses the order of the Car images compared with the colour histogram results in Figure 3.4 which could be explained by the emphasis on contours rather than colour. Also of note is the similarity value 59 given to both Car images which indicates a lack of ability in distinguishing the differences between images. However, there is a much greater range in similarity values for the Contour Similarity approach compared to the Hausdorff distance.

The Contour Similarity approach also performed well with the Wedding image as it successfully returned all wedding photos. The Contour Similarity approach also performed well against the colour histogram results of Chapter 3 successfully returning all wedding photos that contain people in them whereas some colour histogram results contained a wedding photo which does not contain people (for example, \( HSV(18, 2, 2)F \) and \( HSV(6, 3, 3) \) in Figure 3.5).

The Contour Similarity approach performed poorly with the Bush image, which could only be explained by its relatively strong weighting on position, even though the position similarity is normalised to reduce this effect. The most similar photo to the Bush image was successfully returned
by the Hausdorff distance in the second position, however the Contour Similarity approach does not return this image at all in the top ten. It should be noted that this test image contains a lot of texture information and indicates that the Contour Similarity approach is not as well suited for such queries.

5.9.4 Contour Similarity Discussion

The results show that the Contour Similarity approach performs quite well for comparing image contours and performs significantly better than the Hausdorff distance and also compares well with the colour histogram approaches except for images that are characterised primarily by texture. However, like the Hausdorff distance, one of the major problems of the Contour Similarity approach is its representation and querying overheads. It is not unlikely for 1000 contours to be extracted from an image. Assuming five bytes are required to store the summarised features for each contour then 5 KB are required for each image. This is four times less than the 20 KB required for the Hausdorff distance but is still too large for a content-based retrieval system.

In terms of computational requirements, the Contour Similarity technique requires every contour in each image to be compared. With 1000 contours in each image, 1,000,000 comparisons would be required. The current Java implementation uses 89 arithmetic operations, 9 conditions, and 4 trigonometric functions for each individual contour similarity calculation and takes approximately half a second to compare two images. If a database contained only 1000 images then it would take 8 minutes to compute a query. This is significantly better than the 4 hours required by the Hausdorff distance but is still too slow for a content-based retrieval system.

The computational impact of the contour similarity approach can be reduced by caching similarity results. A half matrix of contour similarities can be stored allowing a near instantaneous look up. A 1000 image database would require 500,000 similarities to be stored, which is only 10% of the storage requirements of the raw summarised contour data. However, the storage requirements increase by $n^2$ as the number of images increases. A 10,000 image database requires 50,000,000 similarities to be stored which is five times more than the summarised contour data.

Like the Hausdorff distance, the computational and storage requirements of the Contour Similarity approach make it unsuitable for today’s content-based retrieval systems. Two areas where the approach can be improved is in providing a more compact representation and also more efficient image comparison. In the next section a new technique for representing and comparing contours based on fuzzy histograms is presented and compared with the Hausdorff distance and Contour Similarity approaches.
Figure 5.13: Contour histogram and contour similarity results for the Car, Wedding, and Bush images. Histogram results indicate the number of bins for each dimension (orientation, length, curvature, x, and y) and whether a fuzzy histogram was used.
5.10 New Contour Histogram Approach

The primary problem of the Contour Similarity technique presented in the previous section is that it uses a variable length representation. As a result the data can not be efficiently indexed using a fixed-sized feature vector. In this section we present a novel technique of representing contours in a fixed-sized feature vector using the fuzzy histograms introduced in Chapter 3.

It is not uncommon for edge and texture distribution to be represented in content-based retrieval systems [16, 4, 46, 27] but the representation of contour distribution is rarely used. The difference between contour distribution and edge and texture distribution is that contour distribution is a representation of a higher-level feature.

A histogram consists of axes and each axis represents one feature. Section 5.9.1 identified the four features of contours being the $x$ and $y$ centroid positions, length, prevailing orientation, and curvature. Each feature becomes an axis in a five-dimensional histogram (the centroid is actually two features, $x$ and $y$). Each axis must be quantised into a number of bins to reflect the distribution of features across each axis. The number of bins can affect the representation overhead as well as the querying overhead. Selecting 5 bins per axis for a 5 dimensional histogram would result in a total of 3,125 bins which would require approximately 3 KB of storage space, almost as much as the raw summarised contours, and would require 3,125 comparisons for every image. Therefore the number of bins and range of each bin must be carefully determined so that the total number of bins is minimised without adversely affecting the matching results.

One of the problems with using a very small number of bins per axis is that histogram matching techniques do not consider adjacent bins. So for example, if the length axis was divided into two bins representing short and long contours and two contours fell just either side of the dividing value between a long and short contour then the histogram matching technique would determine that the two contours were completely different based on length. This problem has been addressed in Chapter 3 through the novel approach of fuzzy histograms where instead of each bin being incremented by one, an amount is added to adjacent bins proportional to the closeness of the value to the centre of each bin.

Another problem facing contour distribution representation is that images generally consist of many smaller contours and fewer long contours. A histogram matching technique would therefore place more importance on the smaller contours and the longer contours would have little impact on the matching results. However, this is the reverse of how human perception works, where longer contours are given greater significance than shorter contours which generally represent texture rather than shape. The solution to this problem is to increment each bin by the number of pixels in the contour rather than by one for each contour. The result is that longer contours have equal weighting with the shorter contours. Finally, the size of the source images may be different so the contour histograms are normalised by the total number of pixels in the image which should be proportional to the number of contours that can be extracted from that sized image.
Table 5.1: Bin parameters.

<table>
<thead>
<tr>
<th>Axis</th>
<th>min0</th>
<th>centre0</th>
<th>max0</th>
<th>min1</th>
<th>centre1</th>
<th>max1</th>
</tr>
</thead>
<tbody>
<tr>
<td>orientation (i = 0, \pi/4, \pi/2, 3\pi/4)</td>
<td>i - \pi/8</td>
<td>i + \pi/8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>length</td>
<td>0</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>\infty</td>
<td></td>
</tr>
<tr>
<td>curvature</td>
<td>0</td>
<td>0.1</td>
<td>0.25</td>
<td>0.25</td>
<td>0.4</td>
<td>\infty</td>
</tr>
<tr>
<td>x</td>
<td>0</td>
<td>0.25</td>
<td>0.5</td>
<td>0.5</td>
<td>0.75</td>
<td>1</td>
</tr>
<tr>
<td>y</td>
<td>0</td>
<td>0.25</td>
<td>0.5</td>
<td>0.5</td>
<td>0.75</td>
<td>1</td>
</tr>
</tbody>
</table>

5.10.1 Contour Histogram Experiments

There were two purposes of the contour histogram experiments. The first was to determine the ideal contour histogram construction by evaluating fuzzy and non-fuzzy histograms with different numbers of bins. The second purpose was to determine how well the contour histogram representation performed against the Hausdorff distance and Contour Similarity approaches and whether it would be suitable for use in a content-based retrieval system.

Since the total number of bins can increase rapidly with five axes the number of bins per axis was kept as low as possible. For the orientation axis 4 and 8 bins were evaluated, with 4 bins providing an orientation granularity of 45° and 8 bins providing an orientation granularity of 22.5°. For the length axis granularities of 2 and 4 bins were evaluated. The remaining axes were limited to two bins each. Two bins were sufficient for the centroid position axes as it allows for positions in the four quadrants of an image to be represented. We also evaluated the performance of a contour histogram that did not represent the position of contours at all, thereby making it a translation invariant representation. By removing both position axes the size of the histogram can be reduced by a factor of 4. The curvature axis used two bins allowing contours to be classified as straight or curved. The bins’ ranges and centres (used for the fuzzy histograms) are shown in Table 5.1.

The image database used in the Hausdorff distance and Contour Similarity experiments was used to evaluate the contour histogram performance. Like the colour histogram experiments of Chapter 3 the histogram intersection method was used to compare histograms. The three search images of Car, Wedding, and Bush were also used.

5.10.2 Contour Histogram Results

The results for the various contour histogram experiments are shown in Figure 5.13. For the Car query, the other two car pictures were only returned as the top two results for histograms that included axes that represent contour location showing that location is an important feature for performing contour queries. However, the fuzzy histogram \(4,2,2F\) performed relatively well with no location information returning the two car images in the top three results. No improvement was gained in increasing the number of bins in the orientation and length axes from \(4,2,2,2,2\) to
There was little difference between the contour histograms and the Contour Similarity approach except that the order of the car images for the contour histogram approach is the same as the colour histogram approaches presented in Chapter 3. Also the contour histograms provided a greater dynamic range of similarity measures with values ranging from 67 to 83 for the top two car images as opposed to 59 being given to both car images by the Contour Similarity technique.

The Wedding query performed poorly when the contour location axes were used, with two non-wedding photos entering the top ten results indicating that sometimes contour location is important for contour similarity whilst other times it is not. Interestingly these problems were fixed when a fuzzy histogram was used. Once again there was no benefit in increasing the number of orientation or length bins.

The Bush query was difficult to evaluate as many of the images returned contain some ‘bush’ in them. The (4,2,2F) fuzzy histogram performed better than the (4,2,2) non-fuzzy histogram with the third image being more representative of bush. Including the centroid (4,2,2,2,2) didn’t seem to improve the results however applying the fuzzy histogram improved the fifth result image which contains more bush than the non-fuzzy histogram. Increasing the number of bins in the orientation and length axes returned the same images but two of them were arguably in better positions. Compared with the Contour Similarity approach all contour histograms performed better returning the bush image compared with the car images returned by the Contour Similarity approach. However, the contour histograms did not perform as well as the Hausdorff distance which was able to return the bush image at position 2 rather than 4.

5.10.3 Contour Histogram Discussion

The contour histogram experiments indicate that incorporating the contour centroid can improve results but only if the fuzzy histogram is also used. The benefits shown by using the fuzzy histogram are explained by the low number of bins used to represent each axis which is where the fuzzy histogram technique excels.

Increasing the number of bins used to represent the contour orientation and length did not improve the results significantly. The total number of bins required for the (8,4,2,2,2) histogram is 256 which is considerably large than the 64 bins required for the (4,2,2,2,2) histogram. Being four times smaller, the 64 bin histogram will also require four times less storage and processing requirements.

Compared with the Hausdorff distance the contour histograms performed significantly better except for the Bush image. Compared with the Contour Similarity approach the (4,2,2,2,2F) histogram provided better results, especially with the Bush query image, and significantly lower representation and querying overhead. Assuming one byte is used to represent each histogram bin then only 64 bytes are required to represent the contours in an image using contour histograms compared with the 5 KB for the Contour Similarity method. In addition only 64 comparisons are
required per image compared with 1,000,000. Also since the histogram representation is a fixed sized feature vector it could also be indexed using a multi-dimensional indexing technique such as R-trees [28].

Based on these results the (4,2,2,2,2F) fuzzy histogram combined with histogram intersection provides the most efficient form of representation and querying of contours in a content-based retrieval system for comparing whole images.

### 5.11 Combined Contour and Colour Histograms

In this section we look at combining the contour and colour approaches from this chapter and Chapter 3. The similarities are combined using multiplication:

\[
S = S_{\text{colour}} \times S_{\text{contour}}
\]

As in the other experiments the same image database and query images are used. The smallest colour histogram that gave the best results from Chapter 3 is the HSV (6,3,3F) fuzzy histogram which uses 54 bins. The best contour histogram from this chapter is the (4,2,2,2,2F) fuzzy histogram with 64 bins. However, in Figure 5.14 two smaller histograms are combined, the HSV (3,2,2F) fuzzy colour histogram and the (4,2,2F) fuzzy contour histogram.

The results are as good if not better than the best individual colour and contour histogram results and certainly better than the component histograms that make up the combined result. However, the combined number of bins is only 28 which is less than either the 54 bins of the best colour histogram or 64 bins of the best contour histogram. The result is lower storage requirements, more efficient query computation, and better ordering of results.

The benefits of the combined histograms approach can be attributed to both colour and contour information being used, which human perception also uses, but also to the fuzzy histogram approach.
introduced in Chapter 3. The fuzzy histograms allow for less numbers of bins to be used, down to a minimum of two per axis. As can be seen in these results four axes only need two bins whilst the remaining two axes use three and four bins. Since the number of bins per axis are multiplied together to achieve the total number of bins, a reduction in the number of bins per axis can benefit the storage and computational requirements significantly.

5.12 Conclusion

This chapter has taken the tuned edge results of the last chapter and investigated the best way to extract contours from this edge information and to represent them in a content-based retrieval system. An edge linking scheme has been devised that can take advantage of the multi-oriented edge results of the last chapter and produce better contours than the conventional local processing approach whether combined with the new edge detector or a conventional edge detector such as the Sobel operator. The novel edge linking scheme begins to show the advantages of the single pixel, non-ambiguous, multi-orientation edge detector of the last chapter, and fulfils the goals in extracting the desired contours. The new edge linker takes advantage of multi-oriented edge input by allowing contours of different orientations to cross at the same pixel and also takes into account the relative location of pixels with respect to the orientation of adjacent edge points.

This chapter also addressed the issues of representing contours as they contain variable sized high level information in contrast to the fixed sized feature vectors required by common content-based retrieval systems. Two novel approaches were investigated. One attempted to reduce the high-level information into a fixed size feature vector using fuzzy histograms whilst the other attempted to compare all contours using a brute force method. Both approaches require a summarisation of contour features. We introduced four contour features and techniques for determining them. Only four features are required because the edge linker and edge detector can ensure that contours do not contain sharp angles and therefore can be represented as straight or slightly curved lines.

The contour histogram approach benefits from the fuzzy histograms presented in Chapter 3 allowing small, fast histograms to be constructed that provide good results. The brute force Contour Similarity matching approach is novel in that it allows contours to be compared whilst preserving colinearity grouping effects that occur in human perception. Both techniques performed significantly better than the existing Hausdorff distance however contour histograms were much more suitable for content-based retrieval systems due to the reduced computational and storage requirements.

Finally, the colour and contour histogram approaches were combined allowing even fewer total bins to be used than the best individual colour and contour histograms. The result is an extremely compact feature vector of only 28 histogram bins that performs as well, if not better, than either individual colour or contour histogram of 54 and 64 bins respectively.