Chapter 6

Segmentation of natural images

6.1 Context and problem specification

This chapter examines the problem of segmenting colour images into spatially disjoint regions which are homogeneous with respect to local intensity, colour and textural characteristics. The restriction is made to natural scenes, or real-world images, and hence does not include grey scale images, or artificial palette reduced images. No further assumptions are made about the nature of the data.

The intention is to use techniques successfully applied to the more restricted problems addressed in the previous two chapters. Ideally the segmentation should locate objects with considerations for levels of rotational and scale invariances, occlusion, and the relative spatial position of homogeneous sub-objects.

This chapter, then, examines the decomposition of images into sub-images for a wider class of images.

6.2 Example data and applications

Any real-world scenes are admissible, however most of the illustrations used are taken from a selection of images from the Corel professional photo CD library [267]. The images chosen are also used by researchers from the University of California Berkeley to illustrate content based image retrieval [265] and classification results [263] [264] (outlined in Section 2.7.4). In all, 1000 images are considered from 10 different CDs which are loosely classified into 11 classes based on the name of each CD collection as shown in Table 6.1. The *Bears* collection forms two classes, namely *Polar Bears* (35 images) and *Brown Bears* (65 images). Each image is considered at a resolution of 768×512 or 512×768 pixels and no images are rotated or preprocessed in any way. Some typical examples of each image class are shown in Figure 6.1. All of these are good examples for which a human clearly agrees with the classification. Many ambiguously categorised images also exist.

FIGURE 6.1: Example images from the Corel photo CD collection.

ID	Name of collection	ID	Name of collection
010	Airshows	108	Tigers
028	Fields	134	Cheetahs, Leopards and Jaguars
032	Deserts	135	Bald Eagles
100	Bears	276	Canadian Rockies
107	Elephants	345	Sunsets

TABLE 6.1: The images used from the Corel photo CD collection.

Applications

Broadly speaking this type of image segmentation extracts structure useful for image compression, content based image retrieval (CBIR), scene analysis and robot applications. These are outlined in Section 1.5 with CBIR methods discussed in Section 2.7.

More specifically it serves as a tool for image and object recognition. The ability to distinguish between two or more previously trained image classes provides a powerful system which may be claimed to posses *human intelligence.*

6.3 Local feature extraction

The same method of feature extraction applied to anomaly detection and newspaper segmentation is used. This is fully outlined in Section 4.3. In summary, for coloured images the feature extraction process maps an image to a set of *measurement based* local feature vectors. The dimension of the feature space is governed by both the blocksize $\mathbf{k}$ and the dimension of each window measurement, N.

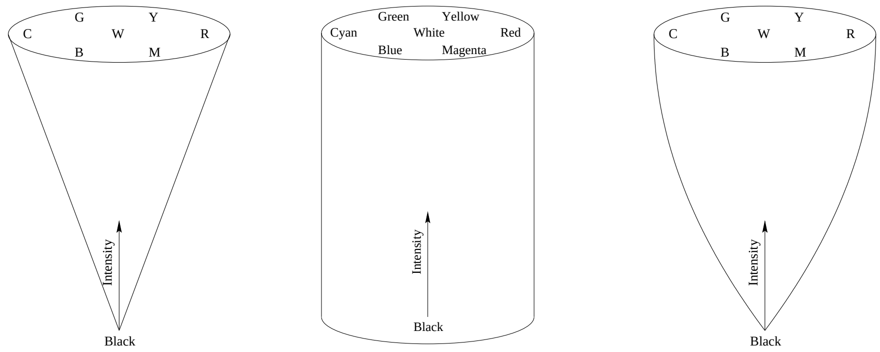
$$\mathbf{I}[\mathbf{r},\mathbf{g},\mathbf{b},\mathbf{x},\mathbf{y}] \quad \longmapsto \quad \mathbf{P}[\mathbf{m}_{1,1},\mathbf{m}_{1,2},\ldots,\mathbf{m}_{1,N},\mathbf{m}_{2,1},\ldots,\mathbf{m}_{\mathbf{k}^2,N},\mathbf{x},\mathbf{y}]$$

For this problem four local window measurements are used, i.e. $N = 4$ and hence $\mathbf{m}_i \in \mathbb{R}^4$ and $\mathbf{P} \in \mathbb{R}^{4\mathbf{k}^2+2}$. These measurements consist of

- average intensity as used for anomaly detection.
- standard deviation of pixel intensity as used for newspaper segmentation.
- average colour consisting of two numbers.

6.3.1 Colour measurements

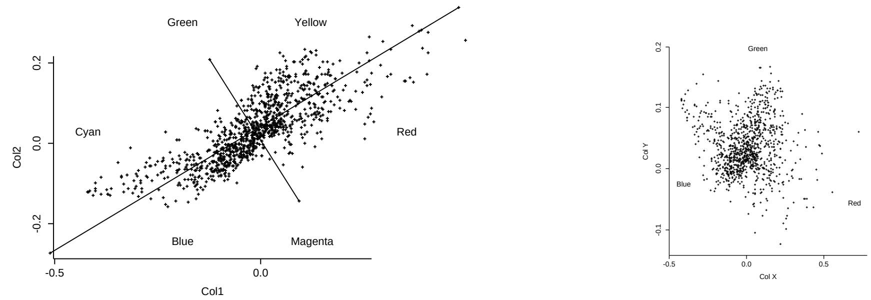
Since RGB colour representation is not well suited to image analysis, other colour spaces such as CIE-XYZ and HSI are typically used. While the CIE-XYZ colour space does provide an equation for *perceptual difference* between colours it requires a white reference and is highly nonlinear ruling out averaging and statistical techniques. The more commonly used HSI cylindrical space suffers from noise sensitivity causing very dark (black looking) colours to be assigned a fully saturated colour. The alternative HSI conical space prevents this by adding a linear dependency between intensity and maximum saturation, hence unfairly distinguishing between the same perceived colour from mid to high range intensities. These and related issues are covered more fully in Section 2.6 where all of the mentioned colour spaces are defined along with related references.



(a) HSI colour cone ($\alpha = 1$) (b) HSI colour cylinder ($\alpha \to \infty$) (c) proposed HSI tradeoff ($\alpha = 2$)

FIGURE 6.2: Proposed tradeoff between the HSI conical and cylindrical spaces.

In order to obtain a balance between the HSI cylindrical and conical spaces a tradeoff is used whereby the saturation is limited by $1-(1-\mathit{intensity})^{\alpha}$ as depicted in Figure 6.2.[1] For $\alpha=1$ this is equivalent to the HSI colour cone in which maximum saturation is directly proportional to intensity. Larger values reduce this dependency, the limiting case corresponding to a cylinder in which intensity and saturation are independent. Experimentation suggests that $\alpha=2$ provides an appropriate tradeoff.



(a) hue-saturation plane (b) adopted colour measurements

FIGURE 6.3: The distribution of the hue-saturation plane for natural images.

The resulting hue-saturation plane is considered in Cartesian coordinates. The plane is usually defined so that red is at 0°, green at 120° and blue at 240°, although this

[1] It should be noted that at higher intensities the saturation is limited by the separate RGB components and a singular extreme point corresponding to the *white* corner of RGB space. This constraint is not shown in Figure 6.2.

choice is arbitrary. Analogous to Ohta's colour features[2] [228] it is desirable to choose the axes in directions which decorrelate the colour measurements. Figure 6.3(a) shows the distribution of colours in the hue-saturation plane as taken from the 1000 outdoor scenes along with the appropriate principal axes. As is clear from this illustration, purely red and purple colours are uncommon in nature, as are lurid greens.

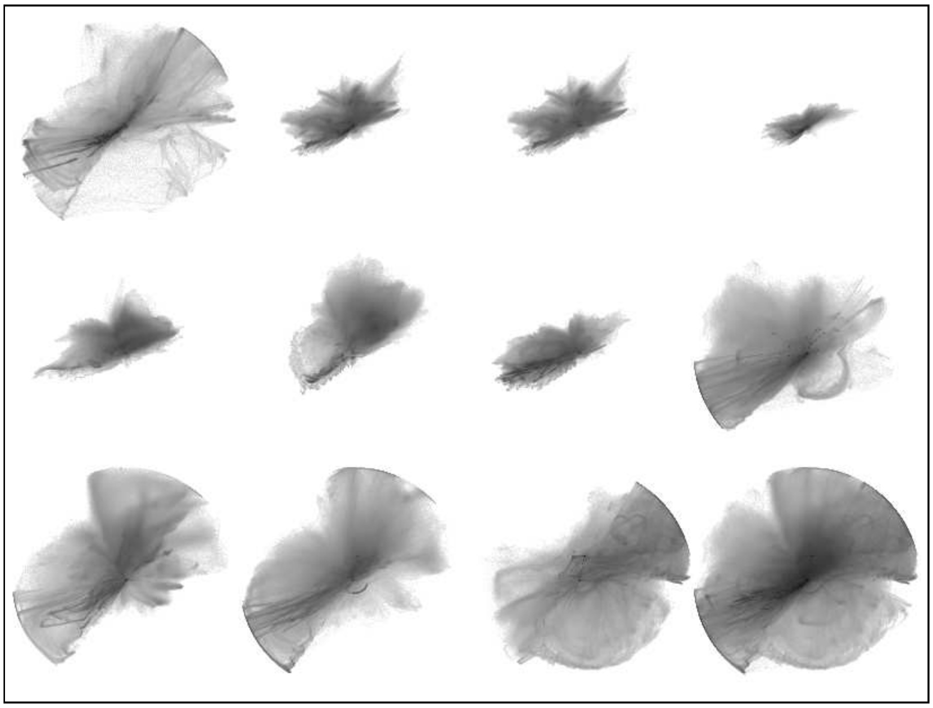


FIGURE 6.4: Histograms of chroma values (hue-saturation plane) for the considered Corel photo CD images. Each histogram corresponds to a single class except the one to the bottom right which uses all images. All are illustrated on a logarithmic scale, with the final histogram calculated from over 390 million pixel values.

Figure 6.4 further illustrates this distribution using numerous histograms. For each different photo CD collection (or class of image) approximately the same correlation is observed, although some images are more heavily saturated than others. In the histogram to the top right a low saturation is observed (grey and white colours) whereas the second last histogram produces abnormal red and purple colours. These correspond to *Polar Bears* and *Sunsets* respectively. Noting that values are shown on a logarithmic scale, the final histogram shows a very strong correlation. The principal axis calculated using all 393.216 million pixel values is at 31.22° to the x-axis. Furthermore the corresponding eigenvalues indicate that the spread, or standard deviation, is approximately 3 times greater along this correlation.

In view of these distributions the adopted colour values are obtained using a tradeoff between the HSI cone and cylinder ($\alpha = 2$) which is represented by co-ordinates along the

[2]Refer to Section 2.6.1 on page 94.

two axes shown in Figure 6.3(a). Furthermore, the measurement along the minor axes is scaled approximately for normalisation. The distribution of these two colour measurements is illustrated in Figure 6.3(b). These final measurements outperform normalised RGB values, such as *triangle colour*, in the segmentation algorithms outlined in [226]. The decorrelation and scaling also significantly improves results.

6.3.2 Colour deviation

The average colour is equivalent to the centroid of the distribution in the modified hue-saturation plane. A general measure of colour deviation is not so obvious. The following possibilities are based on the spread of colours within a window, or the covariance $\mathbf{C}$ of the distribution [226].

- The covariance matrix itself. This suffers from the problem that measurements are of variance (squared quantities) rather than standard deviation.

- The $\mathbf{B}$ matrix such that $\mathbf{B}\mathbf{B}^T = \mathbf{C}$ and $\mathbf{B}$ is symmetric positive semi-definite. This is essentially the particular square root of the covariances which is positive semi-definite. A brief analysis suggests that these three unique values contain unnecessary information.

- The square root of the principal eigenvalue of $\mathbf{C}$, or $\sqrt{\lambda_1}$. This tends to bias different colour distributions based on their *shape*.

- The value $\sqrt{\lambda_1 \lambda_2}$ to give an indication of the area of the distribution. This is also the square root of the determinant of $\mathbf{C}$. Analysis shows that this is a poor measurement. It tends to over-estimate the variation within a colour, such as a section of blue sky which forms a round cluster, and under-estimate linear transitions between colours, such as a sudden transition from blue to red.

- Other eigenvalue based measurements. For example $\sqrt{\lambda_1}\max(\sqrt{\lambda_2}, \frac{1}{4}\lambda_1)$ which may overcome biases incurred by simply using $\sqrt{|\mathbf{C}|} = \sqrt{\lambda_1 \lambda_2}$.

- The standard deviation of the saturation from the modified HSI colour. Although this subjectively produces the best results, it makes no consideration for changes in colour, only saturation. Consequently it unfairly distinguishes between different shades of sky, but fails to identify vibrant colour changes.

After examination of all of the above measurements, the determinant and standard deviation of saturation appear to be superior. However in both cases the important features captured by these measurements are also captured by the standard deviation of intensity which identifies much crisper and more localised features. Segmentation results over a range of images is not improved, and is typically retarded, by the addition of either measurement. Consequently a measure of colour deviation is not used or further examined.

6.3.3 Measurement normalisation

The raw window measurements are in the range $[0, 255]$, $[0, 127.5]^3$ and $[-1, 1]$ for intensity, standard deviation and colour, respectively. It is desirable to scale these so that the standard deviation in each measurement is approximately the same to allow the Euclidean distance to be used between feature vectors. This is equivalent to using a weighted Euclidean distance and further assumes independence between each measurement. An alternative is to use a quadratic distance which considers covariance between measurements, although this requires significantly more computation. Since the covariance between the two adopted colour measurements is zero a quadratic metric does not offer any significant improvement.

Based on the Corel photo CD collections, other natural images and newspaper images, scaling values of 1, 4, 415 and 1225 are used. In other words the intensity measurement remains unchanged, the standard deviation is multiplied by 4 and the colour measurements by 415 for the major axis and 1225 for the minor axis.

6.3.4 Parameter values and implementation summary

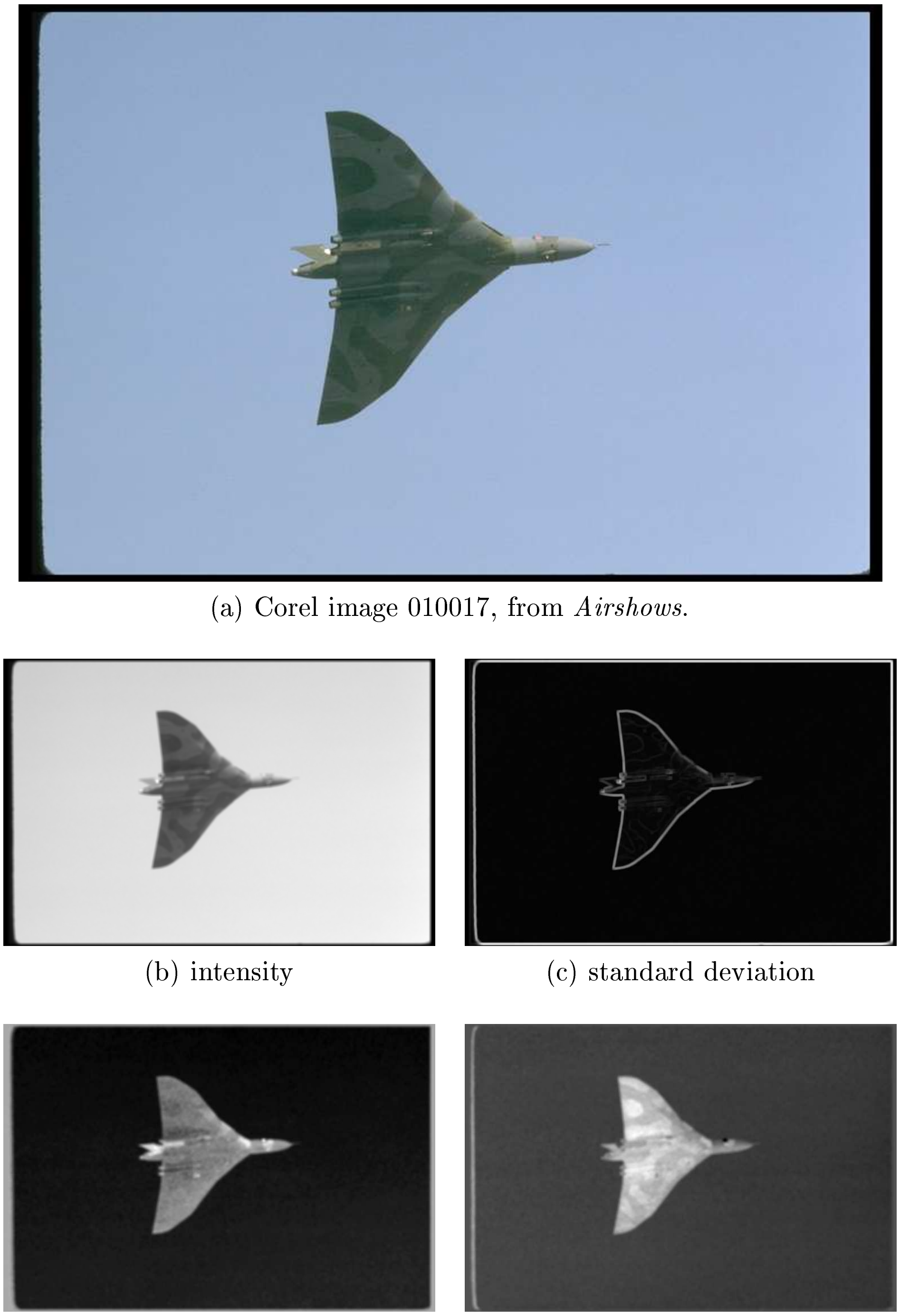
The detail of the Corel images is suited to a windowsize $\mathbf{n}$ of between 4 and 8 pixels. The value $\mathbf{n} = 6$ was chosen and used throughout with a blocksize $\mathbf{k} = 3$, a shiftsize $\mathbf{s} = 6$ and $\delta_x = \delta_y = 1$. The four window measurements used are given by the following equations based on the definition for HSI colour on page 96. The final feature vectors form a set of points in $\mathbb{R}^{38}$.

$$
\begin{aligned}
\mathbf{m}_{average} &= \left(\frac{\sum \mathbf{i}}{\mathbf{n}^2}\right) \\
\mathbf{m}_{deviation} &= 4\left(\sqrt{\frac{(\sum \mathbf{i}^2)}{\mathbf{n}^2} - \left(\frac{\sum \mathbf{i}}{\mathbf{n}^2}\right)^2}\right) \\
\mathbf{m}_{colour1} &= 415\left(\frac{\sum \left(s(1.0 - (1.0 - \mathbf{i})^2)\cos(h - 31.22^\circ)\right)}{\mathbf{n}^2}\right) \\
\mathbf{m}_{colour2} &= 1225\left(\frac{\sum \left(s(1.0 - (1.0 - \mathbf{i})^2)\sin(h - 31.22^\circ)\right)}{\mathbf{n}^2}\right)
\end{aligned}
$$

6.4 The feature space

To illustrate the structure and general properties of the feature space, the relatively simple image shown in Figure 6.5(a) is used. This image consists of two distinct homogeneous regions corresponding to the sky and an aeroplane. Although not immediately obvious the aeroplane is covered by a dull green and grey camouflage and has various markings and light reflections from the sun. The image is surrounded by a black boundary region

[3] This is the extreme case where half the pixels are *white* and half are *black*.



(a) Corel image 010017, from *Airshows*.

(b) intensity

(c) standard deviation

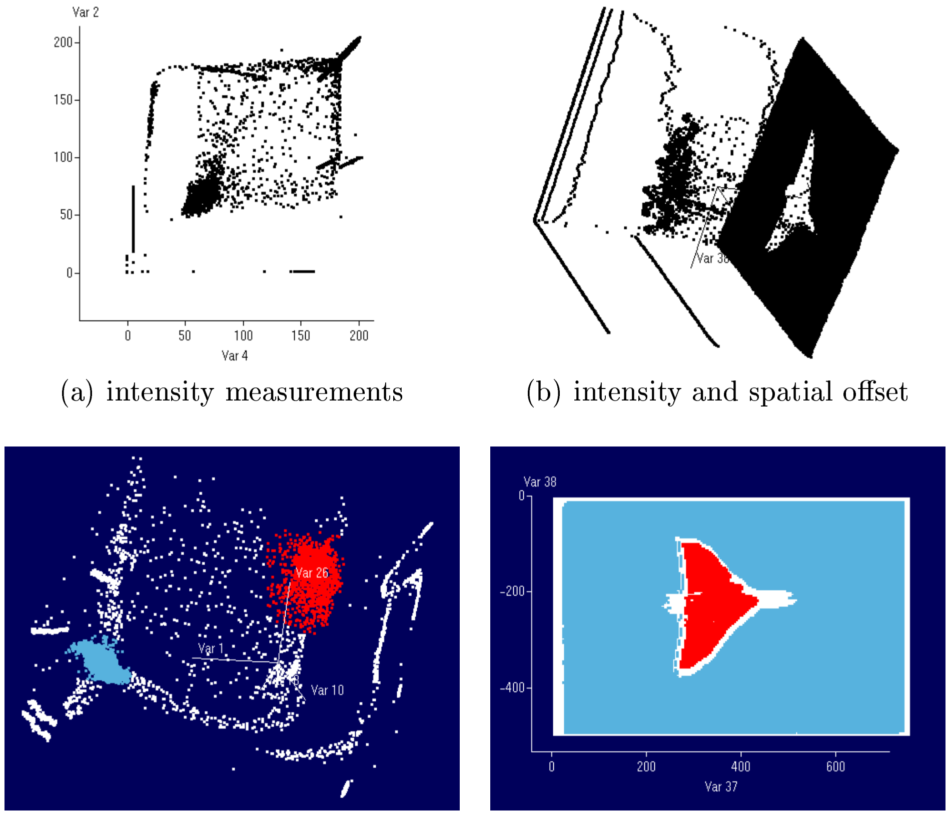
(d) colour (major)

(e) colour (minor)

FIGURE 6.5: An image with well defined regions and its window measurements for $\mathbf{n} = 6$.

close to which the colours of individual pixels are severely distorted.

Each of the four window measurements appear in Figures 6.5(b)-(e). For average pixel intensity the large sky region bears similar features to the painted steel images examined in Chapter 4. The lighting varies diagonally across the image with values deviating slightly from this expected value. The standard deviation measurement highlights the edges of the aeroplane and sky region and also gives significant mid-range values within the aeroplane. The two average colour measurements also clearly distinguish between objects. The first ranges from the blue/cyan (displayed as black) to yellow/red resulting from the decorrelated hue-saturation plane as clear from Figure 6.3(a). The second varies from magenta to green. The values indicate that the aeroplane is less blue and more green than the sky. Note that all measurements are displayed relative to this image by scaling between extreme values.



(a) intensity measurements (b) intensity and spatial offset

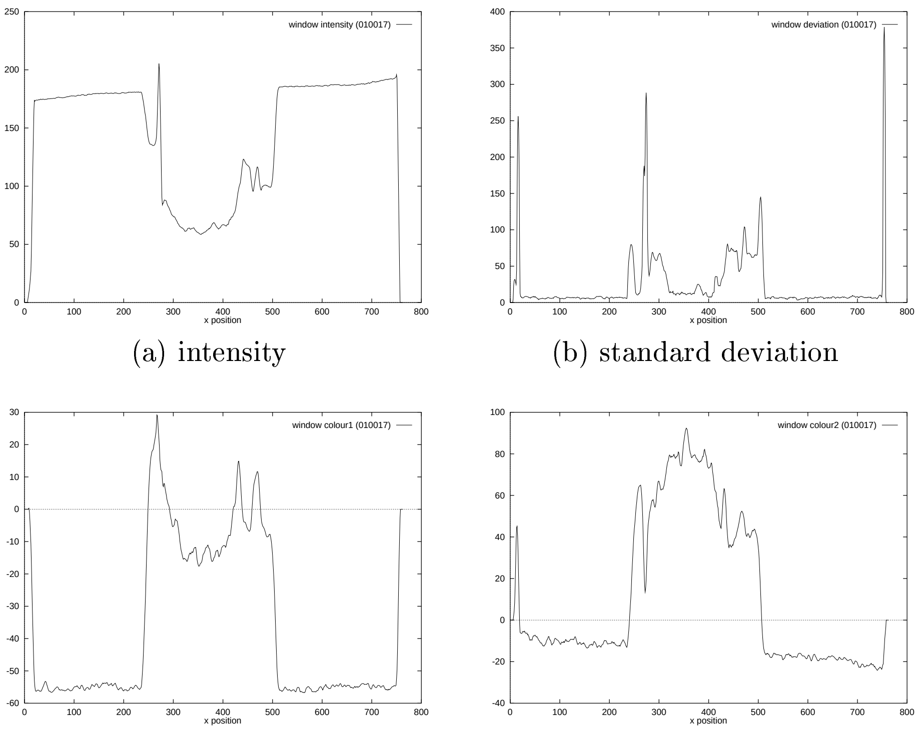
(c) manually coloured projection (d) corresponding spatial offset

FIGURE 6.6: Projections of the feature space.

The projection of the feature space onto the middle top and middle left window measurements for average intensity is shown in Figure 6.6(a). The majority of points form an extremely dense cluster to the top right where the intensity is high and virtually the same in both windows. A second dense cluster at an intensity of about 50 corresponds to the aeroplane. As with the feature spaces from painted steel and newspaper images, the regions which are homogeneous with respect to the window measurement lie close to the *one vector* $\vec{\mathbf{1}}$ and the anomalous or boundary regions form a transition which deviates from this. In this example the isolated points to the far left and bottom result from the black border. Figure 6.6(b) further shows the characteristics of average intensity with respect to spatial position. Each object is clear from this projection from $\mathbb{R}^3$.

Figure 6.6(c) shows a projection onto the four different window measurements. Again two distinct clusters and associated transition regions are apparent. To verify the apparent structure, these *clusters* are manually coloured from the observed projection and re-projected onto the feature vector elements for spatial offset. The $\mathbf{y}$ position (*Var* 38) is negated to account for the different coordinate system used for images.

To more completely outline the structure of the feature space from a practical point of view, a horizontal cross section though the centre of the aeroplane is examined. The average value measurements (Figures 6.7(a),(c) and (d)) maintain a well defined value within each region and an edge like change at region boundaries which is smoothened as an effect of the windowing operation. The region in Figure 6.7(d) corresponding to sky is particularly interesting. The values slowly vary across each region, hence making it possible to infer that they correspond to the same object occluded by the aeroplane. This inference is further strengthened by the other measurements.



(a) intensity

(b) standard deviation

(c) colour (major)

(d) colour (minor)

FIGURE 6.7: Window measurements along a horizontal line through the middle of the aeroplane shown in Figure 6.5 (for $\mathbf{y} = 216$, $\mathbf{n} = 6$).

The standard deviation of pixel intensity maintains a relatively low but constant value within regions and experiences short spikes at region boundaries. For this particular image the standard deviation within regions is very low, but in general regions may be characterised by a high local deviation in which case the histogram peaks at boundaries, but plateau accordingly between them.

In general each homogeneous region is characterised, or well approximated by, a constant value or constant derivative. Each region is bounded by changes in average window measurements which disagree with this homogeneity and furthermore by a ridge of high standard deviation measurements. As with most segmentation problems, the edges in these measurements are not well defined making edge based segmentation susceptible to smooth edges or noise and ineffective over a range of images. Similarly flood fill or region growing algorithms are prone to leak into neighbouring regions due to smooth transitions in local characteristics. However, the feature space possesses additional structure. Region boundaries are further characterised by a large distance $\mathbf{d}$ from $\vec{\mathbf{1}}$. Therefore, given a model which correctly defines the characteristics and variations of a region, the location of boundaries involves determining whether the local characteristics stray from the expected value restricted to a particular relative spatial position. This expectation may be further weighted by the model of the entire region.

6.5 Identification of spatially disjoint homogeneous regions

This section outlines the segmentation algorithm used to locate image regions. In view of the structure observed in the feature space, a region growing algorithm is proposed which stems from techniques presented in [226]. Seed regions are chosen based on a measure of local change, modelled and iteratively grown and remodelled until a stopping condition is met.

6.5.1 Determining local change

First consider the feature extraction process using a single window measurement. In this case, a small region with homogeneous characteristics produces a feature vector $\mathbf{p} \in \mathbf{P}$ which lies close to the one vector $\vec{\mathbf{1}}$, regardless of dimension. Hence, as already discussed in Chapters 4 and 5, the orthogonal distance $\mathbf{d}$ from $\mathbf{p}$ to $\vec{\mathbf{1}}$ provides a sufficient measure of local change, i.e.

$$\mathbf{d} = ||\mathbf{p} - \mathbf{l}\,\vec{\mathbf{1}}|| \qquad \text{where } \mathbf{l} = ||\mathrm{proj}_{\vec{\mathbf{1}}}\mathbf{p}||$$

This simple measurement produces comparable results to modelling the entire feature space with low order geometric moments using only a fraction of the time. Extending this measure for feature vectors with multiple window measurements requires that individual window measurements have comparable magnitudes and standard deviations. Alternatively the local change measure is obtained by appropriately *adding* the value for each measurement. This only requires measurements to be normalised to comparable standard deviations which is done during feature extraction.

For this application the four $\mathbf{d}$ values are simply added together. This is obviously equivalent to a scaled average or the first Minkowsky metric.

6.5.2 Algorithm details

For the feature vectors $\mathbf{p} \in \mathbf{P}$, the algorithm consists of the following steps.

- Obtain an *image* containing the measure of local change for each feature vector given its ($\mathbf{x}$,$\mathbf{y}$) offset. This is referred to as the *local change image* Δ.
- Threshold Δ and identify all sufficiently large disjoint regions which are 4-connected. The initial threshold allows a very low tolerance to local change to identify the most homogeneous areas. Each appropriate region is considered as a seed region.
- Iteratively grow the largest remaining seed regions for this threshold and model the final regions accordingly.
- Label spatial positions for the modelled regions hence excluding these from further modelling. Furthermore re-evaluate the seed regions for this threshold to determine the next largest seed region which is still valid. If a valid seed region exists repeat the previous step.
- Repeat these steps with an increased tolerance to local change.

Seed regions are modelled and grown as follows.

- Calculate an initial region model by taking low order geometric moments. The region is modelled by a mean $\mu \in \mathbb{R}^6$ and symmetric 6 by 6 covariance matrix $\mathbf{C}$ using the 4 window measurements and location (x, y) for all window placements within the region.
- Perform a flood fill algorithm using the Mahalanobis distance with respect to the region's model. Each position which is 4-connected to the current region is examined. If the corresponding vector (the window measurements centred at this point and the spatial offset) is within an appropriate Mahalanobis distance it is added to the region.
- Remodel the region and repeat the flood fill algorithm until it no longer grows. Typically each iteration will form a less dense model thus allowing incremental growth with respect to the current region model.
- Model the final region and store this model for later use. The appropriate representation is discussed in Section 6.6.

The idea behind this method is to locate large homogeneous and spatially disjoint objects, model these and continue to look for smaller ones. The initial tolerance to local change is made extremely low to ensure smooth object boundaries are not crossed. Once an object has been modelled, the tolerance is increased since the object's model prevents boundary crossings. Ideally objects should be separated by *unused* regions corresponding to the edges or areas between spatially disjoint objects.[4] Should objects touch, it may be desirable to compare their characteristics and possibly amalgamate these into one single object. Such amalgamation may be dependent on the particular application. This and other possible amalgamation criteria are addressed in Section 6.7.

6.5.3 Region growing conditions

The above algorithm requires the choice of a *Mahalanobis threshold* as a criteria for region growth. Although a static threshold produces good results, it suffers from two opposing problems.

Firstly seed regions which are almost constant, such as those produced by sky, form extremely dense clusters and even singularities. Even given small perturbations or programming techniques to avoid degeneracy, the Gaussian model is so tight that points which are intuitively *close by* are a large number of standard deviations away. Consequently regions appear to be under-grown. This is akin to the problem of identifying clusters of various sizes as illustrated in Figure 2.8 on page 32. In this case the appropriate scale is based on distances at which values are perceptually indistinguishable.

On the other hand, regions which vary significantly form sparse clusters which spread across relatively large proportions of the convex bounded feature space. This causes points which are intuitively distant to be deemed close enough with respect to the refined model. As a result the algorithm leaks into neighbouring regions. Since the most homogeneous regions are located first this is not as big a problem as it would appear. However in extreme cases the spread within an acceptably defined region is so great that the entire *feature space* is within relatively few standard deviations of the model. Consequently the final located regions are prone to filling all previously unassigned connected image positions.

To cater for this, a simple dynamic Mahalanobis threshold is proposed in which the number of standard deviations is restricted according to the spread of the region model. The threshold below is used, where Σ_{max} is the largest window measurement standard deviation as obtained from the covariance matrix. This inhibition factor prevents regions from leaking into one another along small fingers of gradual local change.

$$\text{threshold} = \max\{5.5 - 0.1\Sigma_{max}, 0\} \qquad \text{where } \Sigma_{max} = \max\{\sqrt{\mathbf{C}_{ii}} \;\; \forall\, i \in \{1..4\}\}$$

[4] This is assuming the feature extraction process makes use of neighbourhood measurements.

6.5.4 Implementation details

The algorithm is implemented in C, transforming an image of any standard format into a set of identified regions. All algorithm parameters are obtained from *UNIX environment variables* prior to program execution. The adopted parameters values are outlined with the implementation below. The routines used for anomaly detection and newspaper segmentation are reused along with some general image processing routines which form a custom image processing library.[5] References to these functions are outlined throughout, although the syntax is simplified in places for ease of reading. This program is named "**upwrite**".

1. Load the image. Image formats and *gzip* compression are automatically handled using *gunzip* and *convert* if the image format is unrecognised.

$$\mathbf{image} = \mathit{LoadImage}()$$

2. Obtain the feature vectors using the method and parameters described in Section 6.3.

$$\mathbf{fv} = \mathit{GetScaledFeatureVectors}(\mathbf{image}, \mathbf{n} = 6, \mathbf{k} = 3, \mathbf{s} = 6, \alpha = 2, \theta = 0.545216)$$

   This routine is specific to **upwrite** and consists of the following more general routines and steps.

$$\begin{aligned}
\mathbf{intensity} &= \mathit{CopyImage}(\mathbf{image}, \mathit{RGB2BYT}) \\
\mathbf{average} &= \mathit{GetAverageWindowImage}(\mathbf{intensity}, \mathbf{n}, \mathbf{k}, \mathbf{s}) \\
\mathbf{sd} &= \mathit{GetSDWindowImage}(\mathbf{intensity}, \mathbf{n}, \mathbf{k}, \mathbf{s}) \\
\mathbf{tmp} &= \mathit{GetModifiedHSPlane}(\mathbf{image}) \\
\mathbf{hs1}, \mathbf{hs2} &= \mathit{SplitXYImage}(\mathbf{tmp}) \\
\mathbf{c1} &= \mathit{GetAverageWindowImage}(\mathbf{hs1}, \mathbf{n}, \mathbf{k}, \mathbf{s}) \\
\mathbf{c2} &= \mathit{GetAverageWindowImage}(\mathbf{hs2}, \mathbf{n}, \mathbf{k}, \mathbf{s}) \\
\mathbf{scaled_sd} &= \mathit{MultImageByConstant}(\mathbf{sd}, 4) \\
\mathbf{scaled_c1} &= \mathit{MultImageByConstant}(\mathbf{c1}, 415) \\
\mathbf{scaled_c2} &= \mathit{MultImageByConstant}(\mathbf{c1}, 1225) \\
&\mathit{return}(\mathbf{average}, \mathbf{scaled_sd}, \mathbf{scaled_c1}, \mathbf{scaled_c2})
\end{aligned}$$

3. Calculate the local change image Δ. At each location the value is simply the sum of the distances from $\vec{\mathbf{1}}$ for each window measurement, i.e. $(\mathbf{d}_{average} + \mathbf{d}_{scaled\_sd} +$

[5] Refer to "http://ciips.ee.uwa.edu.au/Papers/PhD_Theses/1999/01/" for the code to this exploratory image analysis library and other associated information.

$\mathbf{d}_{scaled\_c1} + \mathbf{d}_{scaled\_c2}$). This routine is specific to **upwrite**.

$$\Delta = GetLocalChange(\mathbf{fv})$$

4. Establish an image to store the region label. The implementation further assumes that an image contains at most 255 disjoint regions. If this is exceeded an error is reported and the program fails. For the Corel images 20 to 40 regions are typically produced by this algorithm. Over all 1000 images the average is 29.6, the standard deviation 14.85, the median 29, the maximum 121 and the second highest 86. The image allocation also ensures all *pixels* are initialised to 0.

$$\mathbf{classed} = AllocImage(\mathbf{X}, \mathbf{Y}, BYT)$$

5. Locate and model all seed regions considering a range of local change thresholds **lc_thres**. The threshold is initialised such that $\sqrt{\mathbf{lc\_thres}} = \delta = 5$, with the square root incremented by 2 each iteration until it exceeds 80. There is no justification behind this choice and these values are affected by the number of window measurements and the blocksize **k** (see Section 5.4.4). However, as long as the value starts sufficiently low and does not increase too rapidly the choice is somewhat arbitrary. Obviously the fewer iterations the quicker the program. For each value the following are performed.

   - Threshold the local change image, setting all values below **lc_thres** to 0.

   $$\Delta_{thres} = ThresholdImage(\Delta, \mathbf{lc\_thres})$$

   - Clear the regions of Δ_{thres} which have already been assigned to a region. These are defined as the points of **classed** which are non-zero.

   - Model the largest region. This routine is specific to **upwrite** and further outlined below. The parameter **smallest_seed** $= 400$ is used.

   $$\mathbf{region} = ModelLargestBlob(\mathbf{fv}, \mathbf{classed}, \Delta_{thres}, \mathbf{smallest\_seed})$$

   - Update **classed** by labelling the position defined by **region** with the next available label.

   - If **classed** has changed save it to provide the segmentation at this particular of tolerance to local change. For implementation the image is coloured according to the routine *get_RGB_colour* and filtered, or piped, through the external program *ppmtogif*.

$$\begin{aligned}\mathbf{tmp} &= \mathit{ConvertClassedToRGBImage}(\mathbf{classed}, get_R GB_c olour()) \\ &\quad \mathit{IMAGEprintf}(ppmtogif, \mathbf{tmp}, 0);\end{aligned}$$

6. Model each final region with a Gaussian.

$$\mathbf{gaus\_list} = \mathit{GetGaussianModelFromClassedImages}(\mathbf{fv}, \mathbf{k} = 1, 0, \mathbf{classed}, \lambda_{min} = 0.01)$$

   It is desirable to model only the single window measurements and the spatial position rather than the entire corresponding feature vector (in $\mathbb{R}^{38}$). Hence, the general routine is called with a parameter ($\mathbf{k} = 1$) to override the blocksize maintained in the array of symbolically stored feature vector images **fv**. The third parameter indicates that no dimensions should be ignored, as opposed to using 2 to omit the spatial offset. The final parameter outlines the minimum eigenvalue to be assigned to avoid degeneracy. The routine returns a linked list of structures. Each structure contains information including the number of data points, mean, covariance matrix, eigenbasis, inverse covariance matrix, sum of vector elements and sum of vector element products.

7. Save the regions to *disk*. Rather than storing the low order geometric moments, the running sums are stored. These provide the statistics necessary to calculate these moments, but also to support combining the statistics from two or more regions. In addition to these characteristic values the lossless shape of each region is stored. These representations are outlined in the next section. The implementation writes the region representation to standard output thus allowing output to be redirected to a file or *piped* into another process such as a DownWrite or classification program.

The routine *ModelLargestBlob* consists of the following steps.

1. Locate the largest seed region. The series of routines outlined below are used. The routines *CompressClassedImage* and *ConvertToByteImage* ensure that the region is at least **smallest_seed** pixels and also support a **classed** image with a maximum of either 256 or 2^{32} classes. The *BlobifyImage* routine is an advanced connected component labelling routine with numerous optimisations [304] [305] combining techniques used in several standard algorithms [20, p28–48].

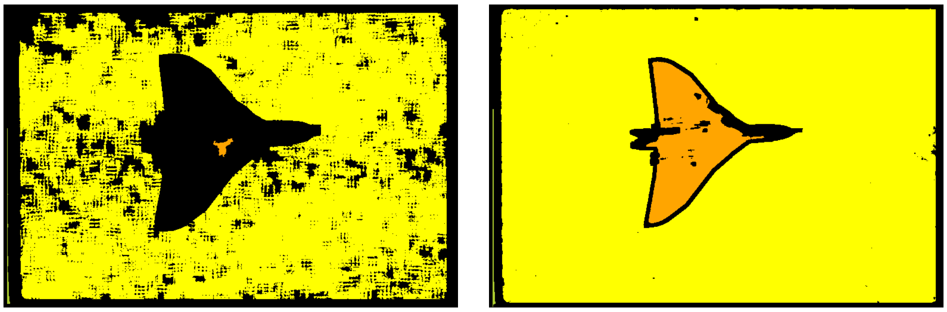
$$\begin{aligned}\mathbf{tmp} &= \mathit{InvertImage}(\Delta_{thres}) \\ \mathbf{labelled} &= \mathit{BlobifyImage}(\mathbf{tmp}) \\ \mathbf{tmp} &= \mathit{CompressClassedImage}(\mathbf{labelled}, \mathbf{smallest\_seed}) \\ \mathbf{tmp} &= \mathit{ConvertToByteImage}(\mathbf{tmp}) \\ \mathbf{region} &= \mathit{GetLargestNonZeroClass}(\mathbf{tmp})\end{aligned}$$

2. Fit an initial Gaussian model using the implementation outlined in Section 4.5.3. The scaled feature vectors **fv** within the region defined by **largest** are used (modified with $\mathbf{k} = 1$). As well as calculating the mean μ and inverse covariance matrix $\mathbf{C}^{-1}$ necessary to evaluate the Mahalanobis distance, the running sums are maintained.

3. Perform flood fill. For implementation a *stack* is maintained containing all of the positions within the region. Each position on the stack is examined to determine if any adjacent unlabelled positions agree with the model with respect to the dynamic Mahalanobis threshold. Those which agree with the model are set in **region** and added to the stack for further expansion. Also the feature vector values are appropriately added to the running sums. Those which do not agree are tagged to avoid further examination. Once the stack is empty the model is recalculated from the maintained running sums and if the number of data points in the model has changed the all positions are untagged and the flood fill process is repeated. On completion the updated **region** is returned.

6.5.5 Segmentation results

The number and size of regions formed by the segmentation algorithm varies considerably across the Corel images which differ in complexity, number of objects, scale and structure. It should be noted that with respect to the general approach it is more desirable to locate a number of smaller regions rather than incorrectly combine inappropriate ones. The justification for this is that regions may be amalgamated at a higher level of processing whereas a separation process is not within the framework of the UpWrite.



(a) seed regions (b) final regions

FIGURE 6.8: The location of homogeneous regions for the aeroplane image (010017).

Figure 6.8[6] shows the located homogeneous regions of the aeroplane image. The sky region is located at $\delta = 5$, the aeroplane when $\delta = 15$ and the black border section at $\delta = 75$. The initial seed regions outline the highly constant characteristics of the sky as opposed to the aeroplane. The seed region for the aeroplane is more typical. In this

[6]The colours used throughout are automatically chosen in sequence by the order of located regions. A total of 39 unique colours are cycled through beginning with *yellow*, *orange*, *skyblue* and *slateblue* [63].

example 4, 14 and 5 remodelling iterations are required for the three regions respectively. The sky benefits from the massive seed region and the black border from the constant values. The black region is extremely thin due to the window effects as clear from earlier illustrations, in particular Figure 6.7. The measurements in this region form a degenerate model. In general the result is very good on this well suited image.

The segmentation process is better illustrated on the *Field* image in Figure 6.9(a). The highly homogeneous sky region is identified from the initial low tolerance to local change, $\delta = 5$. As the tolerance increases, the grassy foreground and isolated sky regions, then the hillsides and large trees in the background, are revealed. Finally smaller less homogeneous regions are modelled. This example is indicative of typical results.

The next example, in Figure 6.9(b), illustrates undesirable characteristics of the algorithm implementation. Firstly the sky is identified by 5, or debatably more, regions. The yellow and orange regions which are identified first form extremely tight and almost degenerate clusters hence limiting growth where subtle changes occur. The remaining 3 basically fill in the gaps. It is perhaps desirable to increase λ_{min} to overcome this. Secondly, a similar *problem* occurs from small anomalous characteristics which cause numerous separate regions in the foreground field. In a way this is ironic since the algorithm depends on such *anomalous* features to correctly locate legitimate regions. Consequently no attempt is made to avoid this, but rather these are left to be combined by a higher level of analysis.

Figure 6.9(c) shows a *Bald Eagle* image in which the *important* features are not characterised by homogeneous regions. In this case, while the background and surrounding objects are well modelled the birds are too small to be considered. Similarly, the tiger in Figure 6.9(d) is not characterised by a single homogeneous region and hence results in numerous subregions.

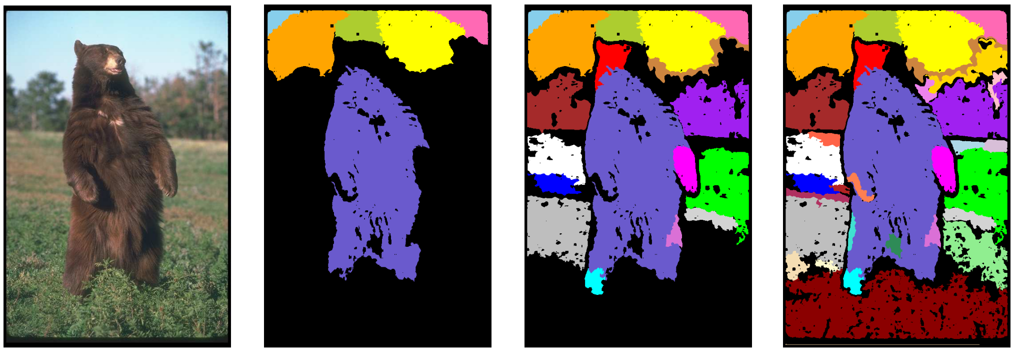
The final 5 results in Figure 6.9(e)–(i) are from the categories *Airshows*, *Polar Bears*, *Elephants*, *Sunsets* and *Canadian Rockies*. In all of these significant spatially disjoint homogeneous regions are successfully located. The final image contains numerous small regions of which 56 are identified. This segmentation does not appear particularly useful for identifying meaningful objects.

6.5.6 Computational requirements

The implementation is not well optimised. Throughout, output information is written to files detailing how regions expand and identifying the change in variable values such as the dynamic Mahalanobis threshold. Illustrative information is unnecessarily saved and models recalculated to maintain well structured code which is general and easily adaptable to explore algorithm modifications. Consequently the entire operation takes between 1 and 2 minutes for a typical image using a single R-10000 processor. This time is reduced by almost a factor of 3 by using a 400MHz Pentium 2 running Linux. Consequently, for the remainder of this chapter all times are based on this more common processor. The

(a) Field (028041) $\delta = 5$ $\delta = 23$

$\delta = 27$ $\delta = 59$ $\delta = 79$



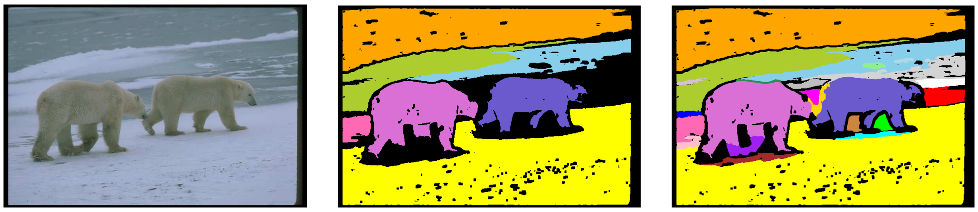
(b) Brown Bears (100037) $\delta = 19$ $\delta = 39$ $\delta = 79$

(c) Bald Eagles (135066) $\delta = 13$ $\delta = 75$

(d) Tigers (108020) $\delta = 23$ $\delta = 79$

FIGURE 6.9: [part 1] Segmentation results.

(e) Airshows (010081) $\delta = 11$ $\delta = 69$



(f) Polar Bear (100032) $\delta = 15$ $\delta = 79$

(g) Elephant (107000) $\delta = 19$ $\delta = 35$ $\delta = 79$

(h) Sunset (345031) $\delta = 13$ $\delta = 79$

(i) Canadian Rockies (276078) $\delta = 25$ $\delta = 79$

FIGURE 6.9: [part 2] Segmentation results.

breakdown of the computation time, unnecessary overheads, possible optimisations and general discussion follow.

A complete breakdown of the time taken to segment the sample aeroplane image is given in Table 6.2. The image loading, feature extraction and calculation of local change takes about 4 seconds regardless of the image content. As outlined in the implementation details these use general atomic routines and hence this time can be significantly reduced. Similarly the routines required to locate the largest seed region are non-optimal and require $38 \times 0.16 = 6.08$ seconds with little dependency on the data. A further 0.71 seconds is wasted to provide partial results which are only produced for illustrative purposes (e.g. Figure 6.9). The final remodelling of each region (1.03 seconds) can also be omitted by maintaining region models from calls to *ModelLargestBlob*.

Description of step	Routine	Time in seconds	Total time
loading and uncompressing image	*LoadImage*	0.01	0.01
feature extraction	*GetScaledFeature-Vectors*	2.19	2.20
calculation of Δ	*GetLocalChange*	1.81	4.01
modelling for $\delta = 5$ (sky, 4 iterations, 325081 pixels)	*ModelLargestBlob*	1.43	5.44
saving partial result	*ppmtogif*	0.25	5.69
modelling for $\delta \in \{7, 9, \ldots, 19\}$ (no regions found)	*ModelLargestBlob*	7×0.16	6.81
modelling for $\delta = 21$ (aeroplane, 14 iterations, 21738 pixels)	*ModelLargestBlob*	0.73	7.54
saving partial result	*ppmtogif*	0.23	7.77
modelling for $\delta \in \{23, 25, \ldots, 73\}$ (no regions found)	*ModelLargestBlob*	26×0.16	11.93
modelling for $\delta = 75$ (border, 5 iterations, 720 pixels)	*ModelLargestBlob*	0.42	12.35
saving partial result	*ppmtogif*	0.23	12.58
modelling for $\delta \in \{77, 79\}$ (no regions found)	*ModelLargestBlob*	2×0.16	12.90
image alignment	*PadImage*	0.01	12.91
recalculate all models	*GetGaussian-ModelFrom-ClassedImages*	1.03	13.94
save final regions	in **upwrite**	0.12	**14.06**

TABLE 6.2: Breakdown of time taken to segment the image from Figure 6.5 using a 400MHz Pentium 2 processor. The total time using a single R-10000 processor is 41.27 seconds. No code optimisations are considered.

In summary, for this image the ≈ 12 seconds outlined above can be reduced to 10 seconds by removing unnecessary code and maintaining region models, and further to 2–8 seconds through specific optimisations. The flood fill modelling time for the three regions ($2.58 - 3 \times 0.16 = 2.10$ seconds) constitutes the majority of the remaining time.

The total time taken to segment images from Figure 6.9 is outlined in Table 6.3. These times exclude saving partial regions. In general about 30 seconds is required for each image using this implementation.

class	ID	Figure	# regions	time
Canadian Rockies	276078	6.9(i)	56	51.10
Elephants	107000	6.9(g)	42	40.76
Brown Bears	100037	6.9(b)	34	36.24
Tigers	108020	6.9(d)	32	35.79
Sunsets	345031	6.9(h)	24	30.81
Polar Bears	100032	6.9(f)	24	29.46
Airshows	010081	6.9(e)	24	28.24
Fields	028041	6.9(a)	21	28.19
Bald Eagles	135066	6.9(c)	7	19.02

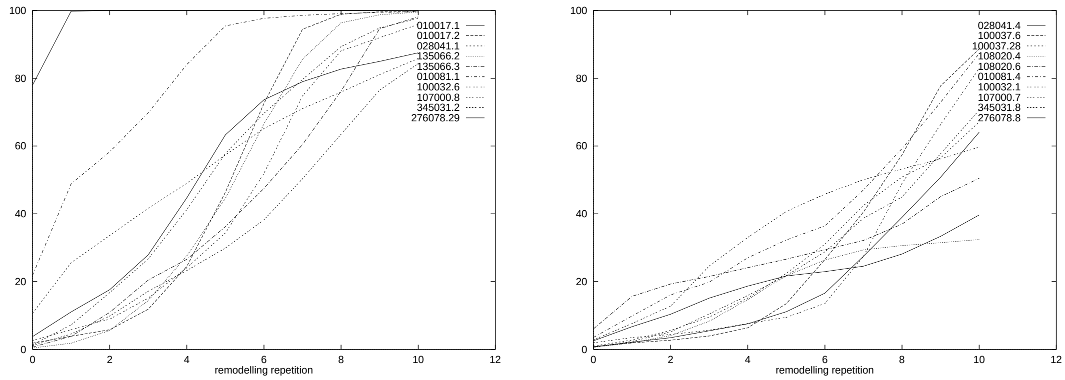
TABLE 6.3: Total time to segment the images using a 400MHz Pentium 2 processor. These times exclude saving partial results.

Possible algorithm optimisations

The iterative remodelling of region characteristics is inherently slow, but much improved by maintaining running sums. This constitutes the majority of the time which cannot be improved by code optimisations and is dependent on the number of times the region needs to be remodelled. Typically 10 to 30 flood fill iterations are required for each region, however in extreme cases this exceeds 50. Figure 6.10 shows the proportion of final regions located within the first 10 iterations for numerous regions requiring the indicated number of total iterations. Two regions are shown from each image based on their interest, size, number of remodelling iterations and time to *converge*.

To avoid unnecessary remodelling it is desirable to stop growth when the region approaches its *full size*. As seen from the table in Figure 6.10 there is no obvious general relationship between the size of the seed region, the region after 10 iterations and its final size. For example consider regions 108020#4 (foreground) and 276078#29 (reflection), both of which start from seeds of about 400 points. After 10 iterations the *foreground* region has grown to 22755 points and appears to have stabilised but ends up reaching 70212 points after 30 iterations. On the other hand the *reflection* reaches 10209 points at 10 iterations with a high gradient, but takes a total of 37 iterations to crawl to 11666 points. Many other cases are apparent from Figure 6.10.

In general the graph to the left is more typical with the majority of regions growth occurring within the first 10 iterations. Unfortunately there is no clear structure or stopping condition which significantly reduces computation without risking premature termination.



The proportion of region growth over the first 10 iterations.

region	colour	its.	seed size	after 10 its.	final size	description
010000#1	yellow	4	253497	325081	325081	sky
010000#2	orange	14	402	21703	21738	aeroplane
028041#1	yellow	34	10260	82413	96033	sky
028041#4	pink	20	846	69889	109085	vegetation
100037#6	slateblue	23	581	68080	72157	brown bear
100037#28	brown	23	581	35475	52866	foreground
135066#2	orange	15	440	99153	99603	background
135066#3	yellowgreen	22	1526	216618	220840	lake
108020#4	pink	30	425	22755	70212	foreground
108020#6	slateblue	59	430	3573	7079	vegetation (left)
010081#1	yellow	16	20784	94680	95016	sky
010081#4	pink	26	704	17347	19874	aeroplane component
100032#1	yellow	25	2240	94390	113833	vegetation
100032#6	slateblue	27	565	20338	21210	polar bear (right)
107000#7	orchid	28	613	12817	21473	elephant back
107000#8	peru	16	702	50574	51779	foreground
345031#2	orange	46	774	57784	68526	orange sky
345031#8	peru	24	695	75062	106318	ocean
276078#8	peru	27	528	8231	20750	water
276078#29	turquoise	37	442	10209	11666	bottom left reflection

FIGURE 6.10: The number of remodel iterations required to locate regions from illustrated results.

6.5.7 Closing comments

The final segmentation is very successful at locating regions which are homogeneous with respect to the feature vectors. The algorithm relies on using the distance **d** as a measure of homogeneity to determine seed regions. This further relies on window measurements with comparable expected standard deviations or Euclidean distances in the feature space. The method does not rely on static parameters or global values, but rather models each region relative to its local characteristics and grows regions accordingly. In view of the overall framework, the location of a few smaller sub-objects corresponding to what appear to be the same region is acceptable, whereas the inappropriate combination of non-homogeneous areas is not.

The final results are good although they suffer from the inherent problem that from both an information theoretical and intuitive point of view, most of the information is contained in areas which are anomalous to the otherwise homogeneous regions. These occur at boundaries, or edges between regions, or correspond to detailed specific features such as a house on the horizon, a tractor in a field or the facial features of animals. Obviously the results can be interpreted to account for this, the regions may be used as a mask to roughly locate edges and regions of interest, or the algorithm used in conjunction with anomaly detection methods.

The main problem with the algorithm is its inability to locate regions which consist of a mixture of *colours*. For example, a green tree and a section of sky may occupy the same region hence indicating the presence of two homogeneous objects at the same location. In this case a colour histogram would most likely form a bimodal distribution, hence allowing more traditional CBIR techniques to match static regions between images. To cater for this the algorithm presented here would need to identify seed regions corresponding to a mixture of two *clusters* in the feature space, model these using a Gaussian mixture model and perform the flood fill algorithm considering the Mahalanobis distance of neighbouring points with respect to each mixture. Ideally such a scheme would allow the tree and sky to be located as two separate, but spatially overlapping, regions. It follows that objects such as tigers would be located as an overlapping mixture of two homogeneous regions rather than consist of smaller sub-parts. This extension, or alternative amalgamation process, has not been implemented.

Overall the segmentation algorithm is effective in locating spatially disjoint homogeneous regions with reasonable computational requirements. It is not well suited to *busy* images such as those found in cities or human environments.

6.6 Representation of disjoint regions

6.6.1 Measurement representation

The window measurements and spatial position form a set of points in $\mathbb{R}^6$. To represent the local characteristics these are modelled with low order geometric moments, namely the number of data points n, a mean μ and covariance matrix $\mathbf{C}$. These provide information pertaining to region size, the expected value of window measurements and position, the spread of these vector element values, and the linear change with respect to each of them. The properties and attractive characteristics of this representation are apparent from Chapter 4. For instance μ and $\mathbf{C}$ allow the window measurements to be predicted at a specific spatial location as illustrated by a partial DownWrite to restore anomalous images.

This representation completes an UpWrite in which an image, or set of points in $\mathbb{R}^5$, is transformed into a number of region represented as points in $\mathbb{R}^{1+6+21} = \mathbb{R}^{28}$. This particular UpWrite captures only the basic local characteristics of each region and does not contain detailed shape information.

6.6.2 Shape representation

A number of different shape representations are discussed in Section 2.1.3. One of the simplest representations is the use of geometric moments. Obviously the information from the zeroth, first and second moments is already contained in the measurement representation outlined above. Storing higher order moments is one technique of capturing shape, although it is not clearly appropriate for describing and defining a distance between objects such as aeroplanes, trees, bears, sky regions, et cetera. Since the majority of regions produced by the segmentation algorithm are solid, or contain a few isolated holes, an alternative is to examine the boundary shape information.

Consequently, the FT-boundary representation has been considered for shape representation. After performing a border trace, N evenly spaced co-ordinates are used to model the object's shape. Ideally this provides invariances to translation and size, but allows for distinction between aspect ratios. However, in practice it is difficult to obtain N (e.g. 64) *evenly spaced* co-ordinates. Using points evenly spaced around the perimeter unfairly distinguishes between shapes with rough edges or small concavities, and using angular spacings clearly fails for significantly concaved objects. Given an appropriate set of points, a simple yet effective method of comparison is to minimise the squared error between two shapes by representing each in a desirable polar co-ordinate system. Choosing the origin at the object's centre of mass and scaling radially provides the appropriate translation and scale invariances. Rotation invariance is achieved by brute force. The alternative is to directly compare the FT-boundary representation to achieve this phase invariance. Such representations require further investigation and are not considered in this thesis.

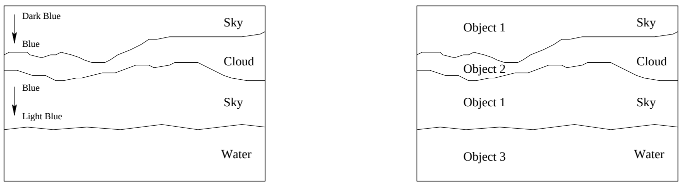
6.6.3 The upwrite program

The **upwrite** program uses a routine called *SaveRegion* to represent each region. Rather than storing n, μ and elements of $\mathbf{C}$ the number of data points, the sums of elements and sum of products are stored. This results in the same number of points, i.e. a set of points in $\mathbb{R}^{28}$, and allows the mean and covariance matrix to be recalculated. However, it also allows regions to be amalgamated with the mean and covariance matrix easily calculated for the corresponding object. This representation is a permissible UpWrite.

The program also stores the shape of each region. Rather than deciding on a representation that is a point in a Euclidean space, the choice of the appropriate representation is deferred. Each region is represented by a *quad-tree* which provides a lossless shape representation stored by an arbitrarily long string (corresponding to a tree). It is important to note that this representation does not describe important shape features and cannot be used to directly obtain a measure of shape similarity. Neither is it within the scope of the adopted framework.

6.7 Amalgamation of represented disjoint regions

The segmentation algorithm described so far in this chapter identifies connected regions with homogeneous characteristics reasonably well. This process does not consider small local perturbations, or subtle boundaries, which may cause a single object to be identified as multiple regions, nor does it cater for occlusion. To cater for these in the general case, regions should be amalgamated based on characteristics such as intensity, colour, shape, the distance between them, and positional changes in local intensity or colour. Under the adopted framework this is performed using the representation, or model, of each region.



(a) Image content. (b) Ideal object segmentation.

FIGURE 6.11: This example illustrates a common pattern recognition problem known as occlusion.

6.7.1 Using the Fisher distance

The Fisher distance, defined on page 82, provides a measure for merging regions. Application of this general criterion produces unsatisfactory results which are almost arbitrary in places. It does not consider variations with respect to position, the distance between regions, and is heavily dependent on the regions' standard deviation. For example, the regions depicted in Figure 6.12 have a Fisher distance of 0 since they have the same mean. Also, because the Fisher distance is inversely proportional to the weighted variance, coarse regions tend to reduce the distance. Consequently a rough grey rock to the left of an image is deemed similar to a green bush to the right. In general the Fisher distance is an inappropriate measure for this application.

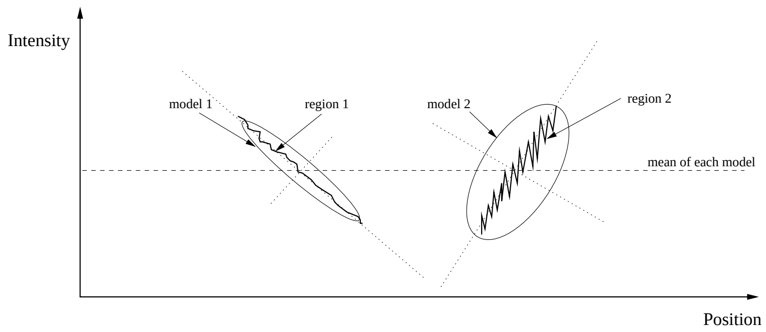


FIGURE 6.12: Regions which inappropriately have a Fisher distance of 0. The lines marked 'region' correspond to fragments of the 'intensity' of the image (similar to Figure 6.7). The ellipse corresponds to the region's model. The width of the ellipse is proportional to the standard deviation of the model along its minor axes. These synthetic regions could correspond to the sky and its reflection off a lake.

6.7.2 Mutual prediction method

Rather than attempt to use a simple distance, it is far more powerful to consider each region model as a predictor.[7] Figure 6.13 illustrates an image containing five modelled regions. Regions 1, 3 and 5 are part of the same object, for example the sky, and regions 2 and 4 represent objects which overlap this object, for example the trunks of two different trees. As is clear from this illustration, the model for region 1 can be used to predict the local measurement at the location of regions 3 and 5, and similarly the model for region 5 adequately predicts the local characteristics of regions 1 and 3.

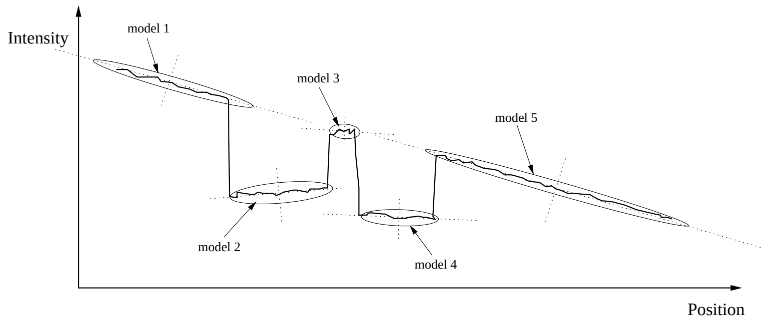


FIGURE 6.13: Isolated regions corresponding to the same object separated due to occlusion.

Extending this illustration to 4 measurements and two positions provides a mechanism for using the model of a region to predict the characteristics at any other position in the image. Since both the position and local measurements are modelled using up to second order central geometric moments (the number of points n, mean μ, and covariance matrix $\mathbf{C}$) the prediction matrix Ψ, as derived in Section 4.9.1, is used.

[7]This is similar to the examples of the UpWrite applied to stroke detection in binary images. Refer to Section 2.8.3 and in particular to Figure 2.58.

Regions are amalgamated if one region predicts the characteristics at the other region's mean within a small threshold and visa versa. This is satisfactory unless a region is extremely small, in which case noise and variations at the region's edge significantly distort the model, making it inappropriate as a predictor. For example, in Figure 6.13 region 1 and 5 successfully predict region 3, however region 3's prediction is less convincing. Consequently more tolerance is allowed for predictions made by small regions.

Results are shown in Figure 6.14. Of interest, the foreground of Figure 6.14(b) is amalgamated, including the region between the bear's legs. A similar foreground amalgamation occurs in (c), in which the sky is combined into two larger regions. In (e), the majority of the water is combined into one background object, and regions of the bird's wings and neck are appropriately joined.

6.7.3 Dendrogram clustering

The mutual prediction method provides a reasonable measurement between two remodelled regions. Since, in this case, the UpWrite process outputs the running sums necessary to form region models (n, μ, $\mathbf{C}$), two or more regions can easily be combined into a single new model. By combining the 'closest' two regions, i.e. those which best predict one another and iterating this process, the regions can be clustered. This hierarchical clustering forms a binary tree called a dendrogram (see page 33).

By using a cluster tree formed in this way, the image can quickly be segmented into a defined number of amalgamated regions or segmented using a defined threshold. Such amalgamation requires further investigation, in particular to consider the shape representation of both the individual and amalgamated regions. Also, to some extent the amalgamation and remodelling of regions blends spatially disjoint homogeneous areas of the image into the same model. Despite these issues, the application of a cluster tree to this mutual prediction method is another useful image analysis tool.

6.7.4 Conclusions

In conclusion the ability for spatially disjoint regions to be combined in this way generically caters for occlusion, intra-region noise and small intra-region artifacts. However it also allows different objects with the same characteristics to be identified as a single entity and does not consider the shape, relative position, nor the distance between the regions.

6.8 Classification

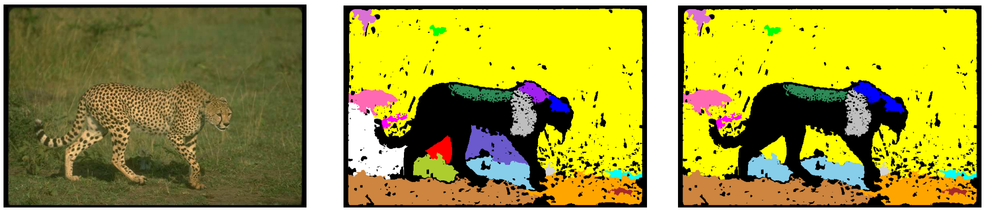
6.8.1 Appropriate images

Figure 6.15(a) shows a number of images taken from the *Canadian Rockies* photo CD which are clearly inappropriate for inclusion in classification. Given the available classes, a human observer would assign the images (from left to right) to the categories *Brown*

(a) Deserts (032004) | $\delta = 75$ | amalgamated regions

(b) Polar Bears (100002) | $\delta = 75$ | amalgamated regions

(c) Elephants (107053) | $\delta = 79$ | amalgamated regions



(d) Cheetahs (134025) | $\delta = 75$ | amalgamated regions

(e) Bald Eagles (135087) | $\delta = 73$ | amalgamated regions

FIGURE 6.14: Prediction results.

Bears, *Sunsets*, *Fields*, and would be unable to justify classifying the final image. As a result, researchers working on the Berkeley digital library project (see Section 2.7.4) omitted these images. However, they also omitted images which seem appropriate for classification. Figure 6.15(b) shows four images from the *Airshows* category. The first and third images from the left were omitted and the remaining two used.

(a) Inappropriate images from the *Canadian Rockies* category.

(b) Images selectively omitted or used in [264] [263]. The first and third images from the left were omitted.

FIGURE 6.15: Example images which are possibly inappropriate for classification.

To avoid subjective judgement all 1000 images are used for classification. Each image is oriented as presented on the photo CD without any rotation.

6.8.2 Human classification

As outlined in Figure 6.15, the 'correct' classification of each image is not clear cut. To provide an insight into this, a very simple human classification experiment was conducted. Each assessor was presented with all 1000 images in a different random order, with the choice of either one of the eleven classes, or the option of claiming that they 'could not possibly justify the image belonging to any of the classes' (for example the right hand image in Figure 6.15(a)). No prior training, other than general knowledge and a small amount of exposure to the Corel images, was given. The assessor was encouraged to classify the images wherever possible, but was not given the opportunity to go back and change any classification. Consequently the individual to some extent became more familiar with the classes on the fly, making the results dependent on the random ordering. The entire process took between two and three hours.

Table 6.4 shows the confusion matrix for the first assessor, a graduate student who obtained a classification rate of 94.4%. It is interesting to note the 100% classification rate of the better defined classes, namely *Airshows*, *Polar Bears*, *Brown Bears*, *Elephants*, *Tigers*, *Cheetahs* and *Bald Eagles*. Excluding these classes, the classification rate drops

Class	A	F	D	PB	BB	E	T	C	BE	CR	S	–
Airshows	100	0	0	0	0	0	0	0	0	0	0	0
Fields	0	90	3	0	0	0	0	0	0	4	0	3
Deserts	0	3	66	0	0	0	0	0	0	8	1	22
Polar Bears*	0	0	0	100	0	0	0	0	0	0	0	0
Brown Bears*	0	0	0	0	100	0	0	0	0	0	0	0
Elephants	0	0	0	0	0	100	0	0	0	0	0	0
Tigers	0	0	0	0	0	0	100	0	0	0	0	0
Cheetahs	0	0	0	0	0	0	0	100	0	0	0	0
Bald Eagles	0	0	0	0	0	0	0	0	100	0	0	0
Canadian Rockies	0	1	0	0	1	0	0	0	0	90	2	6
Sunsets	1	0	1	0	0	0	0	0	0	0	98	0

TABLE 6.4: The classification of the Corel photo CD images by a human. Note (*) that the *Bears* photo CD consists of 35 *Polar Bears* and 65 *Brown Bears.*

to 86.0%. The second and final assessor, also a graduate student, obtained 94.0% with approximately 10 more correctly classified *Desert* and 10 fewer *Field* images.

This simple experiment provides an insight into the data, outlining that about 5% of the images are either badly classified or misclassified. The variation in human judgement and post-experiment debates over the class of specific images, also indicates the subjectiveness of the problem. Consequently, a 100% classification rate is not expected even given a perfect algorithm.

6.8.3 Simple single region based classification

The segmentation results produce numerous regions which cannot easily be represented as a single entity for comparison, clustering or classification. Consequently a simple classification scheme is proposed which uses the identified region which is closest to the centre of the image. A second constraint is that the region must be reasonably large to prevent the image from being classified by a small fragment.

A feature vector obtained from the model of this region is used to represent the entire image. This feature vector consists of the means of the window measurement and the standard deviation of the $(\mathbf{x}, \mathbf{y})$ spread, all normalised accordingly.

A simple K nearest neighbours scheme is used in which each image is classified based on all other images. Classification results yield 55.1% and 50.5% for $k = 1$ and $k = 3$ respectively. The confusion matrix for $k = 1$ is shown in Table 6.5.

The results for this simple scheme are encouraging. For every image class the highest value appears down the diagonal. The weakest classification rate is obtained by *Deserts*, which do not seem well categorised, only yielding a rate of 66% by a human (Table 6.4). Many *Airshows* are misclassified as *Polar Bears*, *Bald Eagles* and *Fields* possibly because they typically contain clouds, sky and are set in fields, respectively. Similarly *Polar Bears* which are white are grossly misclassified as *Airshows* and *Bald Eagles* which often contain

Class	A	F	D	PB	BB	E	T	C	BE	CR	S
Airshows	70	6	3	8	1	1	1	0	6	3	1
Fields	6	48	10	3	2	1	13	6	2	7	2
Deserts	0	10	23	2	8	8	12	12	7	12	6
Polar Bears*	29	6	0	43	0	3	3	0	11	3	3
Brown Bears*	2	3	2	0	48	9	18	5	12	0	2
Elephants	4	1	6	2	5	64	9	5	2	1	1
Tigers	0	9	7	0	11	12	48	5	3	4	1
Cheetahs	3	3	13	1	5	7	4	58	2	1	3
Bald Eagles	5	1	9	1	5	2	5	0	70	1	1
Canadian Rockies	6	10	11	1	0	1	6	0	4	56	5
Sunsets	2	2	6	0	2	4	4	5	3	4	68

TABLE 6.5: Classification results for 1000 Corel photo CD images, yielding an accuracy of 55.1%. Note (*) that the *Bears* photo CD consists of 35 *Polar Bears* and 65 *Brown Bears.*

large dominant clouds. A high misclassification rate between the less well defined *Canadian Rockies*, *Fields* and *Deserts* classes is apparent. The undesirable confusion between *Brown Bears* and *Tigers* is probably caused by modelling their surrounding vegetation.

6.9 Image region reconstruction via a partial DownWrite

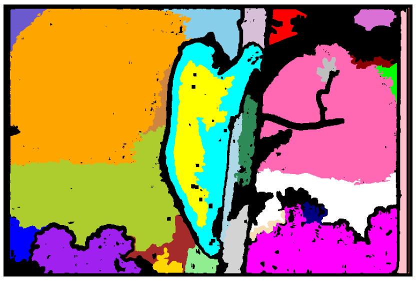
As outlined in Section 3.2 and illustrated in Section 4.9, the DownWrite provides a powerful insight into the information captured by an object's representation. For natural colour scenes, as well as providing insight into and outlining limitation of the model, it is also aesthetically pleasing.

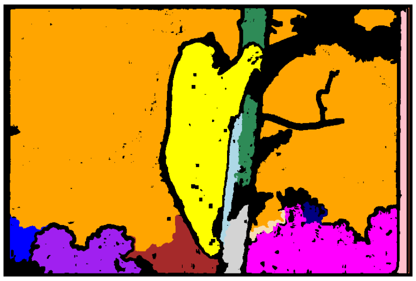
In this case, the representations of local characteristics of each region are reduced to models using geometric moments of simple window measurements. As outlined in Section 6.7.2, the prediction matrix Ψ allows the model of each region to predict the local characteristics at any position in the image. By making use of the quad-tree representation of each region, the shape of each region is reconstructed. For each position within the region the prediction matrix is used to reconstruct each pixel. More completely, the image reconstruction procedure uses the following steps.

- For a selected region model j, calculate the prediction matrix Ψ_j by restricting the last 2 of the 6 values. This matrix is derived in Section 4.9.1.
- Using the quad-tree shape representation, ascertain all ($\mathbf{x}$,$\mathbf{y}$) coordinates in the region.
- For each point, calculate the expected value of the window measurements, as $\hat{\mu}_j = \Psi_i[0,0,0,0,x-\mu_{j_x}, y-\mu_{j_y}]^T$, where μ_{j_x} and μ_{j_y} are from the mean of model j.
- Use $\hat{\mu}_j$ and $\mathbf{C}_j$ to recreate an artificial point. After adding random Gaussian noise with the appropriate standard deviation to the elements of $\hat{\mu}_j$, values for intensity


(a) Brown Bears (100057)


(c) DownWrite


(b) $\delta = 77$


(d) amalgamated regions


(e) Fields (028025)


(g) DownWrite


(f) $\delta = 55$


(h) amalgamated regions

FIGURE 6.16: DownWrite examples with segmentation and region amalgamation results.

and colour measurements are obtained. These are restored by inverting the equations outlined in Section 6.3.4 to produce traditional hue, saturation and intensity values, which are in turn converted into red, green and blue pixel values.

- Repeat this procedure for all modelled regions. All image locations which are not reconstructed are left as black.

The following are noted.

- The DownWrite is partial. It is performed on the model of local characteristics for each region. The shape information is reconstructed in a lossless manner and is not part of the DownWrite.
- Each instance of the DownWrite is not unique due to the random perturbations in the process, although they look the same.
- Each pixel is constructed independently, despite having been modelled by measurements from a local window.
- The DownWrite is performed using the initially located region and not those created after region amalgamation.
- The DownWrite program uses the output from the UpWrite and does not use the original image directly in any way.

Full segmentation results, including initial regions, amalgamated regions and the DownWrite appear in Figure 6.16. Most apparent from the brown bear images are the missing regions around the tree's branches and the inability to reproduce the texture of the trees in the background. Otherwise, the segmentation is reasonable and the amalgamation of the tree trunk and multiple sky regions is encouraging. The example from the *Fields* category is somewhat more artistic. As outlined in Section 6.5.7, the areas of highest information are left unassigned and are hence black in the reconstruction. Consequently an observer of the partial DownWrite is left to interpret the missing details. These objects may be misconstrued as a tank, some sort of aircraft, and a base or building.

In general, this partial DownWrite outlines the limitations of the model of each region, as well as indicating the importance of shape information in the recognition process. It also provides an introspective view of human perception and may be a useful tool for generating art.

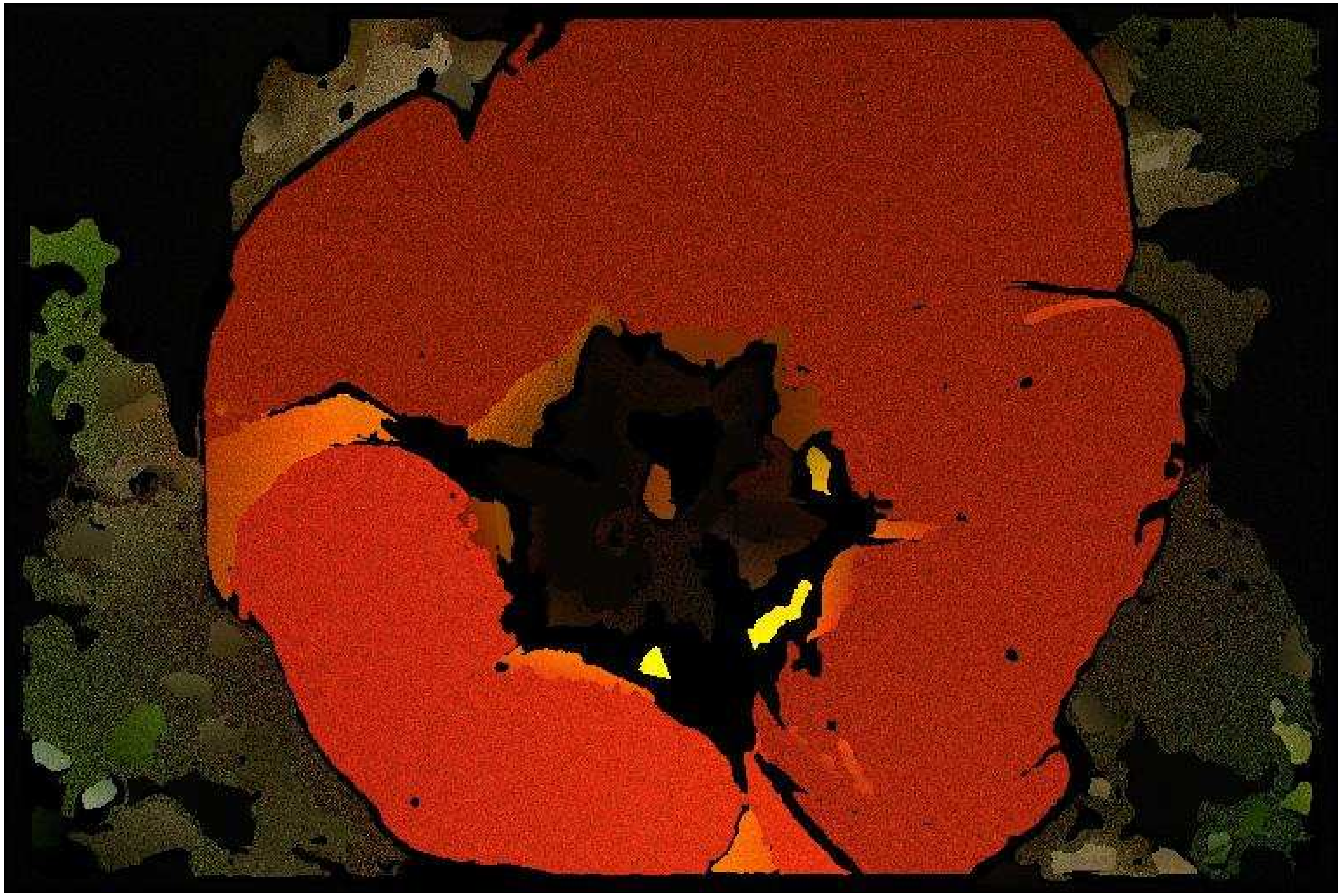


FIGURE 6.17: The DownWrite may be useful for computer generated art.

6.10 Summary and concluding remarks

The segmentation algorithm outlined in this chapter uses the same feature extraction method used for anomaly detection and mixed-mode segmentation. Window measurements for average intensity, standard deviation of intensity and average colour provide adequate information for segmentation. For colour representation, a novel tradeoff between the traditional HSI conical and cylindrical spaces is applied to overcome the inadequacies of these standard representations. Furthermore, colour values are scaled based on the colour distribution of outdoor scenes. A measure of local deviation is not used in the final segmentation algorithm, although several are examined.

Using these local features, a region growing type algorithm is applied to identify spatially disjoint regions which are homogeneous with respect to local window measurements. The algorithm has few parameters, tending to expand regions with respect to measurements from incremental modelling. The main growing criterion is a threshold of the Mahalanobis distance between neighbouring feature vectors with respect to the region's multivariate Gaussian model. This threshold is adapted based on its spread in the bounded feature space. This adaptive threshold overcomes leaking between regions which is a typical problem of region growing segmentation algorithms.

The algorithm produces good results over a wide range of images, successfully identifying disjoint homogeneous regions without leaking into neighbouring ones. However, the algorithm could be improved by local mixture modelling or multi-scale analysis. Modelling regions using an *appropriate* number of mixtures would allow overlapping regions, such as a tree with sky in the background, to be more completely identified. Analysing the image at multiple scales would produce numerous segmentations. Subsequent correlation between these may improve, or at least re-ensure, segmentation results.

Following segmentation, the regions characteristics are modelled appropriately to allow comparison. Using features extracted from this model, 1000 Corel photo CD images from 11 categories are used for classification. Results yield between 50 and 56%, as compared to 54.5% obtained by researchers of the Berkeley digital library project.[8] This measurement between individual regions is also appropriate, and possibly better suited to, user queries for content based image retrieval.

The region models are also used to amalgamate smaller sub-regions into more complete objects and to demonstrate a partial DownWrite. The results here are also encouraging, outlining the effectiveness of multivariate Gaussian modelling and indicating inadequacies produced by simplifying the feature vectors into simple local window measurements.

Overall, the methods used fit into the framework of the UpWrite. Low level primitives are combined to form feature vectors which are used by a region growing type algorithm to identify and represent higher level entities. These phases constitute the first real UpWrite

[8] The results achieved by researchers of the Berkeley digital library project omitted and rotated a significant proportion of the images, although included the additional *Night Scenes* category. Refer to Section 2.7.4 and 6.8 for further details.

in which feature vectors are used to form spatially disjoint regions. The local characteristics of each region are modelled by a set of real numbers, in this case a point in $\mathbb{R}^{28}$. From this representation the appropriate DownWrite provides a means of reconstructing the local characteristics of the region.

The amalgamation of regions constitutes a second UpWrite, which uses a set of regions in $\mathbb{R}^{28}$ to produce a set with equal or fewer regions, also in $\mathbb{R}^{28}$. This UpWrite has the desirable property that the representation both before and after are the same. It is also interesting since it does not provide an associated DownWrite, but rather supports the same DownWrite as the first level UpWrite.