Chapter 2

Background

In this chapter, we give a history of previous work which is related to work contained in this thesis. Along the way, we alert the reader to the high level differences between the previous work and our work.

2.1 The 1D Shape Pattern Problem

The core of this thesis begins in chapter 3 with an algorithm to find a polyline shape pattern within another polyline. Using this algorithm, one can summarize/simplify the description of a polyline by finding pieces of the polyline with more compact descriptions such as "line segment", "corner", or "circular arc".

There are many methods for finding geometric primitives in polylines. The classic pattern recognition book [56] by Pavlidis is an early computer vision reference for approximating, within some error bound, polylines with many vertices by polylines with fewer vertices. Some such approximation algorithms, which essentially find line segments within a polyline, are given in [56] in the chapter titled "Analytical Description of Region Boundaries and Curves". Three excellent, contemporary works from the computer vision community are the Lowe segmentation algorithm ([45]) to divide an edgel chain into straight segments, the "strider" algorithm of Etemadi ([23]) to find straight segments and circular arcs, and the Rosin and West algorithm ([62]) to identify line segments, elliptical arcs, and other high-order curves.

The crucial issue in all such segmentation algorithms is where to stop one description and to begin another. The three works [45], [23], and [62] are all similar in that their breakpoint selection does not use more usual, curvature-based criteria such as curvature zero crossings and extrema. Curvature is sensitive to noise, and the mentioned breakpoint conditions may divide a curve that does not have constant curvature. For example, breaking

descriptors at curvature extrema may result in the division of an elliptical arc into two or more pieces. Regardless of the breakpoint strategy, a single pass through the polyline data may not yield very good results. Each of the previously mentioned algorithms also has a second phase that considers replacing two descriptions of adjacent curve pieces with a single description over their union.

Etemadi's "strider" segmentation algorithm ([23]) uses a symmetry condition to determine an initial set of breakpoints. A chain of pixels is labelled as symmetric or asymmetric using the midpoint R of the pixel chain and the line segment PQ connecting its endpoints. The line through R and perpendicular to PQ splits PQ into two segments PS and SQ. The chain from P to Q is in a symmetric state iff the difference in lengths of PS and SQ is less than $1/\sqrt{1+L^2}$, where L is the length of the chain. Note that circular arcs and straight segments are symmetric chains. The strider algorithm adds pixels to a chain until the chain is in an asymmetric state for three consecutive pixel additions. The process then begins again from the first pixel which caused the chain to become asymmetric.

The segmentation algorithms of Lowe ([45]) and Rosin and West ([62]) both use a maximum deviation criterion to compute breakpoints and a scale invariant "significance" formula to decide whether to split a chain further. The Rosin and West algorithm is a generalization of the Lowe algorithm to handle geometric primitives other than line segments. Lowe's algorithm recursively computes a segmentation tree for a pixel chain. The entire chain is approximated by a single segment and then split into two subchains at the pixel which deviates most from the approximation.[1] The *significance* of the approximation is the ratio of the approximation length to the maximum deviation. The splitting algorithm then recurses on the two subchains. The final segmentation into straight lines is computed by traversing the tree up from the leaves and retaining a segment if it is more significant than its children.

The above idea can be applied using any geometric primitive curve for which there is a method for fitting such a curve to pixel chain data. The maximum deviation between the fitted representation and the data can be obtained by computing the maximum of the minimum distances from each chain pixel to the approximation curve. The significance measure remains the same ratio of approximation length to maximum deviation. The first step in the Rosin and West strategy ([62]) is to compute a line segment representation using Lowe's algorithm. The second step applies Lowe's algorithm again to divide the line segment representation into higher order curves such as ellipses or superellipses. The step

[1] An earlier use of the idea to split at the point of maximum deviation is the Douglas-Peucker (poly)line simplification method ([16]). This method approximates a polyline with one of fewer vertices that is within a given error bound ε. If the error in approximating a polyline by a single line segment is greater than ε, then the polyline is split into two at the maximum deviation vertex, and the approximation procedure is recursively applied to the two smaller polylines.

one line segmentation is kept at the leaves of the step two segmentation tree, so not all line segments are necessarily replaced by a higher order curve. Unlike in Lowe's algorithm, the upward tree traversal considers any two adjacent approximations for combination into a single approximation (versus considering only approximations with a common parent). See [62] for algorithm details.

In the computational geometry community, the process of approximating a polyline with a smaller number of segments is called the *min-#* problem. Given a polyline and an error bound, the goal is to compute an approximation polyline with the fewest number of vertices whose distance to the original polyline is within the error bound. If the vertices of the approximation are required to be a subset of the n vertices of the original, then the problem can be solved in $O(n^2)$ time using a graph formulation. The main idea is to construct a graph G with nodes equal to the vertices of the given polyline and where there is an edge from v_i to v_j iff the polyline from v_i to v_j can be approximated within the error bound by a segment from v_i to v_j. The number of vertices in the smallest approximation is equal to the number of edges in a shortest path between the nodes for the first and last vertices. Chan and Chin [6] show how to compute G in $O(n^2)$ time (the brute force method requires $O(n^3)$ time). Since a shortest path in G can be computed in $O(n^2)$ time, the overall $O(n^2)$ bound follows.

Our work on the 1D shape pattern problem does not look for patterns from a particular family of curves. Instead the input pattern can be any polyline shape. If the given "image" polyline has n vertices and the pattern polyline has m vertices, then our algorithm requires $O(m^2n^2)$ time. This reduces to $O(n^2)$ time if the pattern is of constant size (e.g. $m = 2$ for a line segment and $m = 3$ for a corner). Our algorithm was inspired by the Arkin et al. polygon shape metric work [3]. This work compares two polygons by comparing their arclength versus turning angle graphs once the total arclength of both polygons has been normalized to one unit. To make the metric invariant to rotation, the authors allow for an up-down shift of the turning angle graph; to handle the arbitrariness of the first vertex in a polygon description, the authors allow for a cyclic left-right shift of the turning angle graph. There is some experimental evidence ([72]) that the turning angle graph is one of the best shape descriptors for judging perceptual similarity.

We adopt the approach of matching turning angle graphs in our search for a pattern. For the pattern problem, however, we need to allow for partial matching and changes in scale (in addition to changes in orientation). Partial matching means that we match the pattern graph to only a piece of the image graph. To handle a scale change, we allow the arclength axis to be stretched or contracted before matching the graphs.

2.2 Focused Color Searching

In the literature, *focused color searching* refers to the problem of finding a color pattern or model within a color image. The goal is to *focus* on a region of the image which contains the pattern or ascertain that no such region exists. We refer to this problem as the *color pattern problem.*

An early paper in this field is by Swain and Ballard ([77]) who introduced the *Histogram Backprojection* algorithm. If we denote the image and model color histograms as $I = (I_j)_{j=1}^n$ and $M = (M_j)_{j=1}^n$, respectively, then the first step in Histogram Backprojection is to compute the quotient histogram $R = (R_j)_{j=1}^n$, where $R_j = M_j/I_j$. The value R_j represents the probability that an image pixel with color j belongs to an occurrence of the model in the image, assuming that the model appears in the image. If the model appears in the image, then $I_j = M_j + C_j$, where C_j is the number of pixels in the image with color j that are not part of the model. This analysis assumes that the model described by histogram M is the same size in pixels as its occurrence within the image described by histogram I (which is true if the model is cut out from the image). The more "clutter" pixels C_j of color j, the less likely it is that a random image pixel with color j is part of the model.

The second step in Histogram Backprojection is to replace the image color at every pixel with its probability of being part of the model, thus forming the *backprojection image.* Pixels with color j are replaced by the confidence value $\min(R_j, 1)$. White (confidence close to one) regions in the backprojection image are places where the model is likely to occur, and black (confidence close to zero) regions are places where the model is unlikely to occur. The final step in the Histogram Backprojection algorithm is to find the location of the maximum value in the backprojected image after it has been convolved with a mask of the same area as an expected occurrence of the model. This convolution sums confidence values over local areas, and the location of the maximum in the result is the place where the model is most likely to occur.

In the same paper [77], Swain and Ballard also introduce a measure of histogram distance called *Histogram Intersection.* The Histrogram Intersection between image histogram I and model histogram M is defined as

$$H(I, M) = \frac{\sum_{j=1}^n \min(I_j, M_j)}{\sum_{j=1}^n M_j}. \tag{2.1}$$

The numerator of (2.1) is the number of pixels from the model that have corresponding pixels of the same color in the image. The normalization in the denominator of (2.1) guarantees $0 \leq H(I, M) \leq 1$. This measure was used to determine the identity of an unknown model

at a known position in the image. A weighted version of this measure, however, is also at work behind the scenes of the Histogram Backprojection algorithm to find the unknown location of a known model in an image. Let L^S denote the local histogram taken over a $w \times h$ rectangular subregion S of the image. Now suppose that the final Backprojection step uses a $w \times h$ rectangular mask of all ones. If we let $S_{w \times h}$ denote the collection of all $w \times h$ image subwindows, then the place where the model is most likely to occur is the solution to the optimization problem

$$\begin{aligned} \arg \max_{S \in S_{w \times h}} \sum_{(k,l) \in S} 1 \cdot \min(M_{j(k,l)}/I_{j(k,l)}, 1) &= \arg \max_{S \in S_{w \times h}} \sum_{j=1}^{n} L_j^S \min(M_j/I_j, 1) \qquad (2.2) \\ &= \arg \max_{S \in S_{w \times h}} \sum_{j=1}^{n} \left(\frac{L_j^S}{I_j} \right) \min(M_j, I_j), \end{aligned}$$

where $j(k,l)$ is the histogram bin index for the image color at pixel (k,l), and we have used the fact that $\min(M_j/I_j, 1) = \min(M_j, I_j)/I_j$. The weight quantity L_j^S/I_j is the fraction of image pixels of color j that appear in window S.

One major problem with the original Swain and Ballard approach is that its final convolution step essentially assumes the size at which the model occurs in the image. The idea of Ennesser and Medioni in [22] is to compare histograms of local image areas of different sizes and locations to the model histogram using *weighted Histogram Intersection* to measure histogram similarity. The authors suggest using the weighted Histogram Intersection formula

$$\widehat{H}_W(L^S, M) = \sum_{j=1}^{n} W_j \min(L_j^S, M_j),$$

with weights $W_j = 1/I_j$ or $W_j = R_j = M_j/I_j$. Here it is assumed that the model histogram M is normalized to match the total amount of information in the image subwindow S: $\sum_{j=1}^{n} M_j = \sum_{j=1}^{n} L_j^S$. The quantity $\min(L_j^S, M_j)$ is the number of pixels from the scaled model that have corresponding pixels of the same color in the image subwindow S. The weight W_j is an attempt to give more importance to colors j that are more distinctive in the matching process. If the local histogram region S contains the model, then $\min(L_j^S, M_j) = L_j^S$ (remember that the model histogram is normalized to the same total bin count as the local image histogram) and $\widehat{H}_R(L^S, M) = \sum_{j=1}^{n} L_j^S(M_j/I_j)$, which is the same value used in the Backprojection algorithm. This last observation follows from (2.2) and that fact that $M_j \leq I_j$ (due to the normalization of the size of M).

Ennesser and Medioni's *Local Histogramming* algorithm looks for model matches as it slides a local window across the image. For each center location of the local window, the algorithm increases the size of the window until the weighted Histogram Intersection

measure given above starts decreasing. There is some amount of manipulation to ensure that this scale estimation process avoids local maxima as the window is grown, and that a model is not found in pieces by overlapping local windows. See [22] for some details. The bottom line, however, is that the Local Histrogramming algorithm tries to find the model by an exhaustive search over scale-position space, where a (scale,position) pair is evaluated by a weighted Histogram Intersection between a local image histogram and the scaled model histogram. Swain and Ballard's Backprojection algorithm is an exhaustive search only over position since they assume the scale parameter. The similarity measure used to check a position is a weighted Histogram Intersection between the global image histogram and the model histogram, where the weights are dependent on the position (and assumed scale).

The same exhaustive search over scale-position space using local image histograms is performed by Vinod et al. in [83, 82], but with an upper bound on the Histogram Intersection measure that can be used to prune quickly unattractive (scale,position) pairs. The Histogram Intersection between the model histogram and two local histograms for image regions of similar position and scale will be similar since the two regions have many pixels in common. If the Histogram Intersection between the model and one such local histogram is small, then the Histogram Intersection between the model and the other local histograms will also be small. For any two *focus regions* S and T, the authors show that the Histogram Intersection measures α_S and α_T with the model histogram M are related by the inequality

$$\alpha_T \leq \frac{\min(|S \cap T|, \alpha_S|S|) + |T \Leftrightarrow S|}{|T|}.$$

The goal of the authors' *Active Search* algorithm is to output image regions that have a Histogram Intersection with the model of at least some threshold θ. The Histogram Intersection for an image region T need not be computed if the above inequality proves that it must be less than θ using the result of a previous Histogram Intersection with a region S. See [82] for the details of Active Search. In [83], the authors report that only on the order of 0.5% of possible focus regions require a Histogram Intersection computation. Although this is an excellent pruning rate, it can still leave thousands of (scale,position) pairs to be checked in the three-dimensional scale-position search space, especially if very small scale model appearances must be found.

None of the works discussed in this section attempt to verify that the positional distribution of colors in an image window are similar to the positional distribution of the model or pattern colors. The algorithms only report image regions that have a similar color histogram to that of the model. In this thesis, we are interested in verifying positional similarity as well as color similarity. This positional verification is part of the last stage of

our matching algorithm which also tries to adjust the pattern scale and location to make the positions match as well as possible.

The first phase of our matching algorithm is to estimate, using only color information, the scale at which the pattern might occur in the image. Although our scale estimation algorithm does not work with a histogram color respresention, its instantiation on histograms amounts to scaling the model histogram until the point at which further increases in scale start to decrease similarity to the global image histogram. The Backprojection algorithm assumes the scale is given, while Local Histogramming and Active Search exhaustively search for the best scale at each position.

Armed with our scale estimate, the second stage of our search stategy seeks to find quickly a very small set of promising positions in the image to check for a positionally consistent match. By "very small", we mean fewer than five image locations. If our representation were histogram-based, then these would be the locations of relative maxima in the backprojected image after being convolved with a mask whose size matches our scale estimate. Although our algorithm uses constant color image regions instead of pixels as its basic units of information, the idea of backprojection is still used to identify image regions which are likely to be part of the pattern if it appears in the image. Our notion of a promising image region is also one in which there is a small distance between the color histogram of the region and the color histogram of the pattern. However, we only examine areas around image regions that have high probability of being part of a pattern occurrence in our search for promising locations (this requires preprocessing before query time). In other words, we only use the centers of image regions with a high confidence color as the centers of regions to be checked as promising, instead of using every pixel in the image as in the previously described works.

An important improvement in our application of Swain and Ballard's backprojection idea is the use of a pattern scale estimate in computing the confidences/probabilities that an image pixel of a particular color is part of a pattern occurrence. The true probability for a color c is the number of image pixels with color c that are part of a pattern occurrence within the image, divided by the total number of image pixels with color c. In [77], Swain and Ballard use the number of model pixels of color c as the numerator of this ratio. This gives the correct probabilities if the model is cut out from the image, but it can give very inaccurate probabilities for more general model inputs.

The most computationally expensive part of our search algorithm is the final stage in which we try to adjust the scale and position at which we believe the pattern occurs within the image. This stage makes use of both the colors and their spatial distribution in the image and in the pattern. The adjustment process is started from each of the promising locations

found in the previous step. The Local Histogramming and Active Search algorithms also adjust scale in an attempt to get a better Histogram Intersection value, but they do so by fixed step region growing which does not consider the underlying data (color or position) to help determine how much to grow.

2.3 From Histogram Intersection to the Earth Mover's Distance

The major problem with histograms and the common bin-to-bin distance measures defined between histograms is the use of arbitrary bin boundaries. Histogram distance or similarity measures usually only match pixels in corresponding bins. This is certainly true of the Histogram Intersection similarity measure described above, as well as a simple L_1 distance between histograms which sums up the absolute difference in bin amounts over all the bins. Two pixels with very similar colors might be placed in adjacent bins if an arbitrary bin boundary is drawn between the locations in color space. Once this is done, bin-to-bin histogram comparison measures will never be able to match these two very similar pixels. The fault here lies both with the representation and the distance measure. Usually histogram bins are defined by a uniform subdivision of the underlying attribute (e.g. color) space axes. With this uniform subdivision, the bins in every histogram for every image cover the exact same region in the attribute space. This makes it easy to define distance measures between two histograms for any two images by simply comparing corresponding bins, but such bin-to-bin distance measures do not account for the arbitrariness of the histogram representation.

The notion that is missing in bin-to-bin histogram comparison measures is the distance between bins. In the color case, a distance between bins specifies how different the colors in one bin are from the colors in another bin. This dissimilarity will be relatively low for bins which cover adjacent regions in the color space. The problem of using regularly spaced bin boundaries can be overcome to a great extent by allowing matching across bins, and penalizing according to the distance between bins. This is exactly what is done in the histogram distance measure

$$D(I,M) = \min_{F=(f_{ij})} \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{\text{bin}}(\text{bin } i \text{ in I}, \text{bin } j \text{ in M}), \tag{2.3}$$

where f_{ij} represents the amount of bin i from the image histogram which is matched to bin j of the model histogram. The distance asks for the minimum cost matching of the mass

in the two histograms. Here we have left out some constraints on $F = (f_{ij})$, including the constraints $\sum_{j=1}^{n} f_{ij} \leq I_i$ and $\sum_{i=1}^{n} f_{ij} \leq M_j$ which state that the amount of mass from one histogram that can be matched to a bin of the other histogram cannot exceed the amount of mass in that bin.

Given the freedom to match across different regions of attribute space, there is no need to define these regions by a regular subdivision. Such a representation does not taylor itself to the data in an image and can be quite inefficient in space. If an image contains no red, then the bins of the histogram that correspond to shades of red will all have zero color mass. From this image's point of view there is no need to consider these attribute space regions in the distance function (2.3). We could compute the histogram of an image using regular subdivisions of the attribute space, throw away the zero bins, and re-number the nonzero bins in sequential order. This makes the image summary more efficient in space, but can be improved further by defining regions in attribute space according to the image data. Suppose, for example, we want regions in attribute space which are summarized by a single attribute value to have diameter less than some threshold. Such a cluster of attribute mass might cross arbitrary bin boundaries even when the bin spacing matches the threshold, thus representing this cluster by more than one entry. Each image can have a different number of clusters which cover different regions in attribute space as efficiently as possible. Thus we can change the bin-based distance measure (2.3) to the cluster-based measure

$$D(I, M) = \min_{F=(f_{ij})} \sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} d_{\text{cluster}}(\text{cluster } i \text{ in } I, \text{cluster } j \text{ in } M), \tag{2.4}$$

Here the image has m clusters, the model has n clusters, and we must restrict $\sum_{j=1}^{n} f_{ij} \leq I_i$ and $\sum_{i=1}^{m} f_{ij} \leq M_j$. The distance measure (2.4) (once the constraints on F have been fully specified, and the proper normalization has been applied) represents the Earth Mover's Distance (EMD) between distributions of mass in an attribute space. A more precise description of the EMD and the distributions on which it operates will be given in chapter 4. See the works of Rubner et al. [69, 68] and Gong et al. [26] which effectively argue for the use of clustering over histogramming in color-based image retrieval. Experiments in [68] show the superiority of the EMD for color-based image retrieval over many bin-to-bin histogram dissimilarity measures, including Histogram Intersection ([77]), and cross-bin measures, including a common quadratic-form distance ([51]).

2.4 The Earth Mover's Distance

The history of the Earth Mover's Distance begins in 1781 when Gaspard Monge posed the following optimization problem.

> When one must transport soil from one location to another, the custom is to give the name *clearing* to the volume of soil that one must transport and the name *filling* ("remblai") to the space that it must occupy after transfer.
>
> Since the cost of the transportation of one molecule is, all other things being equal, proportional to its weight and the interval that it must travel, and consequently the total cost of transportation being proportional to the sum of the products of the molecules each multiplied by the interval traversed; given the shape and position, the clearing and filling, it is not the same for one molecule of the clearing to be moved to one or another spot of the filling. Rather, there is a certain distribution to be made of the molecules from clearing to filling, by which the sum of the products of molecules by intervals travelled will be the least possible, and the cost of the total transportation will be a *minimum*. (Monge in [50], p. 666, as quoted in [60], p. viii)

If $c(x, y)$ denotes the per unit mass cost of transporting material from $x \in A$ to $y \in B$ for equal-volume sets A and B, then Monge's mathematical formulation is to compute

$$\inf_t \int_{x \in A} c(x, t(x))\, dx$$

over all volume preserving maps $t : A \to B$. In this original formulation, $c(x, y)$ is the Euclidean distance between x and y, the objective function is nonlinear in t, and the set of admissible transportations t is nonconvex.

A major development in the *mass transfer problem* (MTP) came in 1942 when Kantorovich reformulated the problem as a linear optimization over a convex set. If A and B are distributions of mass with density functions $w(x)$ and $u(y)$, respectively, then Kantorovich asked to compute

$$\inf_f \int_x \int_y c(x, y) f(x, y)\, dx dy$$

over all probability density functions $f(x, y)$ with fixed marginals $\int_x f(x, y)\, dx = u(y)$ and $\int_y f(x, y)\, dy = w(x)$. Here it is assumed that A and B have equal total mass $\int_x w(x)\, dx = \int_y u(y)\, dy$. In Kantorovich's formulation, the objective function is linear in f, and the set of admissible f's is convex.

When A and B are (finite) discrete distributions of equal total mass, where A has mass w_i at location x_i and B has mass u_j at location y_j, then Kantorovich's formulation becomes

the transportation problem ([32]) from mathematical programming: compute

$$\min_{F=(f_{ij})} \sum_i \sum_j f_{ij} c_{ij}$$

such that

$$f_{ij} \geq 0, \sum_i f_{ij} = u_j, \sum_j f_{ij} = w_i,$$

where $c_{ij} = c(x_i, y_j)$. Here F is a density function on $\{ x_i \} \times \{ y_j \}$ with fixed marginals $w = (w_i)$ and $u = (u_j)$.[2] If we think of transforming A into B, the f_{ij} is the amount of mass at x_i which *flows* to y_j.

In this thesis, we are mainly concerned with the discrete MTP. Readers interested in the continuous MTP, its history, theory, connection with the discrete MTP, modifications, and applications should see the recent two volume work [60],[61] of Rachev and Rüschendorf, as well as Rachev's survey paper [59].

Often a result for the finite, discrete MTP has a corresponding result for the continuous MTP. For example, in section 5.1.1 we prove that if $c(x, y) = ||x \Leftrightarrow y||$ for a vector norm $||\cdot||$ or $c(x, y) = ||x \Leftrightarrow y||_2^2$, then the cost $c(\bar{x}, \bar{y})$ between the centroids $\bar{x} = \sum_i w_i x_i$ and $\bar{y} = \sum_j u_j y_j$ is a lower bound on the EMD between the finite, discrete distributions $\{ (x_i, w_i) \}$ and $\{ (y_j, u_j) \}$ (here we assume $\sum_i w_i = \sum_j u_j = 1$). Virtually the same proof with summations replaced by integrals shows that the distance between the means $\bar{x} = \int_x x\, w(x)\, dx$ and $\bar{y} = \int_y y u(y)\, dy$ is a lower bound on the EMD between the probablity distributions $w(x)$ and $u(y)$. This should not be surprising because an integral is just the limit of a Riemann sum. The proof of a continuous MTP result can follow the summation manipulations done in the proof of the discrete MTP result, but care must be exercised to ensure that limit signs can be interchanged or can "pass though" functions when necessary.

For another example of a property that holds for both the continuous and discrete MTPs, consider the problem of matching a finite, discrete distribution $\{ (y_1, u_1), \ldots, (y_n, u_n) \}$ to a translate $\{ (y_1 + t, u_1), \ldots, (y_n + t, u_n) \}$. In section 6.7.3, we prove that the matching $f_{ij} = \delta_{ij} u_j$ which matches all the mass at y_i to the mass at $y_i + t$ is optimal. The same proof, which uses the previously discussed centroid lower bound, shows that $f(x, y) = \delta(x \Leftrightarrow (y \Leftrightarrow t))\, u(y)$ is an optimal matching between $u(y)$ and $u(y \Leftrightarrow t)$. A consequence, for example, is that the EMD between two uniform normal distributions with means μ_1,μ_2 and equal variances is $||\mu_1 \Leftrightarrow \mu_2||_2^2$ when $c(x, y) = ||x \Leftrightarrow y||_2^2$.

There are not many explicit results in the literature for the EMD between two continuous

[2] The discrete formulation follows from the continuous one by putting $w(x) = \sum_i w_i \delta(x - x_i)$ and $u(y) = \sum_j u_j \delta(y - y_j)$, where $\delta(x)$ is the Dirac delta distribution.

distributions, but there is a nice result for matching two normal distributions. Let $N(\mu, \Sigma)$ denote a normal distribution with mean μ and covariance matrix Σ. If $c(x, y) = ||x \Leftrightarrow y||_2^2$, then (p. 119 in [60],[59],[53],[17])

$$\text{EMD}(N(\mu_1, \Sigma_1), N(\mu_2, \Sigma_2)) = ||\mu_1 \Leftrightarrow \mu_2||_2^2 + \text{tr}(\Sigma_1 + \Sigma_2 \Leftrightarrow 2(\Sigma_2^{1/2}\Sigma_1\Sigma_2^{1/2})^{1/2}). \qquad (2.5)$$

This result is symmetric in Σ_1 and Σ_2 since it can be shown that $\text{tr}((\Sigma_2^{1/2}\Sigma_1\Sigma_2^{1/2})^{1/2}) = \text{tr}((\Sigma_1^{1/2}\Sigma_2\Sigma_1^{1/2})^{1/2})$. For uniform Gaussians in $\mathbf{R}^K$ with $\Sigma_1 = \sigma_1^2 I_K$ and $\Sigma_2 = \sigma_2^2 I_K$, formula (2.5) reduces to

$$\text{EMD}(N(\mu_1, \sigma_1^2 I_K), N(\mu_2, \sigma_2^2 I_K)) = ||\mu_1 \Leftrightarrow \mu_2||_2^2 + K(\sigma_1 \Leftrightarrow \sigma_2)^2,$$

which agrees with our result for the equal-variance case $\sigma_1 = \sigma_2$. The result (2.5) is actually a consequence of a more general theorem which also yields the EMD between uniform distributions over equal-volume ellipsoids (p. 119 in [60],[17]).

Let us now return to the case of discrete distributions with a discussion of the *match distance* between histograms defined by Werman, Peleg, and Rosenfeld in 1985 ([86]). The main contribution of this work is the use of a distance between bins as given in (2.3). The match distance is defined between two histograms $H^0 = (h_i^0)_{i=1}^n$ and $H^1 = (h_i^1)_{i=1}^n$ with equal total bin counts $\sum_i h_i^0 = \sum_i h_i^1$. The *unfolding* of histogram $H = (h_i)_{i=1}^n$ is defined to be the multiset $\text{UF}(H)$ with h_i copies of bin i. For example, the unfolding of the 2D histogram

$$H = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 3 & 0 \end{bmatrix} \qquad \text{is} \qquad \text{UF}(H) = \{\, (1,1), (1,1), (2,3), (3,2), (3,2), (3,2) \,\},$$

where bins are labelled by (row,column) pairs. The cost of a 1-1 match between $UF(H^0)$ and $UF(H^1)$ is the sum of the distances between matching bins. The match distance between H^0 and H^1 is the cost of the minimum cost 1-1 matching between the multisets $UF(H^0)$ and $UF(H^1)$.

The match distance between H^0 and H^1 is, in fact, the Earth Mover's Distance between the distributions $\{$ (bin i, h_i^0) $\}$ and $\{$ (bin i, h_i^1) $\}$:

$$\min_{F=(f_{ij})} \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{\text{bin}}(\text{bin } i, \text{bin } j)$$

such that

$$f_{ij} \geq 0,\ \sum_{i=1}^{n} f_{ij} = h_j^0,\ \sum_{j=1}^{n} f_{ij} = h_i^1.$$

It turns out that a transportation problem with all integer masses has an optimal flow $F = (f_{ij})$ with only integer values ([32]). Such an optimal flow is therefore also an optimal 1-1 matching between $UF(H^0)$ and $UF(H^1)$.

The match distance is used by Peleg, Werman, and Rom [57] as a uniform framework for changing the spatial and gray-level resolution of an image. In this work, images are 2D histograms of photons where the value of bin/pixel (i, j) is its gray level $I(i, j)$. The authors pose the problem of reducing the number of gray levels in an image from $2G + 1$ (labelled $0..2G$) to $G+1$ as an optimization problem that seeks an image I' with gray levels $0..G$ such that the match distance between I and $2I'$ is as small as possible, where the bin distance is the Euclidean distance between pixel locations. This problem is reduced to computing a minimal sum-of-distances pairing of pixels in I with odd gray level. See [57] and [85] for details. Since the minimal pairing computation takes time $O(m^3)$, where m is the number of pixels in I with odd gray level, the authors suggest using a linear time algorithm to find an approximately optimal matching.

Peleg, Werman, and Rom also formulate a change in spatial resolution from N pixels $\{ x_i \}$ in I to N' pixels $\{ y_j \}$ in I' as a match distance problem. The final gray level at pixel y_j is taken to be a linear combination of the pixels in I: $I'(y_j) = \sum_i \alpha_{ij} I(x_i)$. The match distance is used to compute the weights α_{ij} between $N'I$ and NI' (which both have total mass NN'). In other words, one multiset has N' copies of every pixel in I and the other multiset has N copies of every pixel in I'. If the optimal matching is r_{ij}, then $\alpha_{ij} = r_{ij}/N$ so that the gray level range of I' is the same as that of I. The authors point out that their formulation is not restricted to rectangular grids and can be adapted to weight some pixels (for example, those close to an image edge) more heavily than others.

The previously discussed works concentrate on the case when the total mass of the two distributions is equal. In this thesis, we pay close attention to the partial matching case in which one distribution is "heavier" than the other. For example, in section 5.1.2 we use the centroid lower bound between equal-mass distributions to derive a lower bound on the EMD between unequal-mass distributions. In the unequal-mass case, some of the mass in the heavier distribution is not matched to mass in the lighter distribution. In this dissertation, we also focus on the problem of finding an optimal transformation (from a predefined set of transformations) of the mass locations in one distribution so that its EMD to another distribution is minimized. Finally, we note that our focus is on the EMD between distributions of masses located at points. This does not expose the full generality of the

EMD, for the EMD can be used to compare two weighted collections of any objects for which there is a notion of distance of between objects to provide the costs in the transportation problem. We could use the EMD, for example, to compare two distributions of distributions, where the cost between the lower level distributions is itself an Earth Mover's Distance.

The seminal work which developed the EMD for use in content-based image retrieval is Rubner's thesis ([64]). We shall refer to his work throughout this dissertation.

2.5 Matching under Transformation Groups

In measuring visual similarity, it is often important to allow some transformation of locations in feature space before measuring the distance between two collections of attributes. In the color case, for example, changing the lighting of a scene causes some transformation of the underlying pixel colors. The images of the same scene under different illuminations may still look quite similar even though the pixel values may be quite different. Directly comparing histograms with Histogram Intersection or distributions of color mass with the EMD will not capture such visual similarity. Another example of the importance of allowing transformations is when the underlying attribute is (or includes) the image plane position of ink on a page. Two images of the same object from different distances and viewpoints will be visually similar despite differences in scale, orientation, and image location. Directly comparing the position of the ink will not capture this visual similarity.

Although not its intended purpose, the EMD defines a distance between point sets if one considers the mass at each point in a set to be one unit. The EMD between point sets is the minimum of the average distance between corresponding points taken over all one-to-one correspondences between the sets. In the next two subsections, we consider two other widely used distances between point sets along with the methods used to compute these distances under transformation groups.

2.5.1 The Hausdorff Distance

The Hausdorff distance between finite point sets $A = \{ a_1, \ldots, a_m \}$ and $B = \{ b_1, \ldots, b_n \}$ is defined as

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

where

$$h(A, B) = \max_{a \in A} \min_{b \in B} \rho(a, b) \tag{2.6}$$

is the directed Hausdorff distance from A to B, and $\rho(a, b)$ is the distance between points a and b. The directed Hausdorff distance $h(A, B)$ from A to B is small whenever each

point of A is close to some point in B. The directed distance is appropriate when trying to find a model within an image, since we want all the points in the model set to be close to some point in the image set, but not necessarily vice-versa. In this case, we use the directed distance from the model point set to the image point set. The symmetric Hausdorff distance $H(A,B)$ is small when each point in A is close to some point in B, and each point in B is close to some point in A. The Hausdorff distance defines a metric between finite point sets if the underlying point distance ρ is a metric. The Hausdorff distance can be computed trivially in $O(mn)$ time, but with some cleverness ([1]) this time can be improved to $O((m+n)\log(m+n))$.

In contrast to the EMD applied to point sets, the Hausdorff distance does not use one-to-one correspondences between points. In the directed distance computation (2.6), the same point $b \in B$ can be the nearest neighbor to many different points $a \in A$. This many-to-one matching can be quite advantageous for problems involving very large point sets which do not require one-to-one correspondences. To be fair to the EMD, it is more general than the Hausdorff distance in the sense that it can be used to match distributions, of which point sets are a special case.

For a transformation group $\mathcal{G}$, the Hausdorff distance under $\mathcal{G}$ is defined as

$$M_{\mathcal{G}}(A,B) = \min_{g \in \mathcal{G}} H(A, g(B)).$$

In words, we find the transformation $g \in \mathcal{G}$ which matches A and $g(B)$ as closely as possible. The problem of computing the Hausdorff distance under a transformation group has been considered for various transformation groups $\mathcal{G}$ (e.g. translation, Euclidean), with different norms ρ (e.g. L_2 or L_∞), and in different dimensions d (e.g. 1, 2, ≥ 3). An $O(n \log n)$ time algorithm for the translation case in one dimension is given in [63]. In [8], Chew et al. give an algorithm for the translation case with the L_∞ point distance that runs in time $O(n^3 \log^2 n)$ in dimension $d = 2$ in time $O(n^{(4d-2)/2} \log^2 n)$ in dimension $d \geq 3$. For point sets in the plane with the L_2 distance, the Hausdorff distance under translation can be computed in $O(n^3 \log n)$ time ([35]) and the Hausdorff distance under Euclidean transformations (translation plus rotation) can be computed in $O(n^5 \log n)$ time ([9]). In presenting the time bounds, we have assumed that $m = O(n)$.

The Hausdorff matchers of Rucklidge et al. ([36],[70]) are aimed at the practical problem of finding a binary model within a binary image. Here the point set defined by a binary image is the nonnegative integer point set which includes exactly the locations of the pixels which are "on" in the image. In [36], a rotated, translated version of the model can be located, while in [70] an affine transformation of the model is allowed. The underlying distance

measure is the directed Hausdorff distance under Euclidean transformations in [36], and the directed Hausdorff distance under affine transformations in [70]. Actually, these works use a more robust version of the Hausdorff distance which prevents outliers from affecting the distance computation. The partial directed Hausdorff distance from the model point set B to the image point set A is defined as

$$h_K(B, A) = K^{th}_{b \in B} \min_{a \in A} \rho(a, b),$$

where $K^{th}_{b \in B}$ denotes the K^{th} ranked distance of the distances $\min_{a \in A} \rho(a, b)$ $\forall b \in B$, and $0 < K \leq n = |B|$. The user specifies the fraction f, $0 < f \leq 1$ which determines $K = \lfloor fn \rfloor$. In this way, the user does not need to know the number of points in the model. The fraction f allows for a fraction $1 \Leftrightarrow f$ of the points in the model to be outliers (not near any image points).

The Hausdorff matchers described in [36] and [70] both search in a discretized transformation space. The search space is limited to a finite (but usually very large) number of transformations by the user. The transformation space is discretized into a rectangular grid so that moving by one grid unit in transformation space produces only a small change in the transformed model. The authors leverage off the fact that if the Hausdorff distance is large for a particular grid transformation, then it will also be large at grid transformations in a neighborhood of that transformation. This allows some transformations to be eliminated from consideration without explicitly computing the Hausdorff distance. In [36], the search algorithm loops over all grid transformations and skips transformations that can be ruled out by Hausdorff distances evaluated at previously visited grid cells. Here "ruled out" means that it can be proven that a transformation yields a Hausdorff distance which is (1) larger than a user specified threshold if all matches within a given threshold are desired or (2) larger than the best Hausdorff distance seen so far if the user only wants the best match to be reported.

In [70], the affine search space is six-dimensional, so developing efficient search techniques is crucial. A multi-level cell decomposition strategy is used instead of a loop over all grid transformations. The initial space of transformations to search is tiled with rectilinear cells of equal size (i.e. an equal number of grid transformations inside). If a cell can be proven *not* to contain a transformation which needs to be reported to the user, then it is not searched further. Otherwise, the cell is marked as "interesting". After considering all the cells at the current level, each of the interesting ones are subdivided and the process is repeated. This search process is a breadth first search in the tree representing the recursive cell decomposition. In the case when a single match is required (either any match or the

minimum distance match), a best-first search in which we investigate the most promising interesting cells first will help reduce the total search time. See [70] for the measure used to rank interesting cells.

The same idea of a hierarchical search in a discretized transformation space is used again by Rucklidge ([71]) to find a gray level model in a gray level image. In this work, the search accounts for an affine transformation of the geometry of the gray level pattern. The pruning strategies given are applicable to a number of block distance functions such as SSD (sum of squared differences) and MAD (mean absolute deviation), and other functions which use the gray level differences between corresponding pixels of the image and the transformed model. Distance functions which compare the gray level of a transformed model pixel to a neighborhood of the corresponding image pixel can also be accomodated to account for noise and small shape changes which cannot be captured by an affine transformation. The following results were reported: (1) correct identification of a translated version of a 28×70 model within a 640×480 image examining only 1303 cells versus the approximately 250000 possible translations, (2) correct registration of a figure under rigid motion examining about 7.2×10^7 cells compared to the approximately 5×10^9 rigid grid motions, and (3) correct patch-by-patch matching of two affinely-related images examining only 6.8×10^8 cells compared to over 2×10^{13} possible transformations. No running times were given.

The main strength of the works [36], [70], and [71] is that these methods do not miss any good matches and are guaranteed to report the globally optimal match under whatever distance measure (SSD, MAD, etc.) is used. In contrast, our search strategy will move from any point in transformation space to a locally optimal transformation, with no guarantee that this transformation is globally optimal. The key to the correctness and efficiency of our strategy lies in the efficient selection of a few good initial transformations from which to begin our iteration so that it will converge to an optimal or nearly optimal transformation. The key to the correctness of Rucklidge's works is that all transformations are potentially considered, while the key to their efficiency is the ability to prune large areas of the transformation space without explicitly considering individual transformations.

The pruning rates reported in [36], [70], and [71] are quite impressive, but there are still many transformations that must be explicitely considered – probably too many for the application of content-based image retrieval (CBIR) in which hundreds or thousands of pattern problems must be solved within a minute or so. Given our CBIR motivation, it is more important for us to solve *almost all* pattern problems *correctly and very quickly*, rather than to solve *all* pattern problems *correctly but more slowly*. Another main difference in our approaches is the behavior of our approaches in an area around the best match. Even

in the case when only the best match must be reported (as opposed to all matches within a certain threshold), the Rucklidge algorithms must consider many transformations around the best match because around any good match are other good matches which although not globally optimal are near optimal enough to be quite difficult to eliminate without a closer look. In our approach of following a downhill path in transformation space (i.e. a sequence of transformations which always improves the EMD value), getting close to the best match speeds up the convergence of the iteration. If "close" is close enough, the downhill iteration will move from suboptimal but good matches near the optimal match to the optimal one, and it will do so without visiting every good match in the neighborhood of the best one. Finally, we note that we have also extended the EMD for robustness reasons so that only some mass in the lighter distribution needs to be matched to mass in the heavier distribution.

2.5.2 The ICP Iteration

The *iterative closest point* (ICP) iteration ([5]) was developed to register two 3D shapes. Given a 3D "model" shape and a 3D "data" shape, the goal is to find the rotation and translation of the data shape which registers it with part of the model. Here the word *model* is used in the exact opposite of the way it is used in the previous discussion of Hausdorff matchers – the model shape is searched for the data shape pattern. The model shape can be represented as the union of any collection of geometric primitives such as points, line segments, curves, triangles, etc., as long as there is a routine available to compute the point on a primitive which is closest to a given point. The data shape must be represented as (or decomposed into) a point set. The distance from the data shape point set $B = \{ b_1, \ldots, b_n \}$ to the model shape primitive set $A = \{ a_1, \ldots, a_m \}$ is given by

$$D^{\mathrm{ICP}}(B, A) = \sum_{j=1}^{n} \min_{a \in A} d^2(b_j, a) = \sum_{j=1}^{n} \min_{a \in A} \min_{p \in a} ||p \Leftrightarrow b_j||_2^2,$$

where $d(b, a)$ is the L_2 distance from point b to the closest point on the geometric primitive a. If we assume that the model is represented as a 3D point set, then the ICP distance function becomes

$$D^{\mathrm{ICP}}(B, A) = \sum_{j=1}^{n} \min_{a \in A} ||a \Leftrightarrow b_j||_2^2.$$

The ICP distance from B to A is very similar to the Hausdorff distance from B to A, except that D^{ICP} sums up the distances to the nearest neighbors instead of taking the maximum.

The ICP iteration is designed to solve the optimization problem

$$D_{\mathcal{G}}^{\mathrm{ICP}}(B,A) = \min_{g\in\mathcal{G}} D^{\mathrm{ICP}}(g(B),A) = \min_{g\in\mathcal{G}} \sum_{j=1}^{n} \min_{a\in A} ||a \Leftrightarrow g(b_j)||_2^2,$$

where $\mathcal{G} = \mathcal{E}$, the group of Euclidean transformations. The idea here is to alternately compute (1) the distance minimizing transformation for a fixed set of nearest neighbor correspondences, and then (2) the nearest neighbor correspondences for a fixed data shape transformation. More precisely, the two steps are

$$\psi^{(k)}(j) = \arg\min_i ||a_i \Leftrightarrow g^{(k)}(b_j)||_2^2 \quad \forall j = 1,\ldots,n \quad \text{and} \tag{2.7}$$

$$g^{(k+1)} = \arg\min_{g\in\mathcal{G}} \sum_{j=1}^{n} ||a_{\psi^{(k)}(j)} \Leftrightarrow g(b_j)||_2^2. \tag{2.8}$$

The correspondence $\psi^{(k)}$ maps the index of a point in $g^{(k)}(B)$ to the index of the closest point in A. The use of the Euclidean distance squared facilitates the transformation step because the least squares optimization problem (2.8) has a known, closed-form solution for $\mathcal{G} = \mathcal{E}$ (and for many other transformation sets $\mathcal{G}$). It is not difficult to show that the sequence $< D^{\mathrm{ICP}}(g^{(k)}(B),A) >_k$ is a monotonically descreasing, convergent sequence starting with any initial transformation $g^{(0)}$. The least squares registration (2.8) reduces the average distance between corresponding points during each iteration, while the the nearest neighbor correspondences reduce the individual distance from the points in the transformed data shape to corresponding points in the model.

The idea behind the ICP iteration is a very general one. To see this, we shall rewrite the correspondence step as

$$\psi^{(k)} = \arg\min_{\psi\in\Psi} \sum_{j=1}^{n} ||a_{\psi(j)} \Leftrightarrow g^{(k)}(b_j)||_2^2, \tag{2.9}$$

where Ψ is the set of all functions from $[1..n]$ to $[1..m]$. Since the functions in Ψ are not restricted in any way, the solution to (2.9) makes $\psi^{(k)}(j) \in [1..m]$ equal to the index of the point in A which is closest to $g^{(k)}(b_j)$ (as in (2.7)). Computing $D_{\mathcal{G}}^{\mathrm{ICP}}(B,A)$ is an optimization problem over $\Psi \times \mathcal{G}$. The ICP iteration alternates between finding the best $\psi \in \Psi$ for a fixed $g \in \mathcal{G}$, and the best $g \in \mathcal{G}$ for a fixed $\psi \in \Psi$. In this way, a path in $\Psi \times \mathcal{G}$ is traced out for which the ICP objective function always decreases or remains the same. The ICP iteration is an example of the general *alternation strategy* in which the path followed in search space always proceeds downhill along subpaths over which some subset of the search space variables are constant. Mathematically, the alternation idea applies

to any optimization problem $\text{opt}_{V=V_1 \cup V_2 \cup \cdots \cup V_N} f(V_1, V_2, \ldots, V_N)$ for which one can solve $V_i^*(V_1, \ldots, V_{i-1}, V_{i+1}, \ldots, V_N) = \arg\text{opt}_{V_i} f(V_1, V_2, \ldots, V_N)$.

When the objective function to be minimized is bounded below, the *go downhill* strategy employed by the alternation strategy is guaranteed to converge, albeit possibly to only a local (as opposed to global) minimum (an analogous statement holds for the alternation strategy applied to maximization problems). This is the case in the ICP iteration since the distance function to be optimized is nonnegative. Of course, the main drawback of the alternation strategy is that there is no guarantee of convergence to the global optimum.

Our method for computing the EMD under a transformation set applies the alternation strategy to obtain a decreasing, convergent sequence of EMD values. As in the ICP problem of registering shapes, there is a step to determine the best correspondences for a fixed transformation. The correspondence step in matching distributions with the EMD is more complicated than (2.9). In the EMD case, there is a real-valued variable f_{ij} that indicates how much mass at location i in one distribution is matched to location j in the the other distribution. The correspondences $F = (f_{ij})$ must be constrained so that each location does not match more mass than it possesses. In addition to allowing transformations of mass locations, we can also handle some sets of transformations which alter both mass locations and amounts.

The key to the effective use of a correspondence-transformation alternation to solve CBIR matching problems is the selection of a small number of promising transformations from which to start the iteration. The number must be small for efficiency reasons, and one of the initial transformations must be close to the globally optimal one so that one of the sequences converges to an optimal or to a nearly optimal match. In the case when the whole model shape matches the data shape, the ICP authors suggest a method for computing a good initial rotation from eigenshape analyses of the model and data shapes. A promising initial translation in this case lines up the centroids of the two shapes. This is a good initial transformation strategy when all the data in the model shape is matched, but comparing global descriptors such as those mentioned above will not work in the partial matching case when the data shape matches only some (possibly very small) amount of the model. In this case, the authors suggest a dense sampling of the transformation space to produce a set of initial transformations.

Even in the case of partial matching, it may be possible to quickly determine a few very promising places in an image to look for a pattern. Consider the color pattern problem. If, for example, there is a yellow blob in the pattern and yellow appears only in a few places within the image, then one of those few places is likely to contain the pattern if it appears at all within the image. Even if yellow appears in many places within the image, it may

be possible to quickly eliminate most of these places based on the other colors surrounding the yellow. The pattern will not match an image region which has very different colors or very different ratios of color amounts even if the same colors are present. These two simple checks do not use the positions of the colors within in an image region. The number of image regions checked for initial pattern placement can be kept quite low by using only the most distinctive pattern colors with respect to the image to select these regions. By examining the amounts of the most distinctive pattern colors in the pattern and in the image, one can often obtain a very good estimate of the pattern scale. This scale estimate is crucial in the preceeding discussion since the scale determines how much of the image to compare to the pattern.

The basic ideas discussed above are also applicable to the shape pattern problem in which only differences in scale and location, not in orientation, are allowed. The orientation of ink on the page in the shape pattern problem plays the same role as color in the color pattern problem: the orientation does not change under the allowable transformations. Distinctive pattern orientations with respect to the image (i.e. orientations which occur in the image mainly in an image region which contains the pattern) give a lot of leverage in computing a scale estimate and possible locations of the pattern within the image. Even if the rotated versions of the pattern are to be located in the image, distinctive relative orientations may give the needed leverage to determine efficiently a small set of initial similarity transformations. The ideas discussed above are used to select initial transformations in SEDL, which is our CBIR pattern problem system.

2.6 The FOCUS Image Retrieval System

The FOCUS (Fast Object Color-based qUery System) image retrieval system designed by Das et al. ([14]) addresses the color pattern problem in the CBIR context. An important non-technical contribution of the FOCUS work is that it highlights the importance of partial matching in CBIR systems without assumptions about scale, orientation, position, or background. The system operates in two phases. During the first phase, database images which do not contain all the colors present in the query pattern are eliminated from further consideration. The second phase uses spatial color adjacency relationships to filter the phase one results.

The preprocessing step for phase one identifies the peaks in local color (HSV) histograms measured over a uniform grid of rectangular image subregions. These peaks are recorded as the colors present in the image. Local histograms are used instead of a global histogram to help prevent color peaks in the occurrence of a query pattern from being masked or shifted

by peaks from the background colors. Ideally, one of the local histogram subregions should be the image area which is exactly the query occurrence since this would guarantee an image color peak which exactly matches a query color peak. Of course, this cannot be done because the problem under investigation is to find the query. The color peaks for all images in the database are collected into an indexing structure which supports range queries. Each peak is tagged with the image and the cell within the image in which it occurs. The phase one index also includes a frequency table which specifies the number of images that will be returned by a range query (over a fixed size range) for every point in a discretization of the HSV color space.

At phase one query time, the peaks in a global color histogram over the query image are computed. For each query peak, a range query centered at that peak is performed to identify all database images with a similar color. Using the frequency table, the query peaks are processed in order of increasing number of returned images. The image lists for each query peak are joined to form a list of images that have peaks matching every query peak. The result of phase one is this list of images along with a set of peak correspondences for each image on the list.

The preprocessing step for phase two matching builds a *spatial proximity graph* (SPG) that captures the spatial adjacency relationships between color regions in an image. The nodes in an image SPG correspond roughly to (color peak,cell) pairs. The construction process starts with a list of color peaks for each cell. Two color peaks are connected if it is possible that their underlying regions are adjacent in the image; there is an edge between two color peaks if the peaks are in the same cell or if two peaks of the same color are four-neighbors on the cell grid. There is some collapsing of connected nodes of the same color to obtain scale invariance. See [14] for details. The edges in an SPG show all possible pixel level adjacencies that could occur (although the SPG construction does not perform any pixel level processing), but some false adjacencies may be included.

During phase two, the SPG of each image returned in the phase one result is searched for an occurrence of the query SPG as a subgraph. If an image SPG does not contain the query SPG, then it is assumed that the image does not contain the query. The subgraph search is done after each image SPG node label is replaced by the query color label of the matching peak (obtained from the phase one result). Image nodes whose color does not match any query color are eliminated before the search. The reduced image SPG can be checked for the query subgraph in time $O(n^m)$, where n is the size of the query adjacency matrix and m is the maximum number of occurrences of a color label in the reduced image SPG (usually ≤ 3 according to [14]). The distance between a query and database image is given by the sum of HSV distances between query and corresponding image peaks. The

phase two matching removes images from the phase one result list ordered by this distance function.

The main high-level difference between SEDL and FOCUS is that we use absolute position information, while Das et al. use relative position information. One positive consequence of using relative information is that adjacencies of color regions remain the same regardless of the scale, orientation, position, or continuous deformation of a pattern occurence within an image. In the FOCUS work, there is no transformation to estimate in order to determine if the pattern is present. The search for an optimal transformation is replaced by the search for a subgraph in the image SPG. Underlying both searches is the question of which regions can match one another.

In FOCUS, a query region can only match an image region whose color is most similar to that of the query region (and is within some threshold). Deciding which color in the query matches which color in the image makes the subgraph isomorphism problem computationally feasible. Position information is used to determine if a pair of image regions can match a pair of query regions. A pair of adjacent regions cannot match a pair of non-adjacent regions. In contrast, we decide which regions can match by using a continuous distance measure that is a weighted combination of color and position distance between regions. Our notion of a good match is that for each query region there is a nearby image region of a similar color. This translates into a small combined color-position distance. Of course, our choice to use such a distance function forces us to make the very difficult decision of how much to weigh the individual color and position distances.

The corresponding difficulty in FOCUS is the lack of robustness that may result from using the `possibly-adjacent-to` relation – the absence of a single edge in the image SPG can cause the algorithm to conclude that the pattern does not exist within the image. The FOCUS answer to this problem is to err on the side of introducing false adjacencies in favor of missing true adjacencies. Recall that their SPG (spatial proximity graph) captures all *possible* spatial adjacencies between image colors. False adjacencies may be introduced because the graph construction works at the level of cells instead of pixels.

The size of the cell is crucial here. At one extreme, using a single cell equal to the whole image will give many false adjacencies, but results in a very small SPG to match. In this case, the nodes of the SPG are all the colors present in the image and there is an edge between every pair of colors. Also, the likelihood that query color peaks will be masked by background colors increases as the cell size increases. At the other extreme, using cells which are pixels results in no false adjacencies but gives very large SPGs that cannot be matched in times which are acceptable for an image database application. Suppose, for example, there is an orange and blue checkerboard that fills one cell. Then this cell contributes a

blue node connected to an orange node. If we change the cell size to the size of a single checkerboard square, then each checkboard square contributes one blue or orange node and all four-neighbors of opposite colors are connected. The challenge faced by the FOCUS work is to choose the cell size so that there are not too many false adjacencies but so that the sizes of the SPGs are small enough to match SPGs quickly.

In [14], Das et al. report a recall rate of 90% and a precision of 75% after phase two. These results are excellent, although the results obtained directly from their web demo at `http://vis-www.cs.umass.edu/~mdas/color_proj.html` were not as good (08/17/98). In section 7.5.1, we show that SEDL achieves better recall and competitive precision results. FOCUS, however, has a sizeable advantage over SEDL in speed, requiring less than a second on average for a query in a database of 1200 images of product advertisements and nature images. SEDL requires an average of about 40 seconds per query on a size 361 subset of the FOCUS database.

Finally, we note that FOCUS never really verifies the occurrence of the pattern within the image. It simply eliminates images that cannot possibly contain the pattern. The hope is that the pattern is distinctive enough to eliminate almost all images except those that contain the pattern. In this elimination process, FOCUS does *not* use the sizes of uniform color regions or the amounts of colors present. In contrast, SEDL uses this information to estimate the scale and location of the pattern within the image. Another difference is that SEDL can show the user where it believes the pattern is located, while FOCUS cannot do so. Such feedback may be useful if users can change system parameters in an attempt to improve query results. A further discussion of differences between SEDL and FOCUS is given in section 7.5.1.3.