Topology-based Feature Grouping and Matching for Object Detection

Dimitri Lisin Edward Riseman

February 23, 2005

Abstract

This paper presents a method for appearance-based object detection that works well in the presence of significant clutter and scale changes. It uses differential point features extracted using an outlier detection technique borrowed from data mining. The approach is based on the observation that the features at finer scales tend to merge together at the coarser ones, which can be expressed as topological parent-child relationships between features. We can thus view the set of features in an image as a directed acyclic graph consisting of a number of trees. The trees are used as groups of features, or constellations, resulting in a rough segmentation of an image. The constellations in the reference and target images are matched by a three phase procedure. In the first phase the primitive features are matched based solely on their individual similarity. The second phase discards spurious matches using the topological relationships among features. We verify the topological consistency of the matches by reducing the tree matching problem to the problem of finding a maximum clique in the corresponding association graph. The third phase further refines the matches using the geometric arrangement of the primitive features.

1 Introduction

In this paper we address the problem of object detection in greyscale images of cluttered scenes. We present a novel approach, based on matching differential point features across images, which works well in the presence of significant clutter and scale changes. This paper's focus is appearance-based object detection. We assume that no explicit model of the object of interest, such as a CAD model, exists. Instead we are given an image of an object (the reference) and are asked to locate it in an image of a cluttered scene (the target), assuming that the object is indeed present in the scene.

The crux of this work is the observation that multi-scale differential features in an image have topological relationships that can be described by a directed acyclic graph. These relationships are invariant to translation, scaling, and in-plane rotation. They should also be largely preserved under small out-of-plane rotations (up to 20 degrees). We use these relationships as a constraint for feature matching.

Actually, the graph formed by the features in an image is usually not a single tree, but a forest. We can refer to a set of features in the same tree as a constellation. This allows us to match each constellation in the reference image to each constellation in the target, to find the corresponding ones, instead of considering all features at once. Grouping features into constellations is essentially a segmentation procedure, even though a constellation may encompass several nearby objects or occupy a part of a single object.

The problem of matching constellations reduces to finding the isomorphism between the corresponding trees. The constellations in the reference and target images are matched by a three phase procedure. In the first phase the primitive features are matched based solely on their individual similarity. The second phase discards spurious matches using the topological relationships among features. We verify the topological consistency of the matches by reducing the tree matching problem to the problem of finding a maximum clique in the corresponding association graph. The third phase further refines the matches using the geometric arrangement of the primitive features. This is done using a voting scheme based on the Generalized Hough Transform [1]. An example of a constellation matched between a pair of images is shown in Figure 1.

Even though we use a heuristic approach to solve the maximum clique problem, it is still quite expensive. To speed up our system we can first compare the constellations using a cheaper method, namely the Hausdorff distance, which does not require explicit correspondences between the primitive features. We can then only keep top n candidate constellation correspondences for further verification by the topological and geometric constraints.

We have tested our approach using two different feature representations. The first one is a vector of responses of the first and second derivatives, and the second one is a vector of the first and second order differential invariants. Both are taken at three consecutive scales. The second representation is invariant to the in-plane rotation, while the first one is not. We used both to show that our approach is general enough to work with different feature representations.

Figure 1: Matching Trees.

2 Appearance-based Object Detection

2.1 Global Similarity Metrics

Appearance based object recognition methods often use global similarity metrics. The simplest one, is the correlation between the reference and the target image, which is not very robust with respect to just about any change in the object's appearance. Examples of more sophisticated global metrics are eigenspaces [27], [17], support vector machines [18], and histograms of differential image features [25]. These methods typically expect the image being recognized to contain a single object with little or no background, and thus are not well suited for the problem of object detection. They can often be adopted for detection by sliding a window of varying size along the target image and running the recognition procedure for each location and size of the window. This, however, results in a very large search space, and may not be practical for large images or a large object database.

2.2 Local Features

Another class of object recognition and detection methods use local image features [23], [14], [15], [24], [20], [16]. This requires a pre-processing step, in which salient features with well-defined image locations are extracted, followed by a feature matching procedure. In [14], [15], and [20] feature correspondences are established using the generalized Hough transform [1], i. e. using their geometric arrangement in the image. In [24] and [16] sophisticated probabilistic matching procedures are used.

The advantage of such approaches is that matching point features for recognition also results in finding the location of the object, i. e. detection. Mainly for this reason we use local features in this work, as opposed to global techniques. Another important advantage of using point features, is the possibility of using an indexing scheme for efficient retrieval of the nearest neighbors of a query feature [5]. This greatly speeds up the matching process and allows the system to scale up to handle a large object database. We are currently evaluating indexing methods for high-dimensional spaces, e. g. [10], for applicability to our domain. Additionally, point features allow for a possibility of inferring a CAD-like geometric model of an object, in addition to an appearance-based one. Such a model can be useful for 3d reconstruction, pose recovery, and vision-assisted robotics.

It is pointed out in [20] that even an imperfect grouping of features that are likely to belong to the same object can greatly reduce the computation requirements for matching. In this paper we propose to group multi-scale differential image feature using the topological relationships among them to help avoid spurious matches.

3 Feature extraction

3.1 Multi-scale Differential Features

In this work we use Gaussian scale-space representation of images [12]. The isotropic zero-mean Gaussian function with variance σ^2 of a two-dimensional parameter space at a point $\mathbf{x} = [x, y]^T$ is defined as

$$G(\mathbf{x}, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{\mathbf{x}^T\mathbf{x}}{2\sigma^2}}. \tag{1}$$

The Gaussian and its derivatives form a family of basis filters for the scale space. Convolving an image with Gaussians of successively larger σ generates a set of scale planes with decreasing amount of detail. In each scale plane structures of a particular size, roughly corresponding to σ become prominent.

The derivatives of the image are an effective description of the local behavior of a patch of the intensity surface. Image features are often represented using the image derivatives or some combinations thereof. In this paper we use two different feature representations based on image derivatives, which we shall refer to as *raw derivatives* and *invariants* respectively. The first one defines a primitive feature at a pixel x, y and scale σ_i as a vector of Gaussian derivative responses at 3 consecutive scales: $\sigma_{i+1}, \sigma_i, \sigma_{i-1}$. We use the first and second derivatives $(I_x, I_y, I_{xx}, I_{yy}, I_{xy})$, resulting in a 15-element vector [13], [23]. The second representation uses the first and second order differential invariants, similar to [28] and [29]:

$$\text{Gradient Magnitude} \quad : \quad \sqrt{I_x^2 + I_y^2} \tag{2}$$

$$\text{Laplacian} \quad : \quad I_{xx} + I_{yy} \tag{3}$$

$$\text{Isophote} \quad : \quad \frac{I_{xx}I_y^2 + I_{yy}I_x^2 - 2I_xI_yI_{xy}}{(I_x^2 + I_y^2)} \tag{4}$$

$$\text{Flowline} \quad : \quad \frac{I_xI_y(I_{xx} - I_{yy}) + I_{xy}(I_y^2 - I_x^2)}{(I_x^2 + I_y^2)}, \tag{5}$$

which are also computed at 3 consectuve scales. In both represenations the feature vectors are normalized by their magnitudes, which makes them invariant to linear lighting changes. After normalization we use Euclidean distance to compare the features.

The second representation is invariant to in-plane rotation, while the first one is not. However, the raw derivatives carry the local orientation information, which increases the robustness of recognition in tasks when we do not *a priori*

expect to have significant in-plane rotation, such as face recognition. Also, we use both representation to show the generality of our approach.

3.2 Salient Image Points

Both the raw derivatives and the invariants are defined at every image pixel at every scale. However, it is clearly infeasable to use all features at every possible location. Because of that, it is necessary to select a small subset of features that nevertheless preserve enough of the image's appearance for matching. Typically such a subset consists of features computed at *interest points* in the image, also referred to as *salient points*.

Most often salient image points are defined as the local extrema of some function of the image. One example is using corners, or points of high curvature [30]. Also, local maxima of "blobs" (the trace of the Hessian) and the gradient magnitude can be used [23]. Since such functions are combinations of image derivatives they can be computed very fast. Another commonly used saliency definintion is the difference of Gaussians [14], which can also be computed very cheaply.

Harris combined edge and corner detector [6] is a more sophisticated method of interest point detection often used for matching. The salient points in this case are defined as the local maxima of the Harris function:

$$R = \det(A) - \alpha \operatorname{trace}^2(A), \text{ with } A = G(\sigma) \star \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}, \tag{6}$$

where α is a scalar and $G(\sigma)$ is a Gaussian kernel with the standard deviation of σ. Another interesting example of a saliency definition is presented in [26], where a multiscale decomposition of an image is computed using a 1D wavelet at various orientations, and the local maxima of the sum of the wavelet responses are used as salient features.

3.3 Dynamic Saliency

In this work we find the salient points using the method described in [13]. We treat the problem of finding salient features as being equivalent to finding outliers in a high-dimensional data set, and use a method known in the data-mining literature as the Local Outlier Factor [3]. Rather than being some pre-selected function of the image, our definition of saliency also depends on the feature representation and the feature distance metric. For example, points that are salient if we use raw derivatives as our feature representation may not be salient in the case of the invariants. Since our definition of saliency adapts itself to a particular feature representation, we call it *dynamic saliency*.

Local outlier factor uses the average of the ratios of the local density of the set at a data point p to the densities at its neighbors to assign a degree of being an outlier to p. It is pointed out in [13] that the blurring of the derivatives causes the space of all features in the image to be smooth, which is especially evident at coarser scales. The smoothness property of the feature space allows us to treat a feature's neighbors in the image as its neighbors in the feature space, which makes finding salient points a linear-time algorithm in the size of the scale volume. This definition of saliency is closely related to the one described in [28], which also uses the density of the feature space. It also follows the reasoning similar to that of [9], which defines saliency in terms of the local complexity of the image. However, in [9], the features are defined as image patches of varying size, rather than points.

4 Topology of Multi-scale Feature Sets

We can see intuitively that multi-scale features form a hierarchy. As detail is removed at coarser scales, features that existed at the finer scales either disappear or merge together. This notion has been formalized for the case of blobs by Lindeberg in [11]. He describes a representation called the scale-space blob tree which is used to analyze the spacial extent of blobs, as well as their extent across scales. The representation is applied to edge detection, histogram analysis, and junction classification. Lindeberg also mentions that this representation has a potential to be used for feature grouping and object detection and matching. It is precisely this idea that forms the basis of our paper.

This idea is further developed by Bretzner and Lindeberg in [2], where a hierarchical multi-scale model is used for object tracking. The model consists of blobs and ridges and is created by hand. If a feature is lost during tracking then its topological relationships to other features are used to generate a hypothesis of its location.

One example of utilizing the topology of multi-scale features for object recognition is presented by Shokoufandeh et al. in [26]. In this case a wavelet pyramid of an image is computed using 1D oriented quadrature bandpass filters at 16 different orientations. The filter responses are summed over the orientations, and the resulting values are used as a measure of saliency. The peaks of this saliency map are considered to be salient features, each one assumed to be a circle centered at a peak with the radius defined by its scale.

The hierarchical relationships between features are defined as follows: a feature f is a parent of a feature f' if and only if the scale of f is coarser than that of f', and the center of f' lies within the circle defined by f. Under this definition the appearance of an object is represented as a directed acyclic graph called the Saliency Map Graph

(SMG), with features being the vertices, and the parent-child relationships being the edges. The topology of SMG is clearly invariant to translation and in-plane rotation and is assumed to be largely preserved under small view changes. We use essentially the same structure in our approach as well. However, we define our primitive features differently, and we use a different matching procedure.

The authors assume that the object has roughly the same size in the reference and the target images, which means the scale plane σ in the reference correspond to the scale plane σ in the target. The features in the corresponding planes are matched using the maximum weight bipartite matching, while taking into account their topological relationships in the SMG.

The main disadvantage of this approach is that it hinges on the correspondence between scale planes, and thus cannot handle scale changes. Also it has been demonstrated to work on images with little or no clutter, and it is not clear whether it will work for object detection in cluttered scenes. In our approach we do not assume the tree level correspondence, and, therefore, we are forced to use a more sophisticated tree matching method described below.

5 Feature Grouping

We now need to establish a correspondence between R and T. In [13] this is done using a feature similarity measure and a self-consistency constraint. Feature $r \in R$ is matched to feature $t \in T$ if r and t are mutually maximally similar. This procedure works well when the object in the target image is on a reasonably uniform background. In the presence of clutter, however, the chances of similar salient features existing at multiple locations in the image are high. Because of that a significant number of the correspondences obtained this way is likely to be wrong.

A reasonable strategy to deal with this problem is to group the features in an image in some meaningful way into smaller sets, often referred to as *compound features* or *feature constellations*. We can then assume that if two features in the reference are in the same constellation, then the same must be true about their counterparts in the target. This way we can look for correspondences between constellations, which would help us avoid some of the spurious matches between primitive features. In fact, Piater reports in [23] that using compound features has significantly increased matching accuracy.

As we mentioned in section 4, our method is based on the idea that the multi-scale features form a directed acyclic graph. We now build a structure identical to the Saliency Map Graph [26], but using a different type of salient features. We treat a feature at scale σ as a circle of radius σ. Let f be a feature at scale σ_i. Let σ_{i+1} be the

scale of the next coarser level. Feature f' at scale σ_{i+1} is the parent of f if and only if the center of f is within the circle defined by f', and the center of f' is the closest to the center of f. We then identify connected components of this graph, which, of course, are trees. We use these trees as compound features, or feature constellations.

We also introduce the notion of the saliency of a constellation, defined as the mean of the saliency values of its constituent primitive features. We can then reduce the computational requirements of our system by discarding constellations whose saliency is lower than some threshold.

6 Feature Matching

6.1 Matching Constellations

Let $R_1, ..., R_n$ be the constellations from R. Let $T_1, ..., T_m$ be the constellations from T. We match R_i to T_j if T_j is the most similar to R_i. In this case the self-consistency constraint is not applicable, since the correspondence between constellation is not necessarily 1-to-1. It may be the case, especially due to scale change, that several trees in the reference correspond to a single tree in the target, or vice versa. We will define $similarity(R_i, T_j)$, the measure of similarity between two constellations, later in this section. After the constellation are matched, we utilize the geometric relationships between their constituent primitive features in a voting scheme to discard spurious matches.

6.2 Candidate Feature Correspondences

To compare two constellations we first establish a set of candidate correspondences between features in R_i and T_j. For every feature $r \in R_i$ we find a feature $t \in T_j$ that minimizes $||r - t||$. Again we do not use the self-consistency constraint described in [13] because the topological constarint applied later (Section 6.3) to discard spurious matches seems to be more effective. This set of candidate correspondences can also be a starting point for an alternative method for matching constellation using the Hausdorff distance described in Section 6.5.

6.3 Topological Constraint

In our approach we have a somewhat original way to deal with the time complexity of tree matching. We simply avoid doing it. Instead, we take the set of matches computed in the previous step and attempt to find its largest subset that is consistent with respect the topological relationships among features. In essence, we only need to *verify* the topological constraints, which can be done in polynomial time. This is done by using a *thresholded association*

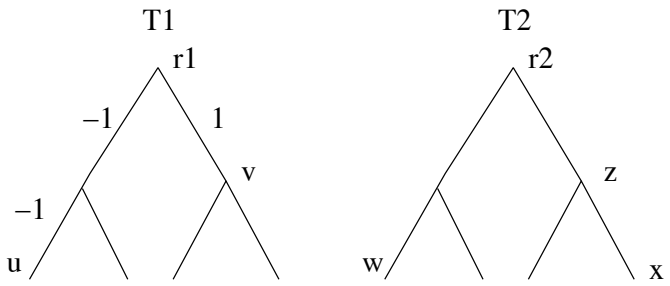


Figure 2: Path string.

graph, based on the *association graph* from [22] (Section 6.3.1).

6.3.1 Tree Matching

The hierarchical structure of multi-scale features can be described by a graph, which is a forest, a set of connected components, that are trees. Thus the problems of object recognition and detection reduce to the problem of tree matching. To simplify the problem of tree matching various heuristics and constraints are often used. For example, as we mentioned earlier, the approach described in [26], which essentially performs tree matching, assumes the correspondence of tree levels, reducing the complexity to polynomial. A more general approach to tree matching is presented in [22]. It reduces the problem of tree matching to an equivalent problem of finding the maximum clique in an association graph, described below.

The approach performs the reduction by defining an *association graph* [22], whose set of vertices is a Cartesian product of the sets of nodes of the two trees being matched. Two vertices in the graph are connected by an edge if their constituent nodes from the first tree have the same relative topology as those from the second tree. The relative topology is defined by the path from one node to the other in the tree.

Formally, [22] defines a notion of a *path string*:

Let u and v be two distinct nodes of a rooted tree T, and let $u = x_0 x_1 ... x_n = v$ be the (unique) path joining them. The *path string* of u and v, denoted by str(u,v), is the string $s_1 s_2 ... s_n$ on the alphabet {-1,+1} where for all $i = 1...n, s_i = level(x_1) - level(x_{i-1})$.

In a rooted tree there is a unique path connecting any two nodes. The path string encodes this path as a sequence of elementary operations required to reach node v from u. Intuitively it corresponds to the degree of relationship between two relatives in a "family" tree [22]. For example in Figure 2 in tree T1 $str(u, v)$ = "-1,-1,1", which is the same as $str(w, z)$ in T2, while $str(w, x)$ = "-1,-1,1,1".

The path string is used to define the *association graph*:

The *association graph* of two rooted trees $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$ is the graph $G = (V, E)$ where

$$V = V_1 \times V_2 \tag{7}$$

and, for any two nodes (u,w) and (v,z) in V, we have

$$(u, w) \sim (v, z) \Leftrightarrow str(u, v) = str(w, z). \tag{8}$$

In other words, every vertex in the association graph corresponds to a pair of nodes, one from each tree. An edge between two vertices signifies that the corresponding nodes from T_1 and T_2 have the same relationships in their respective trees. For example, the association graph resuling from the trees in Figure 2 would have an edge between (u, w) and (v, z), but not between (u, w) and (v, x).

The authors then prove that a maximal (maximum) subtree isomorphism between T_1 and T_2 induces a maximal (maximum) clique in the corresponding association graph G.

6.3.2 Kinship: relationship of nodes in a tree.

We describe the relationship between two nodes in a tree in a way which is similar to a path string, but which is much simpler. Let u and v in a tree G. Let a be the closest common ancestor of both u and v. $kinship(u, v)$ is then defined as an ordered pair (x, y), where x is the distance (number of levels) between u and a, and y is the distance between v and a. Let u, v be a pair of nodes in tree G_1 and u', v' be a pair of nodes in tree G_2. Let $kinship(u, v) = (x, y)$ and $kinship(u', v') = (x', y')$. We can now compare the topological relationship between u and v to that of u' and v' as follows:

$$kinship(u, v) - kinship(u', v') = \sqrt{(x - x')^2 + (y - y')^2} \tag{9}$$

Let u and v have the same topological relationship as u' and v' if

$$kinship(u, v) - kinship(u', v') < t, \tag{10}$$

where t is some threshold.

Besides being simpler than the path string, this definition provides additional flexibility. We can now compare the kinships by computing the Euclidean distance between them. If the two kinships are identical this distance will be equal to 0. If not, it will increase gracefully as the kinships become increasingly different. For example if u is the parent of v and u' is a the grandparent of v' then $kinship(u, v) - kinship(u', v') = 1$.

6.3.3 Thresholded association graph.

The *thresholded association graph* of two directed acyclic graphs $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$ is the graph $G = (V, E)$ where

$$V = V_1 \times V_2 \tag{11}$$

and, for any two nodes (u,w) and (v,z) in V, we have

$$(u, w) \sim (v, z) \Leftrightarrow kinship(u, v) - kinship(w, z) < t, \tag{12}$$

where t is some threshold.

If $t = 0$, meaning that the pairs of nodes in two trees must have identical kinships to yield an edge in the graph, then our graph is identical to the association graph from [22]. However, we have determined experimentally that a threshold of 0 is too restrictive and does not work well in the presence of clutter, scale change, and 3D rotation. In our implementation we use a threshold of 1.75.

We now use a heuristic approach to find the maximum clique in this graph. We simply keep removing the vertex with the smallest degree from the graph, until the remaining subraph is fully connected. This approach does not guarantee that the resulting clique is the maximum, but it works well enough for our purposes.

An example of using this method of tree matching is shown in Figure 3. The large tree in the background is the constellation from the target image, with the relevant portion enlarged on the right. The target image itself is the first image of Figure 4, with the features of the constellation in question shown as bright circles. The small tree on the left is the constellation from the reference (the third image in Figure 4). The matched nodes in the trees are numbered and are represented as circles. We can see that nodes 1, 2, 6, 4 and 9 have exactly the same relationship in the reference and the target. On the other hand node 3 is the parent of node 8 in the reference, while in the target 3 is the grandparent of 8. Also in the reference node 5 is the parent of 7, while in the target nodes 5 and 7 are siblings. This shows that our thresholded association graph approach with a relaxed kinship constraint is robust with respect to small perturbation of the trees being matched. The last two images of Figure 4 show an enlarged view of some of the matched features in the target and the reference, respectively.

6.4 Similarity between Constellations

Finally we can define the similarity measure for constellations. Let R_i and T_j be the two constellations. Recall that we have obtained a set of primitive feature matches from R_i to T_j using the topological constraint described above.

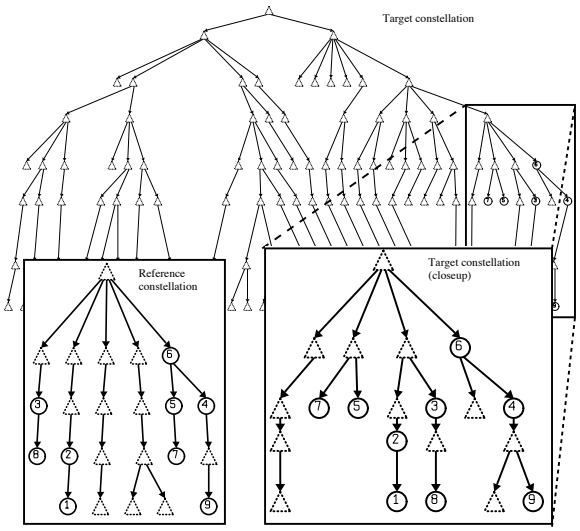
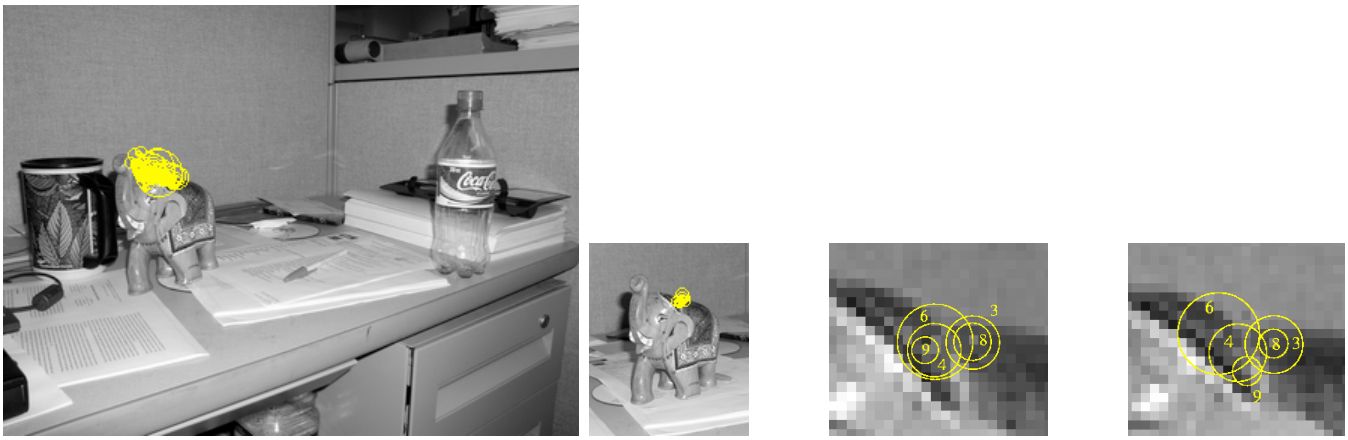


Figure 3: Tree Matching.



Target Reference Target (enlarged) Reference (enlarged)

Figure 4: Corresponding trees in reference and target.

Let $f \in R_i$ and $f' \in T_j$ be a pair of matched primitive features. As we mentioned in section 3.1 we use the Eucledian distance to compare two feature vectors, denoted by $d(f, f')$. Let a *match score* between f and f' be $1 - d(f, f')$, to make it a measure of similarity rather than distance. We then compute $similarity(R_i, T_j)$ by summing up the match scores for all primitive feature matches from R_i to T_j. This metric, referred to as the match score of the two constellations, rewards the cases when a large number of features got matched from one constellation to the other. In practice we take top n matches for every constellation $R_i \in R$ into T and then use a geometric verification step discussed in Section 6.6 to decide which one, if any, is indeed correct.

6.5 Comparing Constellations with Hausdorff Distance

Hausdorff distance [7] is a similarity measure between two finite sets of points A and B. It was originally defined as

$$h(A, B) = \max_{a \in A} \min_{b \in B} ||a - b||, \tag{13}$$

where $||a - b||$ is some measure of distance between two given points. The advantage of using this metric is that it lets us compare two sets of points without having to establish explicite correspondences. However, the Hausdorff distance as it was originally defined is extremely sensitive to noise. In fact, even one noisy point may significantly affect the resulting distance. Because of that we can define the *partial Hausdorff distance* [7] as follows:

$$h^k(A, B) = k^{th}_{a \in A} \min_{b \in B} ||a - b||, \tag{14}$$

where $k^{th}_{a \in A}$ refers to the k-th largest value, rather than the maximum. This definition effectively allows for k points to be outliers or noise, and makes the distance much more robust for practical purposes. If $k = .5||A||$, where $||A||$ is the cardinality of set A, then h^k is called the *median Hausdorff distance*.

The Hausdorff distance has been primarily used in computer vision for matching of edge-based object models [7], [4]. It has also been applied in the context of eigen-space methods [8], and it has been reformulated in a probabilistic framework [21].

We can also use a potentially less accurate but a much cheaper way of comparing constellations by utilizing the Hausdorff Distance We can take our set of candidate correspondences and simply use the k-th largest distance between a matched pair of primitive features as the distance between the two constellations, without computing the topological constraint. Using the Hausdorff distance we can still find the top n matches for every constellation

$R_i \in R$ into T. We would still have to compute the topological constraint for those matches, because the geometric verification step (Section 6.6) requires reliable correspondences between primitive features.

6.6 Geometric Verification

We assume that the object of interest is rigid, so that the geometric arrangement of the features is largely preserved from the reference to the target. Our method uses this assumption to identify and discard spurious constellation matches. It employs a scheme similar to that of [14] and [15] based on generalized Hough transform [1]. Lowe in [14] and [15] assumes a rigid transformation from the reference to the target. He then computes the transformation parameters for each triple of the matched features and stores them in a hash table. After that he finds the hash bin with the highest number of entries, and considers the corresponding parameters to be the correct ones. Finally, the matches that do not fit the transformation are discarded.

One disadvantage of this approach is its time complexity. In [14] the 6-parameter affine transformation is used, which means that every triple of features needs to be considered. The complexity is thus $O(n^3)$ in the number of matched features. In [15] the problem is somwhat alleviated by using the 4-parameter similarity transformation, which only requires looking at all pairs of features, reducing the complexity to $O(n^2)$. In our system, the matches are already grouped into a relatively small number of larger constellations, as opposed to the triples or pairs, greatly reducing the time complexity regardless of the transformation chosen. This builds upon the idea of using imperfect feature grouping with Hough transform, outlined in [20].

In this work we use the similarity transformation, which, according to [15] is better suited to handle out-of-plane rotation of the object. The transformation maps a point $[x, y]$ to a point $[u, v]$ in terms of scaling by a factor s, in-plane rotation by an angle θ, and in-plane translation $[t_x, t_y]$:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} s\cos\theta & -s\sin\theta \\ s\sin\theta & s\cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}. \tag{15}$$

Let $m = s\cos\theta$ and $n = s\sin\theta$. Substituting m and n into Equation 15 we get

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} m & -n \\ n & m \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}. \tag{16}$$

Since we need to slove for the parameters we can collect them into a single vector, rewriting the equation as follows [15]:

$$\begin{bmatrix} x & -y & 1 & 0 \\ y & x & 0 & 1 \\ & \vdots & & \end{bmatrix} \begin{bmatrix} m \\ n \\ t_x \\ t_y \end{bmatrix} = \begin{bmatrix} u \\ v \\ \vdots \end{bmatrix}. \tag{17}$$

This describes a single feature match, but any number of additional matches can be incorporated, by adding two more rows to the first and the last matrix for each one. It is clear, that a minimum of two matches are required to solve for the four unknowns.

To obtain the solution we write the system as

$$Ax = b, \tag{18}$$

and solve the corresponding normal equations [15]:

$$x = [A^T A]^{-1} A^T b, \tag{19}$$

which minimize the mean squared distance error between the transformed points $[x, y]$ and their corresponding $[u, v]$.

We thus solve for the transformation parameters for each constellation, and then store them in the hash table. The bin sizes are 10 pixels for t_x and t_y, 30 degrees for θ, and $\sqrt{2}$ for s. Each entry is placed into two adjacent bins in each dimension, as it is done in [14] to avoid the problem of boundary effects. Outliers can be removed by checking for the spacial error between the transformed reference constellation and the target one. A matched constellation is discarded if the mean error is greater than 15 pixels. Instead of choosing the bin with the greatest number of entries, we sum up the "match scores" of the constellations in each bin, and choose the one that yields the largest sum. The computational complexity of this procedure is $O(n)$, where n is the number of all matched primitive features in all matched constellations.

7 Experiments

7.1 Experiment 1

Our first experiment is based on 20 objects from the Columbia Object Image Library (COIL-20) [19]. It uses a target image composed of the front views of all 20 objects (Figure 5). We matched 5 reference images of each object against

Figure 5: COIL-20 target image.

the target. In the first reference image the object has the same orientation as in the target. In each subsequent image the object is rotated to the right at 5 degree increments spanning 20 degrees.

The first part of the experiment compared our results to those achieved by Moghaddam et al. in [16], which describes a statistical approach to object detection. It is a fair comparison because [16] uses object from the COIL-100 image set, which includes all the objects from COIL-20, but contains color images. Mohgaddam's approach uses the Harris corner detector [6] as the interest point operator and a vector of differential invariants as the feature representation. It does not, however, appear to be able to handle scale changes. Because of that, for this part of the experiment the objects have the same size in the reference and target images. The results are summarized in table 1. The first column lists the detection accuracy for Moghaddam's approach. The second one shows the results of our system using the raw derivatives, and the third one shows the accuracy using the invariants.

The second part of the experiment tested our method's ability to handle scale changes, by reducing the target image to 50% of its original size in both dimensions. The detection accuracy under scale change is presented in figure 6. The horizontal axis represents the out-of-plane rotation from 0 to 20 degrees, and the vertical axis represents the accuracy as percentage. We have tested our algorithm using raw derivatives and the invariants as the feature representation, and also using the Hausdorff matching (Section 6.5), which actually somewhat improved the accuracy.

7.2 Experiment 2

In this experiment we detect a wooden elephant on a cluttered desk (Figure 7). We used 3 reference images of the elephant taken at roughly 10 degree intervals in azimuth, the middle one having the same orientation as in the target.

Table 1: Experiment 1. Benchmark against Moghaddam et al. No change in scale.

	Moghaddam approach	Raw Derivatives	Invariants
0 degrees	100%	100%	100%
5 degrees	96%	100%	100%
10 degrees	94%	100%	100%
15 degrees	91%	95%	100%
20 degrees	86%	95%	100%

Figure 6: Experiment 1. Accuracy under scale change.

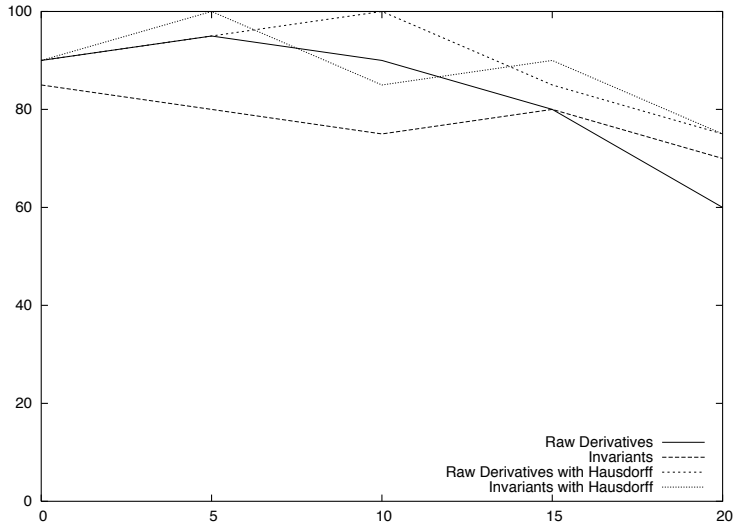


Figure 7: Experiment 2. Target Image.

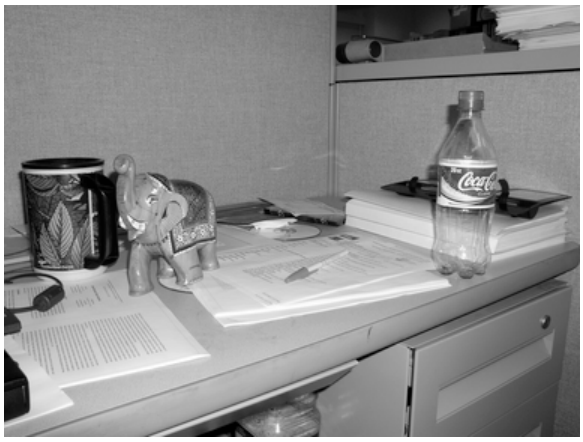
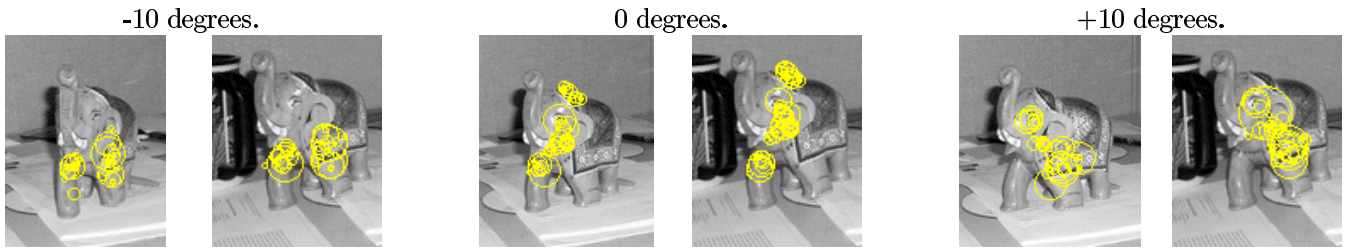


Figure 8: Experiment 2. Detection results.



In Figure 8 each reference image is shown next to a cutout from the target image displaying the matched features as bright cirles.

7.3 Experiment 3

In this experiment we detect a mobile robot in a cluttered lab environment. (Figure 9). Again we used 3 reference images, but taken at approximately 15 degree intervals in azimuth. In Figure 10 each reference image is shown next to a cutout from the target image displaying the matched features as bright cirles. Notice that the robot in the target image is partially occluded by the chair.

Figure 9: Experiment 3. Target Image.

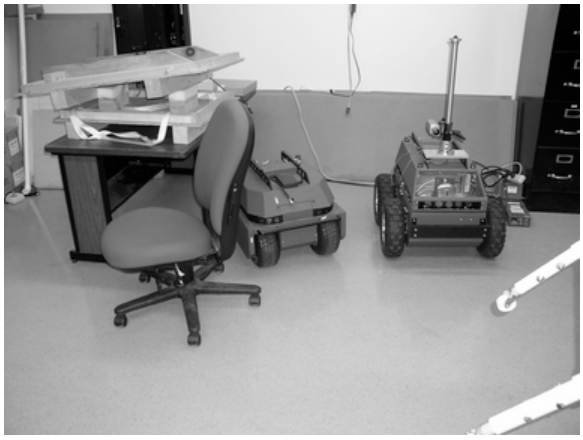
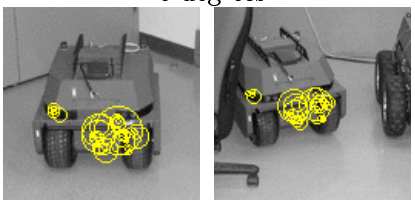


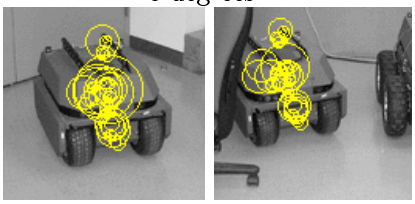
Figure 10: Experiment 3. Detection results.

-15 degrees.



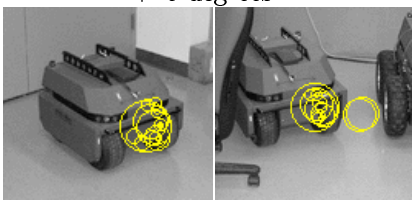
Reference Target

0 degrees.



Reference Target

+15 degrees.



Reference Target

8 Future Work

8.1 Performance Enhancements

In the current implementation or our system we have to compare each feature from the reference image to each feature from the target. This becomes a problem if we want to be able to detect and recognize a large number of different objects. There is extensive literature on indexing of high-dimensional spaces, e. g. [10], which is clearly applicable. For example [5] puts all features from images of several different object into an index structure called k-d-b tree for efficient retrieval of a required number of nearest neighbors of a query feature. It is certainly possible to adopt a similar scheme to our features to make the matching much more efficient.

It is also possible to speed up the entire system by dividing the computation among several processors. The compute-intensive components of the system are embarassingly parallel. Feature extraction can easily be done in parallel at the grain-level of scale planes. To extract the salient features from a scale plane we only require the information from the two adjacent planes. We can therefore divide the scale volume into sets of planes overlapping by one to be processed independently. Only the original image needs to be communicated to the processors to start the computation, and only the resulting subsets of salient features need to be retrieved upon its completion.

Feature matching can also be easily parallelized, especially using an index structure. We can simply have several instances or copies of the indexed feature database so that a set of queries can be prosessed simultaneously. With these modifications the system may be able to achieve real-time performance.

8.2 Probabilistic Learning

Currently, given a reference image of an object our system can locate it in the target image, provided that it is indeed present there. However, it cannot answer more complicated questions, such as "Is an object present in this image?", or "Which of these objects are present and which are not?", or "How many instances of an object are present?".

In order to be able to answer such questions we need to be able to determine the probability of an object being present in the target image. At its present state the system can output a "match score", which is not very informative by itself. We have to start back at the level of the primitive features and be able to map a feature distance to the probability of a feature match being correct. This was done in [23] using the Kolmogorov-Smirnoff distance, but a more sophisticated scheme can be used. For example we could estimate the distribution of the feature distance given wrong or correct matches, and then use the Bayes rule to compute the probability of the match being correct given

the distance.

We then need to propagate this information from the primitive feature matches to the level of constellations, the compound features. In [23] this is handled by a Bayesian network, which seems to be a very appropriate method. A probabilistic framework could provide a more elegant way of integrating individual feature similarity with the topological and geometric relationships among features in a constellation, as well as creating a model of an object from multiple views, as in [23] and [15].

If we convert our system to this probabilistic framework we should also be able to have a learning procedure that would determine which primitive features and which constellation are actually useful for matching. This could compress the representation of the object and increase the robustness of matching.

Acknowledgement

This work was funded in part by a Fellowship from Eastman Kodak Company Research Labs.

References

[1] D.H. Ballard. Generalizing the hough transform to detect arbitrary pattern. *Pattern Recognition*, 13(2), 1981.

[2] L. Bretzner and T. Lindeberg. Qualitative multi-scale feature hierarchies for object tracking. *Journal of Visual Communication and Image Representation*, 11:115–129, 2000.

[3] M. M. Breunig, H.-P. Kriegel, R. Ng, and J. Sander. Lof: Identifying density-based local outliers. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 2002.

[4] Pedro F. Felzenszwalb. Learning models for object recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[5] D. Hall, B. Leibe, and B. Schiele. Saliency of interest points under scale changes. In *Proc. British Machine Vision Conf.*, 2002.

[6] G. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings Alvey Vision Conference*, 1988.

[7] D. Huttenlocher, D. Klanderman, and A. Rucklige. Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, September 1993.

[8] Daniel P. Huttenlocher, Ryan H. Lilien, and Clark F. Olson. View-based recognition using an eigenspace approximation to the hausdorff measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9), 1999.

[9] Timor Kadir and Michael Brady. Saliency, scale and image description. *International Journal of Computer Vision*, 2001.

[10] J. M. Kleinberg. Two algorithms for nearest-neighbor search in high dimensions. In *Proc. 29th Annual ACM Symposium on Theory of Computation*, 1997.

[11] T. Lindeberg. Detecting salient blob-like image structures and their scales with a scale-space primal sketch: A method for focus-of-attention. *International Journal of Computer Vision*, 11(3):283–318, 1993.

[12] T. Lindeberg. Feature detection with automatic scale selection. *Intl. Journal of Computer Vision*, 30(2), 1998.

[13] Dimitri Lisin, Edward Riseman, and Allen Hanson. Extracting salient image features for reliable matching using outlier detection techniques. In *Proceedings International Conference on Vision Systems*, 2003.

[14] David G. Lowe. Object recognition from local scale-invariant features. In *Proc. International Conference on Computer Vision*, 1999.

[15] David G. Lowe. Local feature view clustering for 3d object recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[16] B. Moghaddam, G. Guillamet, and J. Vitria. Local appearance-based models using high-order statistics of image features. In *Proc. IEEE CVPR*, 2003.

[17] B. Moghaddam and A. Penland. Probabilistic visual learning for object representation. In S. K. Nayar and T. Poggio, editors, *Early Visual Learning*. Oxford University Press, 1996.

[18] A. Mohan, C. Papageorgiou, and T. Poggio. Example-based object detection in images by components. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(4), 2001.

[19] S. A. Nene, S. K. Nayar, and H. Murase. Columbia object image library: Coil-20. Technical Report CUCS-006-96, Department of Computer Science, Columbia University, 1996.

[20] C. F. Olson. Improving the generalized hough transform through imperfect grouping. *Image and Vision Computing*, 1998.

[21] Clark F. Olson. A probabilistic formulation for hausdorff matching. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 150–156, 1998.

[22] M. Pelillo, Siddiqi K., and Zucker S. W. Matching hierarchical structures using association graphs. In H. Burkhardt and B. Neumann, editors, *Lecture Notes in Computer Science*, volume 1407. Springer, Berlin, 1998.

[23] Justus Piater. *Visual Feature Learning*. PhD thesis, Department of Computer Science, UMASS Amherst, 2001.

[24] A. Pope and D. Lowe. Learning probabilistic appearance models for object recognition. In S.K. Nayar and T. Poggio, editors, *Early Visual Learning*, pages 67–97. Oxford Universe Press, 1996.

[25] S. Ravela and A. Hanson. On multi-scale differential features for face recognition. In *Vision Interface*, Ottawa, 2001.

[26] A. Shokoufandeh, I. Marsic, and S. Dickinson. View-based object recognition using saliency maps. *Image and Vision Computing*, 17(5-6):445–460, 1999.

[27] M. Turk and A. Pentland. Face recognition using eigenfaces. In *Proceedings IEEE Converence on Vision and Pattern Recognition*, 1991.

[28] K. Walker, T. Cootes, and C. Taylor. Locating salient facial features using image invariants. In *International Workshop on Automatic Face and Gesture Recognition*, 1998.

[29] K. N. Walker, T. F. Cootes, and C. J. Taylor. Locating salient object features. In *Proc. of BMVC*, 1998.

[30] Z. Zhang, R. Deriche, O. Faugeras, and Q.-T. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial Intelligence*, 78(1), 1995.