Fast Search in Transformation Spaces using Orbit Decomposition

Y. Hel-Or

School of Computer Science
The Interdisciplinary Center
Herzeliya, Israel

H. Hel-Or

Dept of Computer Science
University of Haifa
Haifa, Israel

Abstract

Finding appearances of a given pattern in an image at various locations and under various transformations, is often an important step in Image Processing and Computer Vision applications. The process is of very high time complexity since a search must be performed both in the transformation domain and in the spatial domain. In this work we present a new method for fast search in the transformation domain. If the transformation is a group, the set of all transformed patterns forms an orbit. The method is based on the fact that under certain conditions a fast search can be applied by recursively decomposing the pattern orbit into suborbits. This decomposition followed by a rejection scheme, enables the process to quickly reject large percentages of the transformation domain. The suggested technique is presented here using the Euclidean distance, however, it can be applied with any distance metric.

1. Introduction

Pattern Matching is the task of finding patterns in images. This problem may appear in various forms. In the context of this paper we consider the Pattern Matching problem where a pattern may appear under different transformations (e.g. object recognition, motion tracking etc). We refer to this problem as *Generalized Pattern Matching*. To demonstrate the complexity of this problem, we represent a pattern as a point in a high dimensional *pattern space* (e.g. a 10×10 pattern is represented by a point in $\mathcal{R}^{100}$). Consider a given 2D pattern P of size $n = k \times k$ and consider a set of transformations $T(\alpha)$ that may be applied to P, where α is the transformation parameter. For example $T(\alpha)$ might be the set of all 2D rotations about the origin, with α the angle of rotation. Denote by $T(\alpha)P$, the transformation $T(\alpha)$ applied to pattern P. The pattern P and the transformed pattern $T(\alpha)P$ are points in the n-dimensional pattern space. $T(\alpha)P$ for *all* α forms a manifold in the pattern space. If the set of transformations $T(\alpha)$ is a group, then the manifold $T(\alpha)P$ forms an *orbit* in $\mathcal{R}^n$. In general, the number of transformation parameters k is less than n, hence the orbit forms a k-dimensional manifold in n-dimensional space.

We reformulate the generalized pattern matching problem as follows: let W be an image window and $T(\alpha)P$ the orbit of patterns to be matched. In order to evaluate the match between the image window and any pattern in the orbit, the distance between W and the orbit $T(\alpha)P$ must be calculated. Let $d(P,Q)$ be a distance measure between any 2 points in $\mathcal{R}^n$, then the *orbit distance* to be evaluated is:

$$\Delta_\alpha(W, P) = \min_\alpha \{d(W, T(\alpha)P)\} \qquad (1)$$

If the orbit distance is below a given threshold, the window is considered as matching the pattern, otherwise it is a non-match.

Unfortunately, actually calculating the orbit distance is expensive; In many cases, the orbit is highly complex in the sense that two patterns that are close in the transformation domain may be distant in pattern space. This complex behavior of the orbit, in addition to its being non-convex and embedded in a high-dimensional space, makes the calculation of the orbit distance time consuming.

2 Previous Approaches

Several approaches were suggested to deal with such problems. In general, the complexity of search within a complex manifold can be reduced if the manifold has a simpler structure or is of smaller dimension. There are several common techniques implementing this strategy:

Orbit simplification: It is possible to nullify some of the transformation parameters by making the problem invariant to these parameters. Pre-processing of the input data and representing it in a canonical form, is a common strategy in this approach [1]. Nullifying some of the transformation parameters, subsequently, reduces the dimensionality of the manifold and thus simplifies the search problem. Another approach is to find a function defined over the pattern space, that is constant over some transformation parameters and thus invariant to these parameters [6, 5]. For example, applying a dot product between a pattern and a kernel is rotation invariant if the kernel is rotationally symmetric.
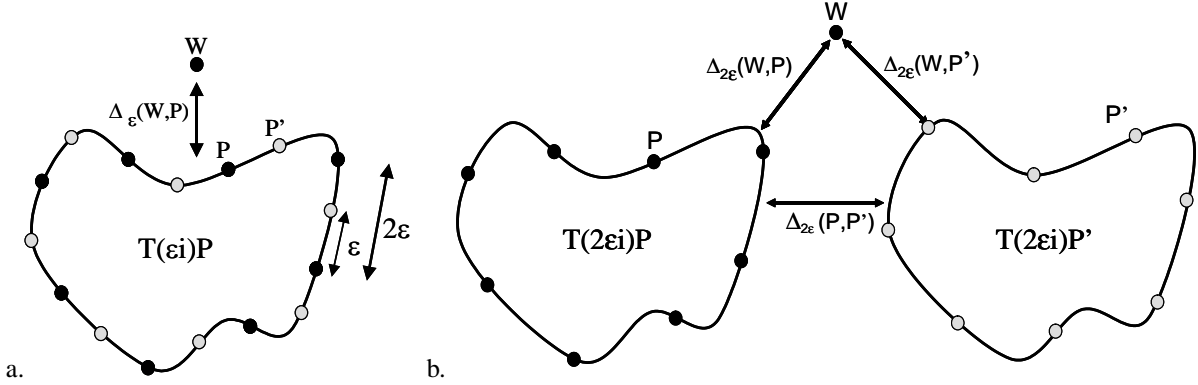
1

Figure 1: a) The continuous transformation group $T(\alpha)$ is uniformly sampled creating a discrete orbit $T(\epsilon i)P$. b) If $\Delta_{2\epsilon}$ is a metric, the triangular inequality can be exploited: if the distance $\Delta_{2\epsilon}(W, P)$ is found to be large and the distance $\Delta_{2\epsilon}(P, P')$ is small, we may deduce that $\Delta_{2\epsilon}(W, P')$ is large as well.

Dimensionality reduction: Another approach attempts to reduce the dimensionality of the pattern space. A popular example is the Wavelets (or DCT) representations where the energy of natural patterns is concentrated in few coefficients, thus, reducing the dimensionality of the pattern space [2, 8]. Another popular approach is to find a reduced linear basis for the pattern manifold using principal component analysis (PCA), and searching the manifold in this reduced space [10].

Fast search: The two previous approaches try to simplify the geometry of the problem. A totally different strategy is to apply an exhaustive search in some of the transformation domains. This may be possible only if a fast search technique is available. For example, it is relatively simple to apply an exhaustive-search in the translation and scale parameters exploited the efficiency of the pyramidal representation and the fast implementation of convolutions.

Although some of the above methods try to reduce the orbit complexity or its dimensionality, their performance is limited due to intrinsic restrictions when using *normed* spaces for representing 2D patterns. In normed spaces, a pattern is represented by a point in a linear space and the distance between two different patterns is defined by the norm of their difference. This paper suggests a new technique that applies a fast search within an orbit of patterns. In addition to the fast performance of this method, it is not limited to normed spaces and can be applied in metric spaces as well. This opens the scope to a large variety of new metric distances that can be designed to simplify the orbit complexity [9].

3 Fast Search in Metric Space

As above, assume $d(Q, S)$ is a distance metric defined between any 2 points in $\mathcal{R}^n$. This measure of similarity be-

tween two patterns may be of any form, linear or non-linear, closed form or algorithmic. The only requirement is that it is a metric. In order to determine whether an image window W is similar to $T(\alpha)P$ for any α, one must estimate the orbit distance $\Delta_\alpha(W, P)$ as defined above (Equation 1). In the general case, an orbit distance Δ_α is not a metric, as it does not satisfy the triangular inequality. However, for a large class of distances d, the orbit distance Δ_α is a metric:

Theorem 1 *If the distance measure $d(Q, S)$ is* transformation invariant, *i.e.*

$$d(Q, S) = d(T(\alpha)Q, T(\alpha)S)$$

then $\Delta_\alpha(W, P)$ is a metric.

(see [3] for proof). Moreover, in such a case where $d(Q, S)$ is transformation invariant, it is easy to show that the point-to-orbit distance is equivalent to the orbit-to-orbit distance:

$$\min_\alpha d(Q, T(\alpha)S) = \min_{\alpha, \beta} d(T(\beta)Q, T(\alpha)S)$$

which is an even stronger result, with respect to pattern matching applications.

In this paper we restrict our approach to distances that are transformation invariant. The metric property of the orbit distance is used to apply fast search within the pattern orbit, by exploiting the triangular inequality.

3.1 The Orbit Tree

The transformation group $T(\alpha)$ is a continuous group since the parameter α forms a continuous domain. In practice, however, the transformation group is approximated by using a discrete group generated by uniformly sampling the parameter domain α. For simplicity, and w.l.o.g, assume $T(\alpha)$ is a one parameter continuous group, and let $\{T(\epsilon i)\}$
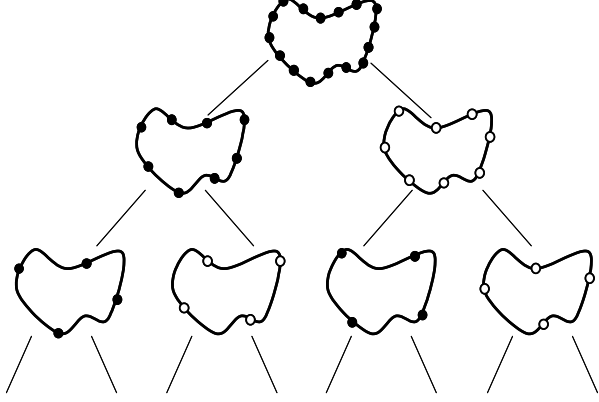
Figure 2: Recursive decomposition of the transformation space, obtained by recursively subdividing the orbit, creates an *Orbit Tree*.

be the discrete group. Using the discrete group, an approximation of $\Delta_\alpha(W, P)$ is then given by

$$\Delta_\epsilon(W, P) = \min_i \{d(W, T(\epsilon i)P)\}$$

$\Delta_\epsilon(W, P)$ can be calculated naively, by computing $d(W, T(\epsilon i)P)$ for all i. However, since distance computation may be time-consuming, run time can be improved, given that Δ_ϵ is a metric:

Consider, again, the orbit $T(i\epsilon)P$. It may be divided into 2 sub-orbits: $T(2\epsilon i)P$ and $T(2\epsilon i)P'$ where $P' = T(\epsilon)P$ (Figure 1a). The distance $\Delta_\epsilon(W, P)$ can then be rewritten:

$$\Delta_\epsilon(W, P) = \min(\Delta_{2\epsilon}(W, P), \Delta_{2\epsilon}(W, P')) \qquad (2)$$

However using the fact that $\Delta_{2\epsilon}$ is a metric, the triangular inequality gives:

$$|\Delta_{2\epsilon}(W, P) - \Delta_{2\epsilon}(P, P')| \leq \Delta_{2\epsilon}(W, P')$$

Note, that $\Delta_{2\epsilon}(P, P')$ can be calculated in advance, prior to the actual search. Thus, if the distance $\Delta_{2\epsilon}(W, P)$ is found to be large and the distance $\Delta_{2\epsilon}(P, P')$ is small, we may deduce that $\Delta_{2\epsilon}(W, P')$ is large as well without any actual distance calculations (Figure 1b).

In terms of pattern matching, this implies that $|\Delta_{2\epsilon}(W, P) - \Delta_{2\epsilon}(P, P')|$ forms a lower bound on all possible values of $d(W, T(2\epsilon i)P')$. If this lower bound is greater than the predefined threshold, these distance measures need not be computed and the patterns associated with this subgroup may be rejected from further computation. Thus, a speed up is obtained by evaluating only half of the distance computations and possibly rejecting 1/2 of the transformation parameters.

This process can be further applied recursively: in order to compute $\Delta_{2\epsilon}(W, P)$, the orbit $\{T(2\epsilon i)P\}$ can be divided into 2 sub-sub-orbits: $\{T(4\epsilon i)P\}$ and $\{T(4\epsilon i)P''\}$ where

$P'' = T(2\epsilon)P$. These orbits can be further sub divided and the process repeated until an orbit is obtained containing a single point. These subdivisions of the original orbit can be described in a tree structure as shown in Figure 2

The Pattern Matching process traverses the tree bottom-up, computing lower and upper bounds on the true distances between image window and sub-orbits of transformed patterns. Branches of the tree are pruned based on the computed bounds. The pruned branches represent distance computations which need not be computed.

3.2 Pattern Matching Algorithm

Given an image window $\mathbf{w}$ and given a pattern $\mathbf{p}$ of the same size. The pattern matching algorithm for determining the similarity between the window and the pattern under a set of possible transformations, proceeds as follows:

Pre-processing: *Create the orbit-tree for pattern $\mathbf{p}$. Compute the inter-orbit distance for each level of the tree.*[1]

Main:
   % Compute the lower (lb) and upper (ub) bounds
   of the root node
   [ROOT.lb,ROOT.ub] := orbitDist(ROOT,$\mathbf{w}$)

   % Evaluate the match
   if ROOT.lb > thresh
      OUTPUT := '$\mathbf{w}$ does NOT match $\mathbf{p}$'
   else
      OUTPUT := '$\mathbf{w}$ matches $\mathbf{p}$'
   endif

Recursive function: [lb,ub] = orbitDist(Node,$\mathbf{w}$)
   % Compute lower and Upper Bound for Node

   % If Node is a leaf of the tree, calculate
   % the actual distance
   if isLeaf(Node)
      d := distance($\mathbf{w}$,Node.pattern)
      Node.lb := d ;   Node.ub := d
      return
   endif

   % Node is internal
   *% **a.** Calculate left son bounds*
   [leftSon(Node).lb,leftSon(Node).ub] :=
         orbitDist(leftSon(Node))

   % if match found, return
   if leftSon(Node).ub < thresh

---

[1] It can be shown that the orbit distance between every sibling nodes on the same tree level ℓ are constant and equals the distance from any point in one orbit to the other orbit: $\Delta_{2^\ell \epsilon}(\mathbf{p}, T(2^{\ell-1}\epsilon)\mathbf{p})$- see [3]
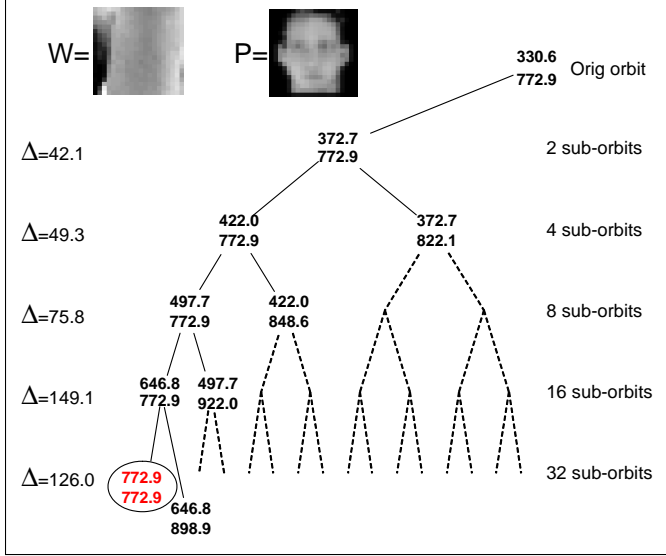
Figure 3: Example of the pattern matching process using the orbit tree. The window **w** was determined to be dissimilar to the pattern **p** after a single distance computation.
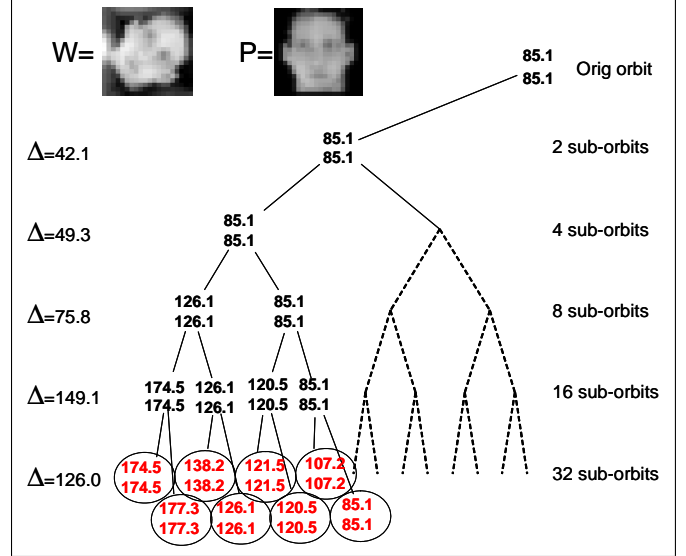

Figure 4: Example of the pattern matching process using the orbit tree. The window **w** was determined to be similar to a rotated version of pattern **p** after 8 distance calculations.

```
    Node.lb := leftSon(Node).lb
    Node.ub := rightSon(Node).ub
    return
endif
```

*% **b.** Estimate right son bounds using*
% the inter-orbit distance
Δ := Node.orbitDist
llb= leftSon(Node).lb ;   lub= leftSon(Node).ub
rightSon(Node).lb := abs(median (llb-Δ,lub-Δ, 0))
rightSon(Node).ub := max(llb+Δ ,lub+Δ)

*% **c.** If estimated bounds are inconclusive,*
% recursively compute right son node.
if (rightSon(Node).lb < thresh) &
          (rightSon(Node).ub > thresh)
  [rightSon(Node).lb,rightSon(Node).ub] :=
          orbitDist(rightSon(Node))
endif

*% **d.** calculate current node's lb and ub*
Node.lb := min(leftSon(Node).lb,rightSon(Node).lb)
Node.ub := min(leftSon(Node).ub,rightSon(Node).ub)

The process always terminates with a matching decision since it can be shown that at leaving every node in the tree, necessarily both lower and upper bounds are either below threshold or above threshold (proof by induction).

The final decision is correct, i.e. if a matching transformed pattern exists at one of the leaves of the tree it will be found since the actual distance will propagate up the tree being the minimum upper bound value. The first leaf (in the sequence of leaves) with distance value below threshold will necessarily be visited since otherwise this implies that one of its ancestor nodes along the branch to the root has had an incorrect lower bound estimated (raising the lower bound above threshold and pruning the subtree which includes the leaf). Since the lower bound estimate is shown to be correct, this should not happen.

The bounds estimated in Step b are explained and proven in [3]. The lower bound estimation can be explained in that subtracting the inter-orbit distance from the sibling's lower and upper bounds, forms a range in which the true distance exists. The lower bound is then the positive value closest to 0 within this range. The expression in Step b calculates this lower bound.

When calculating the bounds of the current node (Step d), the minimum of the upper bounds is taken since the upper bound value bounds the *minimum* distance of W to any of the transformed patterns (Equation 2)

Two examples are shown in Figures 3-4. The pattern matching process was applied to estimate the distance between a 20×20 image window **w** and a pattern **p** of the same size under 2D rotations about the center at a multiple of $360/32$ degrees. In these examples, distances between images were measured using the L_2 norm, and the threshold was set to 100. Figures 3 and 4 show a portion of the traversed orbit tree which contains 32 leaves corresponding to the 32 possible rotations. Values on the tree nodes are the lower bound (top value) and upper bound (bottom value) of the distance between window **w** and the sub-orbit represented at the node. Encircled values at the leaves of the tree denote distances actually computed, all other values were
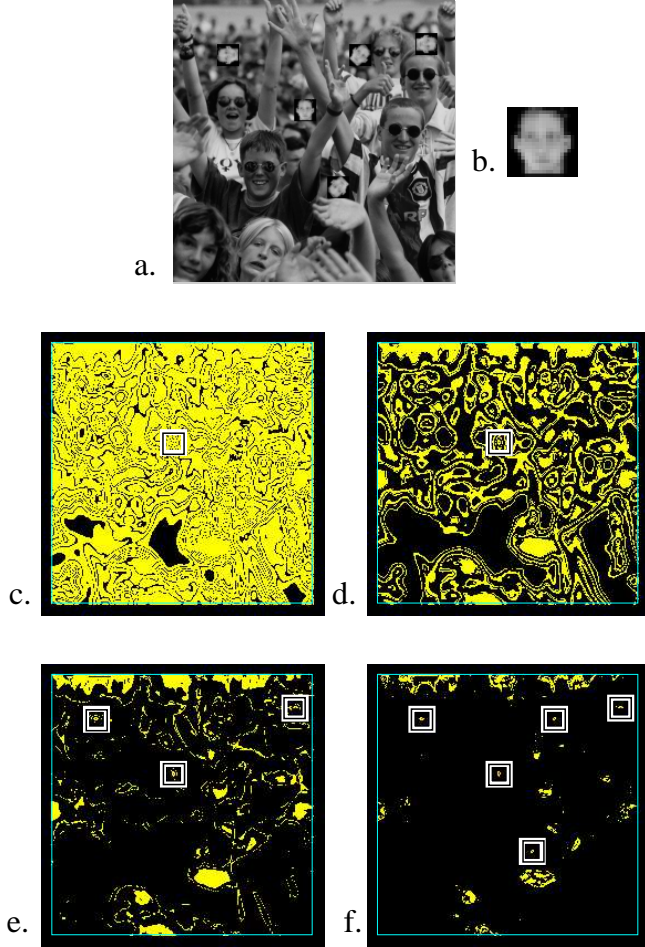
Figure 5: Pattern Matching on a 256×256 image. a) Image. b) Scaled pattern c-f) State of the process after 1, 2, 4 and 8 distance calculations. The percentage of windows that were rejected are 20%, 68%, 91% and 97% respectively.

deduced by propagating the bounds along the tree branches. Dashed edges represent pruned tree branches. The inter-orbit distances, Δ, were pre-computed and are shown on the left for each tree level. The number of sub-orbits at each level is shown on the right. In the example of Figure 3, a single distance evaluation was required to determine that the image window $\mathbf{w}$ is not similar to the pattern $\mathbf{p}$ under any of the possible 2D transformations. Note that the lower bound values, although decreasing when ascending the tree, are always above the threshold.

In the second example shown in Figure 4 the process was applied using the same pattern and set of transformations as above, however, the image window $\mathbf{w}$ is more similar to a rotated version of the pattern $\mathbf{p}$. Due to the low value of the lower bounds, the distance is actually computed for more leaves than in the previous example. However, once the upper bound decreases below the threshold, the process

concludes that the window is similar to the transformed pattern. No more distance computations are required. The upper bound is propagated up the tree to the root, where the process terminates with a positive match decision.

4   Experimental Results

In order to evaluate the performance of the proposed algorithm, the pattern matching scheme was used to search in a large image for a pattern under any 2D rotation.

Figure 5a shows the original 256×256 image. Figure 5b shows a scaled version of the 20×20 pattern. The 2D rotation group was sampled in 32 steps of equal rotation angle. In the original image, several rotated patterns were planted in various locations. All image windows were compared with the pattern under any of the rotation transformations using the proposed scheme. Figure 5c shows the state of the process after a single distance computation per window. For many of the windows this computation was enough to determine the final outcome of pattern matching; black pixels represent those windows for which the process terminated with a negative result, squares represent windows for which the process terminated successfully (i.e. the pattern was found in the window) and yellow pixels represent windows that can not yet be classified and on which the process must continue. Figures 5c-f, show the state of the process for every image window after 1, 2, 4 and 8 distance calculations. The percentage of windows that were rejected are 20%, 68%, 91% and 97% respectively.

Figure 6 plots the percentage of remaining windows for which the process has not yet terminated, as a function of the number of distance calculations performed. Both Figure 5 and Figure 6 show that a very large portion of image windows require very few distance computations. For this example, the average number of distance computations per pixel is 2.868 (compare with 32 computation per pixel using the naive approach).

5   Discussion

In earlier studies, specifically in *Nearest Neighbor Search* problems, the idea of recursively dividing the search space is often used together with the triangular inequality. Analogous schemes, in Generalized Pattern Matching, recursively divide the transformation space [4]. The approach suggested in this paper differs from the previous approaches in the exploitation of the notion of orbits. A naive division of a transformation space into two subsets can impose lower bounds by considering the **maximal** distance between any point in one subset to any point in the other subset. In the approach suggested here, division of the transformation space is unique in that each subset forms an orbit in itself.
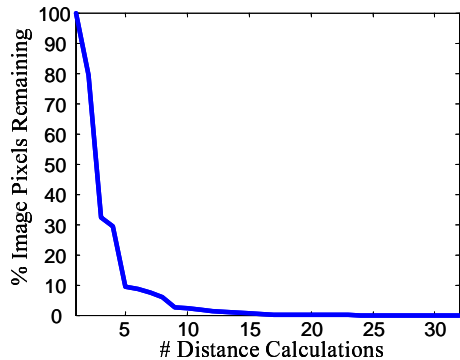
Figure 6: The percentage of remaining windows as a function of the number of distance calculations performed. The average number of distance computations per pixel is 2.868.

The fact that the inter-orbit distance is a metric, allows the calculated lower bounds to be dependent on the **minimal** rather than the **maximal** distance between any point in one subset to any point in the other subset. This tightens the lower bounds and, in turn, increases the rate of pruning of irrelevant possibilities in the search space.

The technique suggested in this paper was presented using the Euclidean distance, however, it is not limited to normed spaces and can be applied in metric spaces as well. This opens the scope to a large variety of new metric distances that can be designed to simplify the orbit complexity and reduce search time [9]. Figure 7 shows an example. A 16×16 image was used as a pattern. The 2D rotation transformation group was sampled at equal rotation angles and applied to the pattern. Figure 7a shows the pattern orbit in Euclidean space. For visualization, the orbit was projected onto the 3 dominant directions (Eigen vectors associated with the three largest Eigen values). The segments connect between orbit points that are associated with consecutive sampled parameter values. It can be seen that the orbit is highly irregular, thus a search within the orbit is not easily simplified and sped up. Figure 7b shows the pattern orbit in a metric space, using the following metric distance: $d(P,Q) = \delta(P,Q) + \delta(Q,P)$ where $\delta(P,Q) \doteq \sum_{x,y} \min_{i,j \in \{-1,0,1\}} [P(x-i,y-j) - Q(x,y)]^2$. In this case, the three dominant directions were calculated using multidimensional scaling [7]. The simple and regular behavior of the metric orbit should reduce pattern matching run time, e.g. by decreasing inter-orbit distance which increases pruning of the orbit tree.

Generalizing the orbit decomposition to multi-parameter transformation groups is straightforward, and can be applied in a similar manner. For a k parameter transformation group a 2^k-ary tree is defined which represents a recursive decomposition of a k dimensional discrete manifold. Recursive decomposition of a non-compact group, however, is
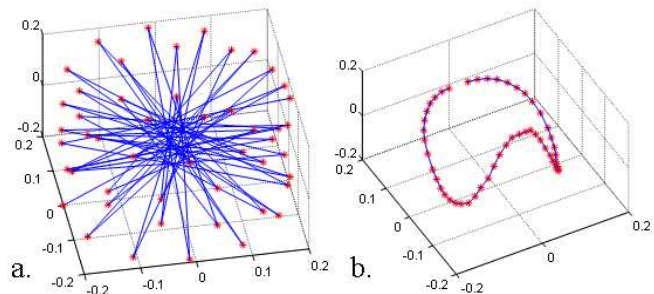

Figure 7: Examples of pattern manifolds (see text). a) Pattern orbit in Euclidean space. b) the same orbit in a metric space.

non trivial. In this case a compact domain of transformations must be defined by bounding the range of the transformation parameters. In order to use the metric characteristics within this bounded range, the search must be performed in a wider range such that the group characteristic is maintained within the original bounded range. For more details the reader is referred to [3].

6 Conclusion

Fast Generalized Pattern Matching was presented, which can be applied when the distance measure is transformation invariant. The technique uses recursive decomposition of the pattern orbit, exploiting the fact that orbit distance is a metric. The suggested method can be applied in metric spaces as well. This allows applications to choose from a wide selection of possible distance metrics.

References

[1] S. Amari. Feature spaces which admit and detect invariant signal transformations. In *Proc. 4th Int. J. Con. Pattern Recognition*, pages 452–456, Kyoto, 1978.

[2] Annonymous. Real time pattern matching using projection kernels. Technical report, 2002.

[3] Annonymous. Fast search in metric spaces using orbit decomposition. Technical report, 2003.

[4] T.M. Breuel. Fast recognition using adaptive subdivisions of transformation space. In *CVPR*, pages 445–451, 1992.

[5] Y. Hel-Or and P. C. Teo. A common framework for steerability, motion estimation, and invariant feature detection. Technical Report STAN-CS-TN-96-28, Stanford University, 1996.

[6] M. Hu. Pattern recognition by moment invariants. *Proc. of the Institute of Radio Engineers (IRE)*, 49:1428, 1961.

[7] B. Kruskal and M. Wish. *Multidimensional Scaling*. Sage Piblications, 1978.

[8] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio. Pedestrian detection using wavelet templates. In *CVPR*, pages 16–20, Puerto Rico, 1997.

[9] S. Santini and R. Jain. Similarity measures. *IEEE T-PAMI*, 21(9):871–883, 1999.

[10] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Neuroscience*, 3(1):71–86, 1991.