Pattern Matching by Sequential Subdivision of Transformation Space

Mingtian Ni and Stephen E. Reichenbach
Department of Computer Science and Engineering
University of Nebraska-Lincoln, Lincoln, NE 68588-0115, USA
mni@cse.unl.edu, reich@cse.unl.edu

Abstract

Pattern matching is a well-known pattern recognition technique. This paper proposes a novel pattern matching algorithm that searches transformation space by sequential subdivision. The algorithm subdivides the transformation space in depth-first manner by conducting boolean operations on the constraint sets that are defined by pairs of template points and target points. For constrained polynomial transformations that have no more than two parameters on each coordinate, a constraint set can be represented as a 2D polygon or a Cartesian product of 2D polygons. Then, the boolean operations can be computed through generic polygon clipping algorithms. Preliminary experiments on randomly generated point patterns show that the algorithm is effective and efficient under practical conditions.

1. Introduction

Pattern matching (or template matching) is one of the well-known approaches for pattern recognition [8]. It has been widely used in visual object location and recognition, data fusion, change detection, *etc.* In pattern matching, a pattern is defined as a set of features and the objective is to establish correspondences from features in a template pattern to features in a target pattern (the unknown pattern).

This paper focuses on point pattern matching, though the proposed algorithm can be easily adjusted for matching other patterns. The most general formulation of point pattern matching problems is as follows: *Given point template (template point pattern) $P = \{p_i(x_i, y_i)\}_{i=1}^{m}$, target point pattern $Q = \{q_j(u_j, v_j)\}_{j=1}^{n}$, and transformation space T, find a transformation t in T that maximizes the number of points in P that can be matched with points in Q under distance tolerance ϵ.* The solution to the problem is either a transformation or a matching. The two forms of solution are usually considered to be equivalent.

Many techniques have been developed for point pattern matching. Based on the search spaces explored, the point pattern matching techniques can be classified roughly as matching space search techniques and transformation space search techniques. Matching space search techniques include pruned tree search (PTS) [1], pruned correspondence search (PCS) [2], *etc.*Transformation space search techniques include relaxation [9], alignment [6], Hough transforms [7], recognition by adaptive subdivision of transformation space (RAST) [3], geometric hashing [11], minimizing Hausdorff distances [5, 10], *etc.*

This paper presents a novel transformation space search algorithm—pattern matching by sequential subdivision of transformation space (PMSST). With this algorithm, each possible pairing between a template point and a target point defines a geometric constraint set in the transformation space. All constraint sets sharing the same template point form a candidate constraint set for the template point. Then, the transformation space is sequentially subdivided by the candidate sets in depth-first manner. During the subdivision, each generated sub-region is the intersection of some constraint sets, i.e., any transformation in the sub-region can match the pairs of points represented by the constraint sets. The algorithm takes a range of numbers of points that are expected to match and returns the sub-region that intersects the maximum number of constraint sets.

The experiments evaluate the effectiveness and computational complexity of the algorithm. Preliminary results on randomly generated point patterns show that the algorithm is effective and its computational complexity is low order polynomial in the size of the point templates under reasonably large transformation space.

2. The Geometric Constraints

Let P, Q, T, and ϵ be as described in the previous section. The objective of pattern matching is to find a $t \in T$ that brings as many p_i's as possible close to some q_j's, i.e., to compute

$$\max_{t \in T, A \subseteq P} \{\|A\| \mid d(t(p_i), q_j) \leq \epsilon, p_i \in A, q_j \in Q\}$$

where $d(.,.)$ is some distance in R^2.

Pairing (p_i, q_j) defines a constraint set T_{ij} in T (a region) under distance tolerance ϵ:

$$T_{ij}(\epsilon) = \{t \in T \mid d(t(p_i), q_j) \leq \epsilon\}.$$

Let C_i be the set of points in Q with which p_i can be matched and T_i be the set of constraint sets that have the same template point p_i:

$$C_i = \{q_j \in Q \mid \exists t \in T, d(t(p_i), q_j) \leq \epsilon\},$$
$$T_i = \{T_{ij} \mid q_j \in C_i\}.$$

Then, solving the pattern matching problem is equivalent to finding the intersection among constraint sets from different T_i's. The computation requires boolean operations (intersection and subtraction) on constraint sets. For general polynomial transformations, the high dimensionality of the transformation space renders the boolean operations inefficient. However, for constrained polynomial transformations that have no more than two parameters on each coordinate, a constraint set can be represented as a 2D polygon or a Cartesian product of 2D polygons. Then, the boolean operations on the constraint sets can be computed through efficient, generic polygon clipping algorithms [4].

This research uses the following transformation model (the combination of non-uniform scale and translation):

$$\begin{cases} u_j = a \cdot x_i + e \\ v_j = d \cdot y_i + f. \end{cases}$$

With Manhattan distance d, the constraint set T_{ij} can be represented as the Cartesian product of 2D polygons T'_{ij} and T''_{ij}:

$$\begin{aligned} T_{ij} &= \{t \in T \mid |a \cdot x_i + e - u_j| \leq \epsilon, |d \cdot y_i + f - v_j| \leq \epsilon\} \\ &= \{(a,e) \in T' \mid |x_i \cdot a + e - u_j| \leq \epsilon\} \\ &\quad \times \{(d,f) \in T'' \mid |y_i \cdot d + f - v_j| \leq \epsilon\} \\ &\triangleq T'_{ij} \times T''_{ij}, \end{aligned}$$

where T' is the projection of T on ae plane, T'' is the projection of T on df plane, and $\times$ denotes Cartesian product of two regions. The relationship between the boolean operations on T_{ij}'s and the boolean operations on T'_{ij}'s (T''_{ij}'s) is as follows:

$$\begin{aligned} T_{i_1j_1} \cap^4 T_{i_2j_2} &= (T'_{i_1j_1} \times T''_{i_1j_1}) \cap^4 (T'_{i_2j_2} \times T''_{i_2j_2}) \\ &= (T'_{i_1j_1} \cap^2 T'_{i_2j_2}) \times (T''_{i_1j_1} \cap^2 T''_{i_2j_2}), \end{aligned}$$

$$\begin{aligned} T_{i_1j_1} -^4 T_{i_2j_2} &= (T'_{i_1j_1} \times T''_{i_1j_1}) -^4 (T'_{i_2j_2} \times T''_{i_2j_2}) \\ &= [(T'_{i_1j_1} -^2 T'_{i_2j_2}) \cup^2 (T'_{i_1j_1} \cap^2 T'_{i_2j_2})] \times \\ &\quad [(T''_{i_1j_1} -^2 T''_{i_2j_2}) \cup^2 (T''_{i_1j_1} \cap^2 T''_{i_2j_2})] \\ &\quad -^4 [(T'_{i_1j_1} \cap^2 T'_{i_2j_2}) \times (T''_{i_1j_1} \cap^2 T''_{i_2j_2})] \\ &= [T'_{i_1j_1} \times (T''_{i_1j_1} -^2 T''_{i_2j_2})] \cup^4 \\ &\quad [(T'_{i_1j_1} -^2 T'_{i_2j_2}) \times (T''_{i_1j_1} \cap^2 T''_{i_2j_2})], \end{aligned}$$

where $\cap^h$, $\cup^h$ and $-^h$ stand for boolean operations in h-dimensional space.

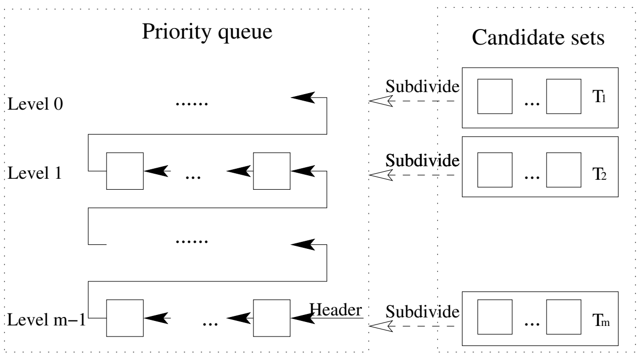

Figure 1. Data structure.

3. The Algorithm

Most pattern matching methods fall into the so-called hypothesis-and-test paradigm [10]. Methods using this paradigm typically involve two phases. Phase one generates a number of hypothetical transformations (or matchings). These hypotheses are checked in phase two to see if they meet some match quality criteria. In PTS [1] and PCS [2], hypothetical matchings are formed in matching space during depth-first search. The feasibility of the matchings are then verified using linear programming techniques. RAST [3] generates hypothetical regions during adaptive subdivision of transformation space. Afterwards, the regions are checked if they contain transformations that match enough points. The PMSST algorithm developed in this paper is closely related to the above three methods. However, PMSST forms hypothetical regions by sequentially subdividing the transformation space using boolean operations on constraint sets. The set of points that the regions can match and the transformations are directly available within the regions. Therefore, the test phase is not needed.

PMSST uses depth-first search. The main data structures are the candidate constraint sets and the priority queue as pictured in Figure 1. For each point p_i in P, its candidate constraint set T_i is calculated as described in Section 2. All such candidate sets are stored in an array $\{T_i\}_{i=1}^{m}$. A region R in the priority queue has three important properties:

- $R.level$: indicates that R is generated by sequentially subdividing T by T_i, $i = 1 \ldots R.level$.
- $R.matchingSize$: the number of constraint sets whose intersection is R. This value gives the number of points that the transformations in R can match.
- $R.css$: The set of template point and target point pairs that the transformations in R can match. Note that the template points in $R.css$ is a subset of $\{p_i\}_{i=1}^{R.level}$.

Regions are enqueued with their priorities. The priorities are compared based on two rules:

- Regions with higher $level$ have higher priorities. This rule forces depth-first search.
- If two regions have the same $level$, the region with larger $matchingSize$ has higher priority. This rule forces best-first search.

When the algorithm starts, the priority queue has the transformation space T as its only node. T is initialized such that $T.level = 0$, $T.matchingSize = 0$, and $T.css = \emptyset$. Each constraint set T_{ij} in the candidate sets is initialized such that $T_{ij}.css = \{(p_i, q_j)\}$. The algorithm stops when the priority queue is empty or when enough number of template points have been matched. Let R be a region extracted from the priority queue and $i = R.level + 1$. Region R is subdivided by candidate set $T_i = \{T_{ij}\}_{j=1}^{m_i}$. The subdivision generates a set of sub-regions $\{R_j\}_{j=1}^{m_i+1}$, where

$$\begin{aligned} R_j &= R \cap T_{ij},\ j = 1, \ldots, m_i \\ &\quad R_j.level = i, \\ &\quad R_j.matchingSize = R.matchingSize + 1, \\ &\quad R_j.css = R.css \cup T_{ij}.css \end{aligned}$$

$$\begin{aligned} R_j &= R - \cup_{k=1}^{m_i} T_{ik},\ j = m_i + 1 \\ &\quad R_j.level = i, \\ &\quad R_j.machingSize = R.matchingSize, \\ &\quad R_j.css = R.css. \end{aligned}$$

With reasonably small ϵ, T_{ij}'s typically do not overlap in T. Therefore, R_j's do not overlap and $\cup_{j=1}^{m_i+1} R_j = R$. This property means that the algorithm typically does not re-explore a region during the depth-first search.

After the subdivision is done, non-empty sub-region R_j is enqueued if $R_j.matchingSize + m - R_j.level$ is no less than the lowest number of matched points that are expected so far. This rule prunes those regions that will not meet the matching criterion.

The inputs to the PMSST algorithm are:

- P: point template.
- Q: target point pattern.
- ϵ: distance tolerance.
- T: transformation space.
- $[k_{min}, k_{max}]$: the range of the number of template points to be matched, where k_{min} specifies the minimum number of matched points that are expected, and the algorithm stops if k_{max} or more points have been matched.

The output of the algorithm is a region that matches the maximum number of points.

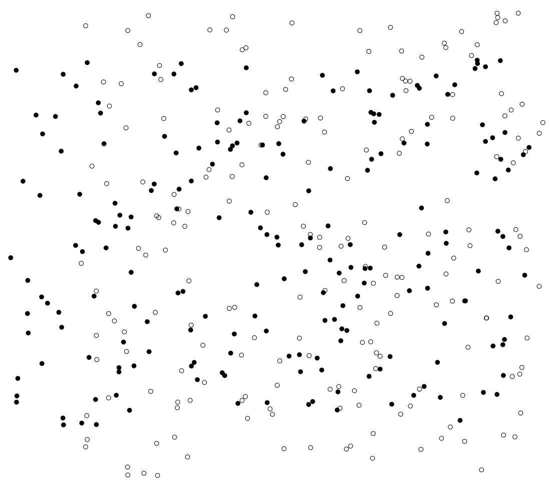

Figure 2. The point template $\overline{P}$ (empty ovals) and the target point pattern Q (filled ovals).

The worst case computational complexity of PMSST is exponential in mn. However, the actual running time of the algorithm depends on many issues: the transformation space, the distance tolerance, the knowledge about the number of matched points that are expected, *etc.* Under practical conditions, the algorithm has computational complexity that is low order polynomial in m, as suggested by the experiments in Section 4.

4. Experimental Results

The experiments evaluate the effectiveness and the computational complexity of the algorithm. The evaluation is based on randomly generated point patterns and noise. The setup of the experiments are as follows:

- point template P: First, a set of 200 points (denoted by $\overline{P}$) are selected independently and uniformly in domain $[-2.0, 2.0] \times [-2.0, 2.0]$. Then, choose P to be a subset of $\overline{P}$. The size of P steps from 10 to 190 with step size 10.
- transformation space T: $\{(a, e, d, f) \mid a \in [0.7, 1.3], e \in [-0.2, 0.2], d \in [0.7, 1.3], f \in [-0.2, 0.2]\}$.
- noise range: $[-0.005, 0.005] \times [-0.005, 0.005]$.
- distance tolerance ϵ: ϵ is set to 0.005 according to the noise range.
- target point pattern Q: First, randomly select a transformation t in T. Then, for each p_i in $\overline{P}$, select a random noise vector ϵ_i uniformly within the noise range and let $Q = \left\{q_i \mid q_i = t(p_i) + \epsilon_i, p_i \in \overline{P}\right\}$. $\overline{P}$ and Q are shown in Figure 2. After that, randomly permutate Q. Finally, for each P, a certain percentage of its

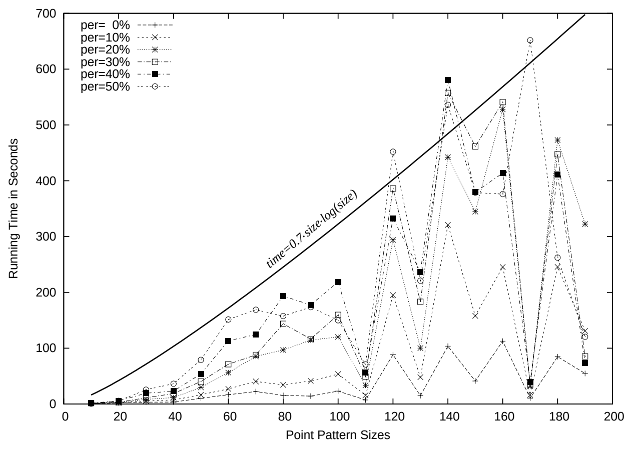

Figure 3. The running time of the algorithm under different percentages of missing points.

transformed points in Q are removed to simulate missing point occasions. The percentage (denoted by per) steps from 0% to 50% with step size 10%.

- $[k_{min}, k_{max}]$: $k_{min} = (1.0 - per - 0.2) \cdot |P|$ and $k_{max} = (1.0 - per) \cdot |P|$ in all occasions, where $|P|$ is the size of P.

The test results are shown in Figure 3. For easy comparison, function $time = 0.7 \cdot size \cdot \log(size)$ is also plotted in the same figure. The results suggest that:

- The expected running time of the algorithm is no worse than $O(m \cdot \log(m))$ under this experimental setting.
- The algorithm degrades gracefully when missing point percentage per increases.
- The algorithm is especially efficient for point patterns of up to 100 points or when per is low.

In all test cases, more than 95% of the template points are matched correctly.

5. Conclusion

This paper develops a novel pattern matching algorithm by sequential subdivision of transformation space. The algorithm represents the constraint sets as 2D polygons or Cartesian products of 2D polygons and then uses efficient polygon clipping algorithms for space subdivision. The subdivision naturally integrates two crucial pruning techniques:

- Any region in transformation space typically is explored only once.
- The regions in the priority queue are expanded in a best-first manner. The more promising regions are likely to be found earlier in the search. These regions are then used to promote minimum expected matching size k_{min} and subsequently to prune the remaining of the priority queue. If a region can not generate better results than what has already been seen, it is pruned. This technique is called the branch-and-bound technique.

Preliminary experiments suggest that PMSST is a practical algorithm.

Because computing boolean operations of polygonal regions in high (three or larger) dimensional space is costly, the PMSST algorithm is difficult to extend to higher dimensional transformation models than those described in Section 2.

Acknowledgment

This research was partially supported by the National Science Foundation, Grant No. 0231746.

References

[1] H. S. Baird. *Model-based Image Matching using Location*. MIT Press, Cambridge, MA, 1985.

[2] T. M. Breuel. Model based recognition using pruned correspondence search. In *Proc. IEEE Computer Vision and Pattern Recognition*, pages 257–262, 1991.

[3] T. M. Breuel. Fast recognition using adaptive subdivisions of transformation space. In *Proc. IEEE Computer Vision and Pattern Recognition*, pages 445–451, 1992.

[4] J. Foley, A. Dam, S. Feiner, and J. Hughes. *Computer Graphics: Principles and Practice in C*. Addison-Wesley, 1995.

[5] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(9):850–863, 1993.

[6] D. P. Huttenlocher and S. Ullman. Object recognition using alignment. In *Proc. International Conference on Computer Vision*, pages 102–111, 1987.

[7] J. Illingworth and J. Kittler. A survey of the Hough transform. *Computer Vision, Graphics and Image Processing*, 44:87–116, 1988.

[8] A. K. Jain, R. P. W. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.

[9] S. Ranade and A. Rosenfeld. Point pattern matching by relaxation. *Pattern Recognition*, 12:269–275, 1980.

[10] W. Rucklidge. *Efficient Visual Recognition Using the Hausdorff Distance*, volume 1173 of *Lecture Notes in Computer Science*. 1996.

[11] H. J. Wolfson and I. Rigoutsos. Geometric hashing: An overview. *IEEE Computational Science and Engineering*, 4(4):10–21, 1997.