Pattern Matching Using the Hausdorff Distance

Fang Yi, Xiong ShengWu

Computer Science and Technology Department

Wuhan University of Technology

Wuhan, Hubei, P.R. China

Abstract Point matching can be a computationally intensive task, especially when large point sets are involved and when the transformation space has many degree of freedom. Here, we employ two efficient algorithms to solve the problem, in an attempt to reduce its computational complexity, while providing acceptable result. The first method is an approximation algorithm based on branch-and-bound approach, it is possible to achieve a tradeoff between the quality of result and the running time. The second method operates within the framework of the first method but accelerate it by using point alignments. We demonstrate the algorithms' performances on synthetically generate data. Moreover, we apply them on finding facial feature points in images and show some preliminary results.

Keywords Feature extraction, Pattern matching, Hausdorff distance, Image registration, Feature space

1. Introduction

In pursuing an image registration task, we are given two images of roughly the same scene, and are asked to determine the transformation that most nearly maps one image into the other. Based on point pattern matching, the problem can be defined abstractly as follows. Given two point sets A and B lying in two different spaces, a space T of transformations mapping one space into the other, a measure of distance between any two point sets, to find the transformation t∈T that minimizes the distance between t(A) and B.

In an attempt to arrive a sound registration scheme, we explore two efficient algorithms that are both accurate and fast. We show the overview of these two algorithms according to the following four factors, proposed by Brown[1] for classifying any image registration method.

1.1 Feature space

Feature space is the domain in which information for matching is extracted. Specifically, we consider feature points that were extracted in the image domain. They may be control points, corners, line segments, etc. All of them are assumed to be available as a result of applying standard feature extraction algorithms. It is important to notice that feature extraction process would yield unexpected result. The first is perturbation errors, which result from a combination of the image digitization process, expansion or shrinkage of objects due to variations in lighting conditions, and the failure of the feature extraction algorithm. The second source of error is the presence of outliers, which can result from many sources, the fact that the two images cover different regions, the presence of occluding objects in images, and the sensitivity of the feature extraction algorithm to variations in lighting, point of view, or other aspects of imaging process.

1.2 Search space

Search space is the class of transformations that establish the correspondence between the sensed image and the reference data. Specifically, we consider two-dimensional affine transformations, allowing for translation, rotation, scaling along each axis, and shearing. Using homothetic coordinates expressing, any linear transformation in the plane can be expressed as affine transformation. Usually, we consider such common subspaces of transformations as translation only, rigid motions (translation, rotation and possibly reflection), homothetic transformations (translation, rotation and uniform scaling). Our algorithm methodology can be applied to any reasonable space of transformations of bounded dimensionality.

1.3 Search strategy

We use two search strategies for finding the optimal transformation. The first is based on a branch-and-bound search of transformation space, and the second combines this with a method based on alignment judiciously chosen candidate feature points. The first method has the advantage that it can provide arbitrarily good guarantees on the accuracy of the final match, and that it naturally uses a priori information to bound the search. The main problem with this method is that the nature of the search leads to rather high running times. The second method is very easy to implement, but it cannot generate results with better than a fixed constant error, and does not lend itself easily to exploiting a priori information in the search.

1.4 Similarity metric

The figure of merit assigned to a match that is determined by a specific transformation is based on the (directional) partial Hausdorff distance[2]. This is a robust measure, it consider the set of distances resulting from taking each point in one set, and finding the nearest point to it in the other set. Rather than taking the sum or the maximum of these distances, which may be affected by outliers, we consider the median or, in general, the k-th smallest distance. More formally, given two point sets A and B, and a parameter k, $1 \leq k \leq |A|$, we define the directed partial Hausdorff distance from A to B to be

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

where K^{th} returns the k-th smallest element of the set, and where d(a,b) is the Euclidean distance from a to b. The parameter k is typically based on a priori bounds on the number of points of A that are expected to have close matches in B under the optimum transformation. These are the inliers[3].

2. The Branch-and-Bound Algorithm

Both of our registration algorithms are based on a branch-and-bound framework, we will describe the branch-and-bound algorithm in this section. For conveniences, we introduce some definitions and notations.

2.1 Definitions and Notations

- A and B, the given ***two point sets***, they are fixed for the remainder of the discussion.
- T, a ***space of transformations***.
- τ, $\tau \in T$, a ***concrete transformation***.
- q, a ***distance quantile***, $0<q \leq 1$, define $H_q(A,B)$ to be $H_k(A,B)$, where k=ceil(q*|A|).
- $sim_q(\tau)$, the ***similarity measure*** of τ, i.e., $sim_q(\tau) = H_q(\tau(A),B)$.
- τ_{opt}, the ***optimum transformation***.
- sim_{opt}, the ***optimum similarity***, i.e. $sim_{opt} = H(\tau_{opt}(A), B) = \min H(\tau(A),B)$.
- M and t, ***parameters for affine transformation***. For any $a = (a_1, a_2) \in A$, $\tau(a) = Ma + t$.
- ε_r, the ***relative error bound***.
- ε_a, the ***absolute error bound***.
- ε_q, the ***quantile error bound***.
- $q' = (1 - \varepsilon_q)q$, the ***weak quantile***. Note that since $q' \leq q$, we have $sim_{q'}(\tau) \leq sim_q(\tau)$, for any $\tau \in T$.
- τ is ***approximately optimal relative*** to ε_r, ε_a and ε_q, if either $sim_{q'}(t) \leq (1+\varepsilon_r)sim_{opt}$ or $sim_{q'}(t) \leq sim_{opt} + \varepsilon_a$ holds.
- ***Tree, node and cell***. We construct a search tree, where each node of the tree is identified with the set of transformations contained in some axis-aligned hyperrectangle in the six-dimensional transformation space. These hyperrectangles are called cells.
- τ_{lo} and τ_{hi}, ***a pair of transformations for each cell***, whose coordinates are the upper and lower bounds on the transformations of the cell.
- T_0, an ***initial cell***. It is assumed to contain the optimum transformation. This is supplied by the user, based on a priori knowledge of the nature of the transformation.
- ***Active cell***, if the cell is a candidate to provide the optimum transformation.
- ***Killed cell***, if it cannot provide the optimum solution.
- ***Cell processing***. Select one of the active cells to process. After processing, a cell is either killed or is split into two disjoint subcells.
- $sim_{hi}(T)$, an ***upper bound of the smallest similarity***, associated transformation of which contained in cell T.
- $sim_{lo}(T)$, an ***lower bound of the smallest similarity***, associated transformation of which contained in cell T.
- sim_{best}, the ***best similarity*** encountered so far in the search.
- τ_{best}, the ***transformation associated***

with sim$_{best}$.

- A ***sim$_{hi}$(T) computing***. We may sample any transformation from within the cell. In particular, this is done by simply taking the midpoint τ of the cell, and then computing $sim_q(\tau)$.
- ***Uncertain region***. Given any cell $T \subset T$, and given any point $a \in A$, consider the image of a under every $\in T$. It is easy to compute a bounding rectangle for this set. We call this bounding rectangle the uncertainty region of a relative to T.
- ***A sim$_{lo}$(T) computing***. To derive our lower bound for T, for each point a $\in A$, we compute the distance from the corresponding uncertainty region to its nearest neighbor in B. Observe that this distance is a lower bound on the distance from (a) to its nearest neighbor in B, for any $\in T$. We then take the q-th smallest such distance. Call this $sim_{lo}(T)$.
- ***The size of uncertain region***, define as its longest side.
- ***The size of a cell***, define as largest size among the associated uncertainty regions for each point in A.
- ***Cell queue***. The active cells are stored in a priority queue, ordered by cell size.
- ***Cell splitting***. Split cell T into two smaller subcells T_1 and T_2, by splitting it along the dimension that contributes most to its uncertainty region size.

2.2 Overview of the Algorithm

Here is an overview of the approximation algorithm. The input consists of the point sets A and B, the Hausdorff quantile q, the approximation parameters ε_r, ε_a and ε_q, and the initial cell T_0.

(1) Build a nearest neighbor data structure for the points of B. Initialize the priority queue to contain T_0. Set $sim_{best}=\infty$. Define the weak quantile q' to be $(1 - \varepsilon_q)q$. Repeat steps (2)-(5) until the priority queue is empty or until $sim_{best} \leq \varepsilon_a$.

(2) Remove the largest cell T from the queue. Compute its lower and upper bounds. This involve the following steps:

  (a) Compute the uncertainty regions for every point $a \in A$ with respect to T.

  (b) For each uncertainty region, compute its nearest neighbor in B.

  (c) Using any fast selection algorithm, compute the q-th quantile among these distances. Call this $sim_{lo}(T)$. If $sim_{lo}(T) > sim_{best}/(1+\varepsilon_r)$ or if $sim_{lo}(T) > sim_{best} - \varepsilon_a$, kill this cell and return to step (2).

  (d) Otherwise, sample a transformation from this cell. Compute the image of each point of A under , and compute the nearest neighbors of these points with respect to B. Find the q'th smallest such distance. Call this $sim_{hi}(T)$.

(3) If $sim_{hi}(T) < sim_{best}$, update sim_{best} and let τ_{best} be the associated transformation.

(4) Split cell T into two smaller subcells T_1 and T_2, by splitting it along the dimension that contributes most to its uncertainty region size. Compute size bounds for T_1 and T_2.

(5) Enqueue T_1 and T_2 in the queue of

active cells.

The final transformation is τ_{best} and its similarity is sim_{best}.

3. Bound Alignment

The branch-and-bound algorithm has many nice features, but its main drawback is its relatively high running time. This occurs especially when high accuracy is required and the optimum similarity is very good. For this reason, we introduce an additional process called bound alignment to help accelerate the search.

Suppose that the search of branch-and-bound has progressed to a stage where most of the uncertainty regions associated with the points of A contain at most one point of B. Consider the points of A that have exactly one point of B in their uncertainty regions. If a cell contains the optimum transformation, we sample one such point at random and compute the unique transformation that maps this point to the corresponding point of B for several times, and there is a good probability that desired optimum transformation is found. On the other hand, if a cell does not contain the optimum, after taking a number of samples and witnessing repeatedly poor similarities, we may regard this as evidence that the cell in question does not contain the optimum and therefore we kill it.

Before giving detailed alignment algorithm, we introduce some definitions and notations.

3.1 Definitions and Notations

- ***Alignment***. The process whereby triples from A are matched against prospective corresponding triples from B in order to determine a candidate transformation is called alignment.
- ***Noise bound*** η. In noisy environments, suppose that for each inlier $a \in A$ there is a point of B that lies within some small distance η from its optimum image point, $\tau_{opt}(a)$. We assume that an upper bound on η, called the noise bound, is provided to the search algorithm.
- ***Alignable uncertain region***. An uncertainty region is said to be alignable if there is at most one point of B in the region, or if the region is empty and there is at least one point of B within distance η of the region.
- ***Alignable cell***. If the current cell has a significant fraction of alignable uncertainty regions, we say that this cell is eligible for alignment.
- q_s, the ***quantile of uncertainty regions*** becoming alignable.
- N_s, the ***number of taking samples***.

3.2 Detailed Algorithm

Here are steps used for the bounded alignment algorithm. (These steps are added after step (2d) in the previous description.) The algorithm is given an expected inlier perturbation η, sampling quantile q_s, and a minimum sample size N_s.

(e) For each $a \in A$, count the number of points of B that lie within a's uncertainty region. If this at most one, and the nearest neighbor is within distance η of the uncertainty region, flag this region as alignable.

(f) If the fraction of alignable uncertainty regions is less than q_s, return to step (2). Otherwise, let A' denote the subset of A such that for each $a \in A'$, there exists at least one point $b \in B$ that either lies inside or within distance η of a's uncertainty region. Repeat the following Ns times:

  (i) Sample (without replacement) triples of points of A', until a triple that is geometrically well-distributed is found.

  (ii) Compute the transformation that aligns each point in the triple with a random point of B in its associated uncertainty region. Compute the similarity of this transformation.

  (iii)If the similarity of this transformation is better than the current best similarity sim_{best}, make it the new best. If the similarity obtained for all of the N_s transformations exceeds the current best by an additive amount of η, kill this cell.

If the similarity obtained for all of the Ns transformations exceeds the current best by an additive amount of η, kill this cell.

4. Experiments

We apply the algorithms on finding Facial Feature Points (FFP) in images. We are given a reference face image, FFP of which is known, and expected to find FFP in sensed face image. To comparing with standard face, the sensed image may have many transformations, such translation, rotation, scaling, etc. Extracting feature point set in the reference image and sensed image, we can use our search algorithm to derive optimum transformation by minimizing Hausdorff distance, and find FFP in sensed image finally.

Fig. 1 gives a standard FFP map as reference image, fig.2 shows a face image and its corresponding feature points that served as candidate FFP, fig. 4 displays a final found facial feature points using optimum transformation computed by the bound alignment algorithm.

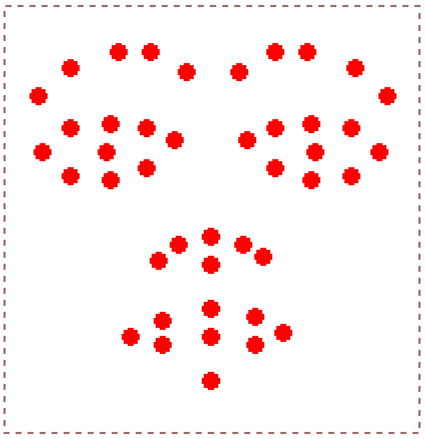

Fig. 1

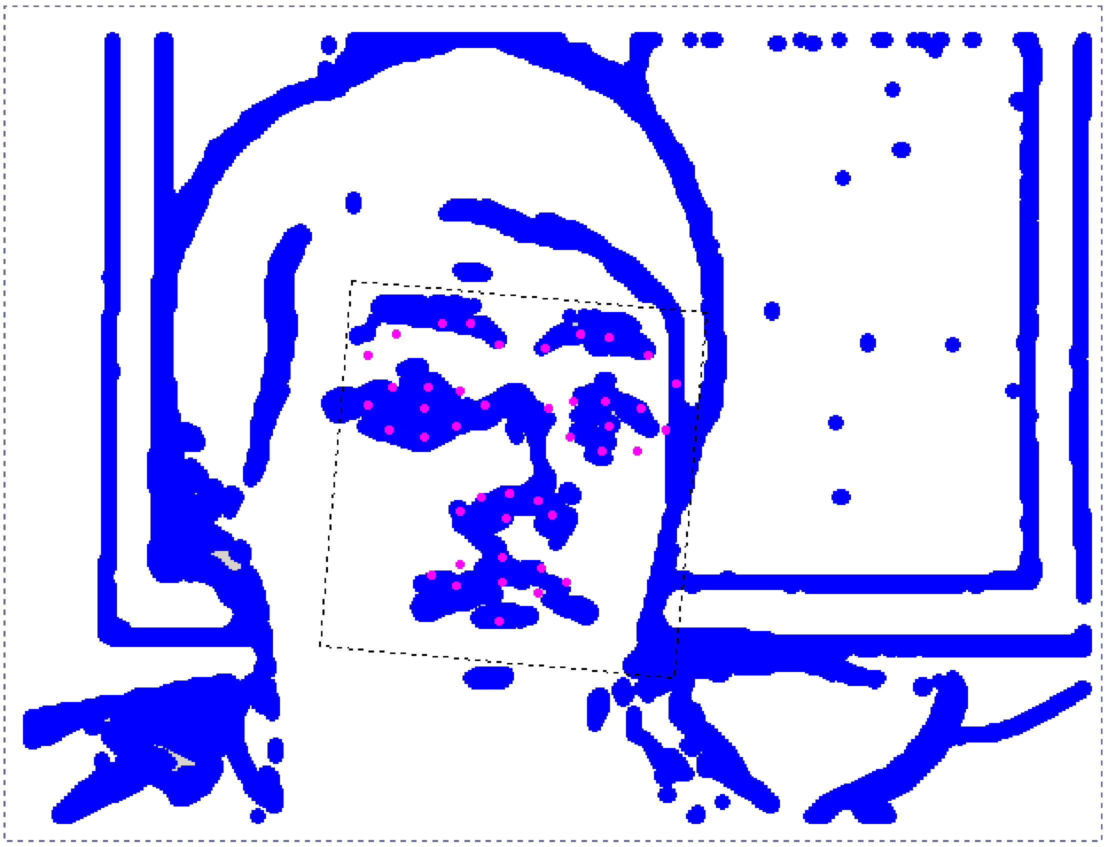

Fig. 2

5. Conclusion

We have explored two algorithms for registering images in a robust manner through the use of feature point pattern matching. Both algorithms allow the user to choose tradeoff between running time and accurate by specifying initial parameters. The first algorithm is based on branch-and-bound search. It is simple and safe, but is relatively slow, especially when high accuracy is desired. The second algorithm, called bounded alignment, is based on combining branch-and-bound with computing point alignments to accelerate the search. It seems to be much faster than the branch-and-bound algorithm in many cases, but it may fail with some small probability.

References

[1] L.P.Chew, D.Dor, A.Efrat, K.Kedem. Geometric Pattern Matching in d-Dimensional Space. Proc. Ninth Ann. ACM-SIAM Symp. Discrete Algorithms, pp. 658-667, Jan. 2003.

[2] M.T.Goodrich, J.S.B.Mitchell, M.W.Orletsky. Approximate Geometric Pattern Matching under Rigid Motions. Proc. First Workshop High Performance Data Mining, Mar. 1998.

[3] M.Gavrilov, P.Indyk, R.Motwani. Geometric Pattern Matching: A Performance Study. Proc. Seventh Ann. European Symp. Algorithms, J. Nesetril, ed., pp. 362-371, July 1999.

[4] D.P.Huttenlocher, G.A.Klanderman, W.J.Rucklidge. Comparing Images

Using the Hausdorff Distance. IEEE Trans. Information Theory, vol. 28, 129-137, 1982.

[5] D.M.Mount, N.S.Netanyahu, J.L.Moigne. Efficient Algorithms for Robust Feature Matching. Data Mining and Knowledge Discovery, vol. 1, pp. 183-201, 1997.

[6] D.P.Huttenlocher, K.Kedem, J.M.Kleinberg. On Dynamic Voronoi Diagrams and the Minimum Hausdorff Distance for Point Sets Under Euclidean Motion in the Plane. Proc. 10th Ann. ACMSIAM Symp. Discrete Algorithms, pp. S931-S932, Jan. 1999.

[7] L.P.Chew, M.T.Goodrich, D.P.Huttenlocher. Geometric Pattern Matching under Euclidean Motion. IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 22, no. 1, pp. 4-37, Jan. 2000.

[8] Sunil Arya, David M. Mount. Algorithms for Fast Vector Quantization. Proc. Data Compression Conference, J. A. Storer and M. Cohn, eds., Snowbird, Utah, 1993, IEEE Computer Society Press, 381-390.

[9] Sunil Arya, David M. Mount. Approximate Range Searching, Proc. of the 11th Annual ACM Symp. on Computational Geometry, 1995, 172-181.

[10] S.M. Smith and J.M. Brady. SUSAN - a new approach to low level image processing. Int. Journal of Computer Vision, 23(1):45-78, May 2002.