Two Dimensional Projective Point Matching

Jason Denton & J. Ross Beveridge
Colorado State University
Computer Science Department
Ft. Collins, CO 80523
denton@cs.colostate.edu

Abstract

Point matching is the task of finding a set of correspondences between two sets of points under some geometric transformation. A local search algorithm for point matching is presented, and shown to be effective at solving problems where the point sets are related by a projective transformation. Random starts local search is shown to be capable of solving very difficult point matching problems, and a heuristic key feature algorithm is presented which can substantially improve the effectiveness of local search in most cases.

1. Introduction

Point matching is a problem that arises in many contexts. Examples include matching interesting points in pairs of images and matching points derived from object models to points extracted from imagery. There are two components to matching, the correspondence mapping between points, and the associated geometric transformation that maps points from one set onto or near corresponding points in another set. Extensive work has been done for two-dimensional point sets and lower order geometric transformations [1, 3, 4, 6], such as rigid motion and similarity transforms. Less attention has been paid to higher order transformations such as general two-dimensional affine transforms, or the projective transformation.

This paper presents new work on two-dimensional point matching under a three-dimensional projective transformation. The projective transform is an eight degree of freedom transform describing how the two-dimensional projection of a co-planar three-dimensional point set changes with changes in camera viewpoint. In practice, this is a good approximation of how objects on the ground appear in different aerial photographs.Our algorithm uses a variant of local search to find optimal and near optimal point matches. Local search is a general and robust search technique which can be applied to the general point matching problem regardless of the specific problem domain. This algorithm is shown to solve difficult projective matching problems derived from both aerial and indoor imagery.

2. Point Matching as Optimization

The point matching problem may be stated as the task of finding a correspondence between model and data points. This correspondence should, under some optimal pose, place as many transformed model points close to data points as possible. The optimal pose should also be physically probably given what is known about the problem. Extreme scale changes should be avoided, as should extreme perspective effects. These conditions allow for the formulation of an error function on a set of correspondences.

$$E_{match} = \frac{1}{\sigma^2}\sum_i ||P^*(m_i) - d_i||^2 + E_{omit} + S(P^*) \quad (1)$$

This evaluation function is asymmetric, matching a point set M designated as the model with data point set D. The model point participating in the i^{th} pairing is m_i, and is matched to the corresponding data point d_i. P^* is the optimal transformation or pose which minimizes the sum of the squared distances between all points m_i and their paired data points d_i. E_{omit} is the number of model points which have no corresponding data point. The value of $S(P^*)$ is a penalty on degenerate or unlikely poses. Under this formulation point matching becomes the task of finding the set of correspondence which minimize this function.

Beyond a certain distance, a transformed model point should not be paired with a data point. This distance, denoted by σ, can be thought of as the error bound on a given data point. If P^* causes the transformed model point to be further from its corresponding data point than σ then this pairing will contribute a value of greater than one to the error. Since the penalty for leaving a model point un-

matched is one, an algorithm attempting to minimize function 1 should drop this pairing from the match. Determining the distance between two points requires the finding of a square root, so it is more efficient to compare the squared distance between paired points with the value of σ^2.

The formulation of the degeneracy function $S(P^*)$ is critical to projective point matching. Ideally this function could be formulated based on a concrete interpretation of P^*. Unfortunately, when all points lie in a plane, a unique interpretation of this pose is not generally possible [7]. Instead, the value of $S(P^*)$ is formulated on the effects of the transformation on the bounding box for the model. The optimal transformation is applied to the bounding box, and the relative scale change of each side is evaluated. If largest scale change exceeds some threshold, usually a factor of two, a linear penalty is applied. This penalty is weighted by one-fourth the number of points in the model set. Thus, if the scale change exceeds the allowable bounds by a factor of one the penalty applied is equal to omitting one-fourth of all model points from the match.

For the projective case, it is possible that the optimal transformation causes the vanishing line to intersect the image plane. Experimentally, this appears to occur in a large number of cases. To drive search away from such conditions, the inverse of the distance from the center of the bounding box on the model to the vanishing line, normalized by the size of the bounding box, is added to $S(P^*)$. When the model has been centered at the origin the distance from the origin to the vanishing line is simply the quantity $\sqrt{g^2+h^2}$, where g and h are parameters of P^* defined below. It should be noted that determining P^* requires that the correspondence in question contains at least four pairs. For those correspondences where there are fewer pairings than this $S(P^*)$ is set arbitrarily high.

2.1. Determination of Optimal Pose

The projective transformation for a model point m_i can be stated as

$$P(m_i) = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} u/w \\ v/w \\ 1 \end{bmatrix}$$

Minimizing the squared Euclidean distance between paired points is difficult due to the ratio of terms. Fully expanding the transformation shows that the error associated with the fit between model and data is

$$\begin{aligned} E_{match} &= \sum_i \left(u_i - \frac{ax_i+by_i+c}{gx_i+hy_i+1} \right)^2 + \\ &\quad \sum_i \left(v_i - \frac{dx_i+ey_i+f}{gx_i+hy_i+1} \right)^2 + \\ &\quad E_{omit} + S(P^*) \end{aligned}$$

Hartley and Zisserman suggest an approximation where the terms being squared are multiplied by their denominator [7]. Thus the previous equation becomes

$$\begin{aligned} E_{fit} &= (u_i(gx_i+hy_i+1) - (ax_i+by_i+c))^2 + \\ &\quad (v_i(gx_i+hy_i+1) - (dx_i+ey_i+f))^2 + \\ &\quad E_{omit} + S(P^*) \end{aligned}$$

The eight parameters minimizing this modified distance term may be found by solving the linear equation $M'p = B$, where M' is the 8×8 upper left sub-block of M, and B is the upper right column vector of M when

$$\begin{aligned} M &= \sum_i (U_i^T U_i + V_i^T V_i) \\ U_i &= \begin{vmatrix} -x_i & -y_i & -1 & 0 & 0 & 0 & x_i u_i & y_i u_i & u_i \end{vmatrix} \\ V_i &= \begin{vmatrix} 0 & 0 & 0 & -x_i & -y_i & -1 & x_i v_i & y_i v_i & v_i \end{vmatrix} \\ p &= \begin{vmatrix} a & b & c & d & e & f & g & h \end{vmatrix}^T \end{aligned}$$

Hartley and Zisserman observe that noise in the location of individual points in either the model or data set can be lead to instability of the pose estimate [7]. This can create problems for search algorithms if the pose changes drastically when pairs are added or dropped. To correct this, both point sets must be normalized. This normalization consists of placing the centroid at the origin, and then rescaling the point set so that the average distance from the origin to each point is $\sqrt{2}$. Because local search moves through correspondence space, this normalization can be done once at the beginning of the search process, along with the corresponding adjustment on σ and maximum allowable scale change. Once the optimal correspondence has been found, the pose can be estimated based on the original point sets.

3. Local Search

Local search requires that solutions can be evaluated and a neighbor definition be imposed upon the discrete solution space. For random starts local search, an arbitrary solution is chosen as the starting point and every solution in that neighborhood is evaluated. If an improvement is found local search moves to the best solution and searches the new neighborhood for an improvement. Search terminates when no improvement can be found.

For point matching the neighborhood for a given solution can be defined as all correspondences which differ by

only a single model to data pairing, with the restriction that model points may not match to a data point already paired with another model point. A null point is introduced so that model points may go unmatched. Local search seeks a correspondence which will minimized equation 1, but it is important to note that, due to the formulation of the error function, this progress through correspondence space is guided by the global pose of the model, which is recomputed at every step. This is similar to work by Beveridge on two-dimensional line matching [2].

Experimentally, it appears that initial random solutions should contain five pairs. This provides enough pairs so that one pairing may be dropped immediately and the optimal pose can still be estimated. When each random trial is independent, the probability that a sequence of t trials will all fail to find the global optima drops as an exponential function of t [11].

3.1. Key Feature Local Search

The effectiveness of local search is highly dependent on the quality of the initial solution chosen, so it is worthwhile to bias the search by choosing an initial solution that is in some sense near the optimal match. Points which are spatially proximal to each other in one point set may be expected to match to points which are spatially proximal to each other in the other data set. This fact can be exploited by constructing a set of key features that serve as initial solutions. Key features are based upon spatially proximal k tuples of model and data points. For each model point, the $k-1$ nearest model points are identified, and likewise, for each data point, the $k-1$ data points are identified. A complete set of tuples are formed by pairing each model point with each data point: there are $n = md$ such pairings when there are m model points and d data points. For each pairing, $k-1$ proximal model points may match $k-1$ proximal data points. Let the ordering of the data points be fixed, then there are as many such matches as there are permutations of the $k-1$ model points. Thus, there are in total, $n(k-1)!$ proximal k tuples.

Once all key features have been created, equation 1 can be applied to rank the tuples and discard those with large scale changes or other pathologies. Those tuples with low match error are likely to be good initial matches from which to start a local search trial, and thus are ranked higher on the list. Using tuples in this manner to index into possible solutions, and subsequently aligning model and data to either accept or reject matches is a common theme in many algorithms [10, 8].

Typically, the size of the key feature is equal to the minimum number of paired points necessary to constrain the transformation: in our case $k = 4$. However, local search has an important aspect missing from typical algorithms: points may be dropped from the match. In part as a consequence of this ability to drop as well as add points, empirically we've observed local search performing better performance with $k = 5$.

4. Results

Random starts and key feature local search where tested on four data sets drawn from real imagery. The simplest imagery showed an irregular polygon printed in black on white paper. Another imagery set showed a framed, abstract, painting hanging on a wall. The third set shows a text book lying on a table. The fourth set was composed of two images drawn from aerial photographs of Fort Hood [5]. Points where extracted from the test imagery using the corner detector provided as part of Intel's OpenCV library [9]. Matching problems where create by pairing every point set with every point set produced from the same set of imagery, including itself.

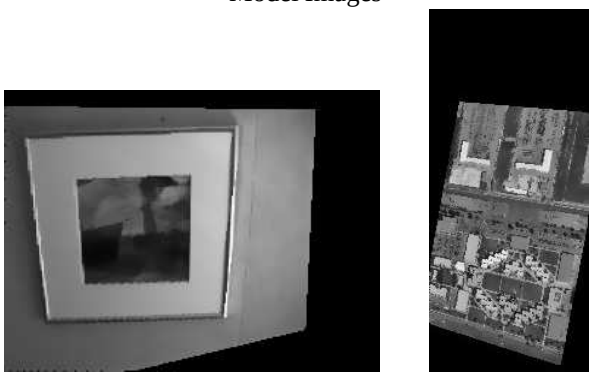
Each point set from the polygon imagery contained between 9 and twelve points. The four images of the picture produced from 14 to 16 points each. Both the polygon and the picture imagery produced point matching problems that effectively contained no clutter, and where each point had very little associated noise. In these problems the majority of points in each image had a match in the other. The six images taken of the text book where qualitatively different. These images produced between 39 and 53 points, and these points contained a great deal more noise. Although most points in each point set continued to match, approximately one-fourth in each match. This additional clutter significantly raises the difficult of point matching. The two images of Fort Hood produced 65 and 67 points respectively, and also contained significant amounts of noise and clutter.

In order to provide a visual assessment of the effectiveness of local search on each problem, the imagery used to produce each model set was transformed by the optimal pose found, and compared against the imagery used to produce the data set. Figure 1 shows typical results for both types of local search.

4.1. Results for Random Starts Local Search

Both forms of local search proved capable of solving all problems drawn from the book and polygon imagery. Ten thousand trials of random starts local search where run on each of these problems with the global optima being found many time. Based on this the failure rate f_r of local search for these problems, and for thus the expected number of trials required to achieve 99% confidence in the result, can be estimated as $t^* = \log(0.99)/\log(f_r)$. Problems from the polygon imagery generally required less than 1000 trials to

Model Images



Transformed Images

Target Data Images

Figure 1. Typical Results for Local Search Point Matching

achieve 99% confidence in the solution. On a 1GHZ Pentium III machine this requires under a second. Problems from the picture imagery where substantially easier, with t^* less than 1800 in all cases. The increased size of the point sets does increase run times, and these problems required on average, about five seconds to solve. Random starts local search was also run on the text book imagery, this time with 100,000 trials per problem. This proved sufficient to solve most, but not all, problems at least once. Based on the results for those problems which where solved it appears that these problems require about a half million trials to be reliably solved.

4.2. Results for Key Feature Local Search

Key Feature local search was run on all data sets. This algorithm solved every problem in the polygon and picture problems sets searching only the first five entries on the key feature list, in most cases the top ranked key feature, the one with lowest E_{match} score, lead to the global optima. In such cases local search can be run in well under a second. Key feature local search also proved capable of solving many problems from the text book imagery. In most cases, the first entry on the key feature list resulted in the global optima being found. In seven cases the first key feature to result in the global optima being found was significantly further down the list. In three of these cases the first successful key feature was within the first sixty entries on the list. In the remaining four the only successful key feature was found later than position 10,000 on the list. The two problems taken from the Ft. Hood imagery where also solved by key feature starts local search, in one case with the key feature ranked 720, the other with a key feature ranked below 90,000.

Both random starts and key feature local search failed to solve every problem taken from the book imagery, but they did not solve exactly the same problems. Three problems that went unsolved by random starts local search where solved by the key feature algorithm, and key feature local search failed to solve two problems for which random starts found the global optima. This is consistent with what should be expected from the algorithms. In theory random starts can solve every key feature local search can by choosing at random an initial solution which corresponds to a successful key feature. In those cases where clutter and noise prevent any successful key feature from being formed it is still possible that some combination of pairings could be a successful initial solution.

5. Conclusion

Point matching under the projective transform is useful in contexts where points derive from co-planar 3d features:

for example registering aerial photographs. The algorithms presented offer a fairly general and robust means of solving such problems. In most cases, key-feature starts local search can solve problems involving a large number of points by searching only a small portion of its key feature list. In situations where this fails, the more expensive but theoretically more general random starts local search algorithm frequently can find the optimal match, and in theory it will always do so given sufficient trials.

More images from and details of these experiments can be found at http://www.cs.colostate.edu/∼denton/pointmatching

References

[1] H. Baird. *Model-based Image Matching Using Location*. PhD thesis, Princeton University, oct 1984.

[2] R. J. Beveridge. How easy is matching 2d line models using local search? *T-PAMI*, 19(6), jun 1997.

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

[4] T. Cass. Polynomial-time object recognition in the presence of clutter, occlusion, and uncertainty. *ECCV*, 92:834–842, 1992.

[5] D. Gerson and S. Wood. Radius phase ii - the radius testbed system. In *Arpa Image Understanding Workshop*, pages 231–237, Monterey,CA, nov 1994.

[6] A. Goshtasby and G. C. Stockman. Point pattern matching using convex hull edges. *IEEE Transactions on Systems, Man, and Cybernetics*, 15(5):631–637, 1985.

[7] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[8] D. P. Huttenlocher and S. Ullman. Recognizing Solid Objects by Alignment with an Image. *International Journal of Computer Vision*, 5(2):195 – 212, November 1990.

[9] Intel. Intel open source computer vision library. Software, 2000. http://www.intel.com/research/mrl/research/opencv/.

[10] D. G. Lowe. *Perceptual Organization and Visual Recognition*. Kluwer Academic Publishers, 1985.

[11] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization Algorithms and Complexity*, chapter Local Search, pages 454–480. Prentice-Hall, Englewood Cliffs, NJ, 1982.