An Algorithm for Projective Point Matching in the Presence of Spurious Points

Jason A. Denton

Texas Tech University

J. Ross Beveridge

Colorado State University

Abstract

Point matching is the task of finding correspondences between two sets of points such that the two sets of points are aligned with each other. Pure point matching uses only the location of the points to constrain the problem. This is a problem with broad practical applications, but it has only been well studied when the geometric transformation relating the two point sets is of a relatively low order. Here we present a heuristic local search algorithm that can find correspondences between point sets in two dimensions that are related by a projective transform. Point matching is a harder problem when spurious points appear in the sets to be matched. We present a heuristic algorithm which minimizes the effects of spurious points.

Key words: geometric pattern recognition, projective transform, point matching

1 Introduction

Finding the correspondences between two arbitrary point sets is a difficult problem with wide spread applications. Point matching has been used for sonar processing [1], finger print analysis [2], and localization of DNA markers [3]. Choice of an algorithm for point matching is largely driven by the dimensionality of the point sets to be matched and the class of geometric transformations which may relate them. Algorithms such as softPOSIT [4] provide for matching three-dimensional model points against two-dimensional

Email addresses: jason.denton@ttu.edu (Jason A. Denton), ross@cs.colostate.edu (J. Ross Beveridge).

data sets. A rich body of literature dealing with matching two-dimensional points to two-dimensional points exists, largely focused around tree search type algorithms [5–7]. These algorithms are very effective at finding matches when the class of transformations between the point sets is of a relatively low order, as with rigid or similarity transforms. They are less effective at dealing with higher order transforms, such as the projective transform. For matching two-dimensional points to two-dimensional points under a projective transform when there are no further constraints on the problem, the most common approach is to apply some variant of Fischler's RANSAC algorithm [8,9]. Here we present an alternative local search algorithm that is more accurate and considerably quicker.

Spurious points are points in one set for which there is no corresponding match in the other. Spurious points makes determining the correct set of correspondences between two point sets very difficult [10,11]. As the number of spurious points increases, point matching algorithms suffer a marked drop in efficiency and reliability. As a means of characterizing the difficulty in such cases, we examine the behavior of RANSAC as the number of spurious points increases. To deal with this problem, we propose a more robust local search algorithm, and a heuristic for helping it cope with spurious points. The combination of this heuristic with local search produces an algorithm better able to deal with spurious points.

The specific problem addressed here is that of finding a partial, one-to-one correspondence mapping between a set of two-dimensional points designated as the model and a set of two-dimensional points designated as the data. With the exception of a some ill-defined or under-constrained cases, all correspondences uniquely define an optimal projective transformation that places model points in close proximity to their corresponding data points. We treat this as a pure point matching problem, meaning that no additional outside constraints are introduced. In many practical applications, additional domain specific constraints can be incorporated to profoundly reduce the space of potential matches. Such constraints include approximate knowledge of camera parameters [12] and auxiliary features associated with points typically derived from local image properties around the point [13].

This paper examines pure projective point matching absent additional and more task specific constraints. We will illustrate our algorithms on real data derived from images related by projective transformations, and indeed the associated point matching problems are highly challenging. However, we claim that our algorithms, because they perform pure point matching, are more general than this single task suggests. This task was selected for illustration, and anyone with a strong interest in fast solutions to this specific task will do well to use algorithms that incorporate additional constraints such as those described above as this will almost certainly enhance performance. Our argument is that the general point matching algorithm is in many cases sufficient, and because it is not tied to domain specific constraints such as prior expectations on camera parameters or image appearance, better able to meet a broader range of potential applications.

1.1 The Projective Transform

The projective transformation captures how two sets of two-dimensional points may be related by a three-dimensional camera motion. It is an excellent approximation for many 3D imaging problems where the scene is essentially planar. For example, the transformation between two aerial photographs, taken from a sufficient height and from different viewing angles can be closely approximated using the 3D projective transformation. As an alternative example, consider finding the correspondences between two different star maps.

In homogeneous coordinates, the projective transformation can be stated as

$$P = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} u/w \\ v/w \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{ax+by+c}{gx+hy+1} \\ \frac{dx+ey+f}{gx+hy+1} \\ 1 \end{bmatrix} \tag{1}$$

Critical to the use of the local search algorithm proposed here is the ability to find a projective transformation, P^*, that aligns two point sets, given a set of correspondences between the point sets. This transformation can be understood as the model's pose, or position, within the data set [1] . Each pairing between model and data gives rise to two constraints; one in the x dimension and one in the y dimension. With four model to data correspondences, P^* can be found exactly. When there are more pairings in the correspondence, a least squares methodology can be employed to approximate the optimal transformation. Hartley and Zisserman present the details of doing this, and the associated stabilizing normalization, in [14]. We discuss how to apply this normalization in the context of the algorithm presented here in [15].

[1] Here we use an asymmetric treatment where the projective transformation is applied to one set, the model, to transform it into the space of the other set, the data. This is convenient, and for problems that are intrinsically symmetric, such as image to image matching, one is arbitrarily designated as the model.

2 RANSAC and Spurious Points

The presence of spurious points in one or both of the point sets to be matched greatly complicates point matching. To explain why, it will be useful to review the analysis of RANSAC, focusing on the effects of spurious points in the point sets. This analysis is not unique, others have presented similar analysis [16,11].

RANSAC itself is a simple algorithm first proposed by Fischler and Bolles [8]. In the context of projective point matching, four model-data pairings are chosen at random, the optimal pose is found, and the entire model set is transformed. Each transformed model point within σ of a data point supports the hypothesis that the given pose correctly relates the model to the data. After a predetermined number of trials the transformation matching the most model to data points is taken as the correct answer. Torr allows RANSAC to iterate [17], using the pairings found in the previous iteration to recompute the optimal pose, which is then used to find a new set of pairings for determining the optimal pose, and so on. This change might be viewed as a first step toward making RANSAC an iterative local search algorithm in the same spirit as those proposed by Beveridge for line matching [12,10]. However, as will become clear when we contrast this enhancement with the local search algorithm developed here, there are significant differences.

RANSAC's ability to find a correct solution depends on it choosing four correct pairings for its initial correspondence [11]. Consider a model set containing m points being matched to a data set with d points, with r pairings in the correct correspondence. We assume that a single trial of RANSAC will be successful if it chooses any four of the r correct pairings as the initial hypothesis. Enforcing a one to one matching constraint, there are $C(r, 4)$ possible ways to choose a correct set of initial pairings. Compare this to the number of ways to choose four pairings in general. Again enforcing the one to one constraint, there are md ways to choose the first pairing, $(m-1)(d-1)$ ways to choose the second, an so on. There are 4! orderings in which four pairings may be chosen, all of which are equivalent for our purposes.

Equation 2 expands all these terms and gives the probability p of RANSAC picking a correct set of correspondences.

$$p = \frac{C(r,4)}{P(m,4)P(d,4)4!} = \frac{r(r-1)(r-2)(r-3)}{md(m-1)(d-1)(m-2)(d-2)(m-3)(d-3)} \quad (2)$$

When r, m and d are large with respect to the number of pairs to be chosen, in this case four, equation 2 can be approximated as:

$$p \cong \left(\frac{r}{md}\right)^4 \quad (3)$$

Using the approximation in equation 3, the number of trials required for RANSAC to find a solution, n, with a given probability or confidence c, is:

$$n = \frac{\ln(1-c)}{\ln\left(-\frac{r^4 - m^4 d^4}{m^4 d^4}\right)} \tag{4}$$

Spurious points increase m or d without increasing r, quickly driving the denominator of equation 4 asymptotically towards zero. As long as m and d are not much larger than r, where spurious points are the exception rather than the rule, this problem is not great. As the number of spurious points grows, RANSAC is quickly overwhelmed; particularly when the point sets to be matched are already large. In practice, equation 4 is somewhat optimistic, since not every combination of four correct pairings will lead RANSAC to a correct solution. If three or more of the pairs selected are co-linear, then the corresponding pose is undefined, and P^* can not be found. Many problems of practical interest contain some underlying structure, which may give rise to such instances.

3 Point Matching as Combinatorial Optimization

When determining the geometric relationship between two point sets, it is not unreasonable to rate any particular transformation strictly on the number of points it aligns. This is the stopping criterion for RANSAC, and the idea behind Hough transform algorithms [18]. However, Hough transforms suffer from a high number of false positives when the location of each point is noisy [19]. To some degree, RANSAC suffers from a similar problem. The projective transform provides enough degrees of freedom that it is often possible to find a transformation which places many model points close to data points, but which has no relation to the correct transformation.

We would like to distinguish between correspondences which match the same number of points. In short, matches that place paired points closer together are better. This suggests an objective function based on the average residual error in fitting the model to the data, with a second term for the number of unmatched points. Torr and Zisserman follow a similar line of thought in developing the MLESAC criterion for RANSAC [20]. We expect that the uncertainty in the location of each point is less than σ. If the residual error for a match is normalized by σ^2, pairings which place the transformed model point a distance of less than σ from the appropriate data point will incur a penalty of less than one. If the penalty for leaving a model point unmatched is set to be one, then the result will be an objective function which favors pairings within the error bound σ, and penalizes those further out.

Here we use the following match error function.

$$E_{match} = \frac{1}{\sigma^2} \sum_i ||P^*(m_i) - d_i||^2 + Omit + S(P^*) \tag{5}$$

The terms m_i and d_i refer respectively to model points and data points that participate in the i^{th} pairing in the given match. The term $Omit$ is the number of model points left unmatched. The term $S(P^*)$ is a discriminator based on the optimal pose for a given match set, explained below. When there are fewer than four pairs in the match, P^* can not be determined and E_{match} is set arbitrarily high. With this formulation, point matching becomes a combinatorial optimization task, finding the set of pairings between model and data which minimize E_{match}.

3.1 Evaluation of Model Pose

Many projective transforms are unlikely or impossible. Some correspond to the vanishing line intersecting the image plane. Others represent poses with extreme projective effects, where it is unlikely the sensors are accurate enough to gather useful data. A means of heuristically guiding search away from such transformations is important, and is provided by the $S(P^*)$ term in the match error. Specifically, here we will define $S(P^*)$ as a sum of two terms, i.e. $S(P^*) = V(P^*) + K(P^*)$ where $V(P^*)$ and $K(P^*)$ will now be explained.

3.1.1 Interpretation of Model Pose

The most straight forward way to evaluate pose would be to determine the relative camera angles that a given pose describes. Unfortunately, this is not possible. When both point sets are two dimensional, the fundamental matrix, and thus the relative camera angles, are not uniquely determined. In this case any skew symmetric matrix will satisfy the fundamental matrix [14]. This means that even though any given correspondence has a unique optimal pose, there is not an associated unique camera position. Without knowing the camera position we can not say definitively what effects scale and perspective are having on the transformed model. In place of a constraint derived from camera position directly, a heuristic technique is developed to avoid transformations which are unlikely to be correct.

3.1.2 The Vanishing Line

When the vanishing line is far from the bounding box on the model it indicates that perspective makes little difference in the position of transformed model points. Conversely, a vanishing line passing close to the bounding box indicates severe perspective effects. Such a situation usually represents an undesirable pose and one that should be avoided by the search.

As can be seen from equation 1, the vanishing line can be found at $gx+hy+1 = 0$. Applying the standard formulas for the distance between a point and a line we can show that the quantity $1/\sqrt{g^2+h^2}$ is the distance from the origin to the vanishing line. We normalize this distance by the length of one side of the bounding box, and use the inverse as a penalty in the match error that discourages point matches that give rise to such a pose estimate. Thus our vanishing line penalty term is

$$V(P^*) = \sqrt{g^2+h^2} \tag{6}$$

A more complete interpretation of the effects of g and h on model pose can be found in [15].

3.1.3 A Scaling factor

It is also desirable to drive the search away from those poses which represent large changes in the scale of the model. This is easily accomplished by examining the effects of the transformation on the bounding box of the model. The optimal pose, P^*, is applied to this bounding box and the change in the length of each side is measured. Large scale changes may indicate either large changes in model scale, or extreme perspective effects; both of which should be avoided. The largest scale change of the four sides is taken as the basis for computing the scale penalty, $K(P^*)$. If the scale change is within a given tolerance, no penalty is applied and hence $K(P^*) = 0$. Beyond this threshold, a linear penalty is applied. In order to keep the penalty applied in line with the size of the model, this penalty is scaled by one-fourth the number of model points.

4 Local Search for Point Matching

Local search is a straight forward algorithm for combinatorial optimization [21]. Given the previous definition of point matching as optimization, it can provide a generic point matching algorithm. Random sampling is typically part of local search, with initial solutions chosen at random. To this extent, it is similar to RANSAC. However, local search imposes a neighborhood structure on the search space and then repeatedly seeks better solutions within these neighborhoods. For point matching, we define the neighborhood as all matches differing by only a single point pairing. More precisely, neighbors are matches with a pair of model and data points added, dropped, or with a model point paired up with an alternative data point. Better solutions are those with lower match errors, as defined by equation 5. Local search is constrained to consider only one-to-one mappings from model to data.

Each execution of local search from an initial match to a locally optimal match is called a trial, and when initial matches are chosen at random and independently, then the chance of all trials failing to find a globally optimal solution drops exponentially [21]. The analysis of how many trials are required depends upon the probability p of obtaining a good match on a single trial, and in this regard random starts local search is similar to RANSAC. However, there is a very critical difference. Where p for RANSAC is precisely defined by m, d and r (equation 2), the probability p that random starts local search finds an optimal or near optimal match is the result of a subtle interplay between the discrete search in correspondence space and the global direction given this search by the continuously updated projective transformation P^*. In short, local search can move through match space, and need not necessarily start at a good match in order to successfully arrive at a good match. In past work with random starts local search we have shown how, for specific problem instances, reliable estimates of p may be obtained [10]. Here we will not replicate the analysis of by which we estimate p for a general set of matches selected at random, as the heuristic for generating initial matches for local search presented in the next section is far superior to a general random sampling strategy.

The heuristic for selecting initial matches for local search identifies five pairings based upon a spatial proximity constraint. Local search then uses equation 5 to evaluate all correspondences which differ by only a single pairing. Since more than the minimum four pairs required to determine the optimal pose are provided, local search will consider dropping individual paired points from the match. It will also consider adding a single pairing to the match, by pairing an unmatched model point to an unmatched data point. Finally, it will consider the possibility of taking a model point that already participates in the correspondence and pairing it with a different data point. Realize that for every alternative considered, a new transformation P^* is determined. If any of these options produces an improved solution, we take the best solution as the starting point for another round of search, using the same neighborhood definition now applied to the improved solution. When no further improvements can be found, the search terminates.

Note that local search is more cautious than iterative RANSAC [17] in the sense that it re-evaluates a match every time a pair of matching points is added. This gives local search an advantage in dealing with spurious points. For example, consider how iterative RANSAC behaves when given a choice between the correct match to a certain point and a point which is closer based upon the current pose estimate. Iterative RANSAC is likely to grab the closer point. In contrast, local search will try both alternatives, recomputed the full projective transformation both times, and in so doing it is more likely to select the correct pairing.

Local search can also drop a bad pairing if it started with one or picks one up later. In some cases it is possible for local search to start with a poor initial solution, either in terms of few correct pairings or a very incorrect initial pose, and move to a more optimal solution. This robustness allows local search to succeed even when the process used to generate initial matches is imperfect.

The formulation of the objective function is important to the behavior of local search. When there are relatively few pairs in the match set, the evaluation of model pose is the most significant term in equation 5. With few pairs there is little residual error, and the definition of the neighborhood means that for any particular step in the search the value of *Omit* in equation 5 can not change by more than one. However, because there are few pairs in the match, the addition of a single pair can significantly affect the optimal pose and its evaluation, $S(P^*)$. As more pairs are added, P^* becomes more stable and the other terms in the objective function become more important. As we will explain, this behavior is one of the reasons the heuristic we present next can often overcome the presence of spurious points.

5 A Key Feature Heuristic for Local Search

The projective transformation changes the relative distances between points, but in general, points which are close to each other will remain close to each other after transformation. In our past work we have exploited this to create a heuristic for line matching under similarity transforms [22,23]. The general approach of enumerating a set of feature clusters as seeds from which to then seek full matches is not new, being evident in many algorithms including those of Lowe [24] and Huttenlocher [25]. Here we adapt the key features from [22,23] to work with points rather than line segments. The result is a set of point groupings, or key features, which can be evaluated and ranked using equation 5, and subsequently used as the starting points for trials of local search.

Key features are constructed by finding "clusters" of points, composed of a single anchor point and its c closest neighbors. There is one cluster for every point in a set. A naive algorithm can find the clusters for a point set in $O(n^2)$ time. Key features are formed by taking one cluster from the model set, and another cluster from the data set. The anchor points from the two clusters are paired. Then, every possible combination of pairings between points in the two clusters is added in turn to the anchor pairing, resulting in $c!$ key

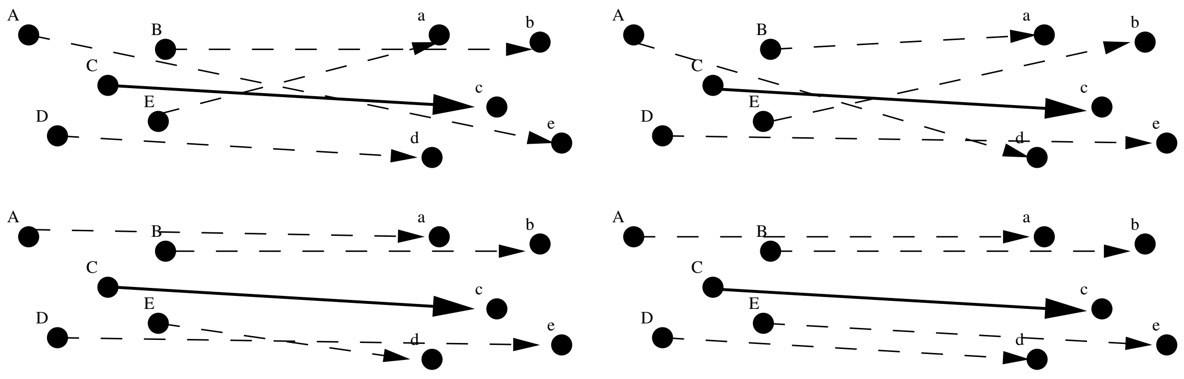


Fig. 1. An Example of Key Feature Construction

features. If every possible combination between model and data set clusters is tried, the result is $c!md$ key features. Figure 1 shows some possibilities for combining nearest neighbors to form different key features. In this example we are combining the four nearest neighbors, so there would be a total of twenty-four key features generated from these two clusters.

It is tempting to take the heuristic further, and restrict the combination of nearest neighbors based on their positions relative to the anchor point. Under the right conditions certain refinements may be profitable. In the general case, however, uncertainty in point locations make such refinements problematic. Errors in measuring each point could cause a pair of points to swap their positions relative to the anchor point even if they are not close to each other. Even when the locations of points in one set are known exactly, such as in model recognition, uncertainty in the locations of points in the other set could cause a more sophisticated heuristic to overlook a correct pairing.

A cluster may be considered to have been disrupted when the c closest neighbors to a given anchor point in the model includes points that have no associated matching point in the data, or vice versa. With a disruption, every combination of neighboring points will contain at least one erroneous pairing. Uncertainty in point locations may result in a reordering of the relative positions of points, disrupting the cluster. The projective transform changes the relative distances between points, and so clusters may be disrupted even if point locations are known exactly. More damaging are spurious points. A single spurious point may disrupt several clusters. As the number of spurious points increases it becomes increasingly probable that all clusters will be disrupted, so that no there will be no key feature composed entirely of correct pairings.

Fortunately, local search can often recover from such errors. At each step in the search process local search explicitly considers the possibility of dropping each of the pairs in the current solution. In order to take such a step, the current solution must include enough pairs such that when one is dropped there are still enough pairs to uniquely determine the optimal pose, P^*. This will increase the penalty for model points not participating in the match by one, offset by a small decrease in the residual error of fitting model points to data. However, if the pairing that was dropped was indeed an incorrect pairing, there is a strong likelihood that it was driving the optimal pose to one that increased the penalty term $S(P^*)$. When the number of pairings in a match is small, this effect can be quite acute as a single bad pairing can change the bounding box on the model set significantly. Consequently, it is relatively easy for local search to discard one or several bad pairing from a match, particularly early on in the search process.

This leads us to a simple solution for dealing with cluster disruption, whether it arises from sensor noise, perspective effects, or spurious points. When forming the clusters used in generating key features we set c to be equal to the number of pairings required to uniquely determine the optimal pose. Together with the anchor pairing this gives each key feature one more pairing than required to determine the pose, allowing local search to potentially discard one of these pairings on the first step. Even if the cluster is disrupted for some reason, local search may still be able to determine a correct set of pairings and proceed to fill out the match. In the case of the projective transform, this requires setting $c = 4$, giving us a key feature list containing $24md$ partial matches.

There is also a simple and obvious means of prioritizing which key features to examine first, in some sense answering the question: "Which features are more likely to be key?" We have already introduced a match error, equation 5, and so the match error is computed for each of the $24md$ partial matches and these are then sorted from lowest to highest error. If at least one key feature is not disrupted and is composed entirely of correct pairings, it will frequently reach the top of the list. If this does not occur, then there is still a reasonable chance that a key feature that is mostly correct will obtain a high ranking on the list and that local search will be able to correct this feature and use it to reach a correct correspondence. In practice, $24md$ is too many key features to search when m and d are large. We normally prune this list, discarding those key features which have extremely degenerate poses, unlikely to have been caused by a single erroneous pairing.

We can now present a concise description of key feature local search. Figure 2 presents pseudo-code for complete algorithm. Note that the actual implementation employees optimizations not shown here.

```
function generate_key_feature_list(M : Point_set, D : Point_set)
returns List of matches

f = 0
for i from 1 to $M_{size}$
  $C_{mi}$ = four_nearest_neighbors(M,$m_i$)
  for j from 1 to $D_{size}$
    $C_{dj}$ = four_nearest_neighbors(D,$d_j$)
    $P_j$ = permutations_of($C_{dj}$)
    for k from 1 to 24
      $K_f$ = pair($m_i$,$d_i$);
      concatenate($K_f$, pairs_from_vector($C_{mi}$,$P_{jk}$))
      evaluate_match($K_f$)  //using eq 5
      f = f + 1
sort(K)   //from best to worst according to eq 5
return K

function local_search_trial(initial : Match) returns Match
best : Match
S : List of Matches

best = initial
do
 S = matches_different_by_one_pair(best)
 For all i in S
   if (evaluate_match($S_i$) < evaluate_match(best))
      best = $S_i$
Until no improvement found
return best

function key_feature_local_search(Model : Point_set, Data : Point_set)
returns Match

K = generate_key_feature_list(Model,Data)
best = local_search_trial($K_1$)
for i from 2 to number_trials
  $R_i$ = local_search_trial($K_i$)
  if (evaluate_match($R_i$) < evaluate_match(best))
    best = $R_i$
return best
```

Fig. 2. The Key Feature Local Search Algorithm

6 Illustration on Real Data

Here we present timing results for key feature local search on five challenging problems. As one point of comparison, a million trials of RANSAC was applied to all four problems and none were solved. This is what we would expect given the few number of good pairs relative to the total number of pairs and the analysis in Section 2. While in past work we have demonstrated random starts local search on projective matching [15] and estimated the probability of finding an optimal match in a given trial, that prior experience has taught us to value the key feature variant of the local search algorithm and that application of random starts local search to the problems presented here is probably not worthwhile.

6.1 Our Data

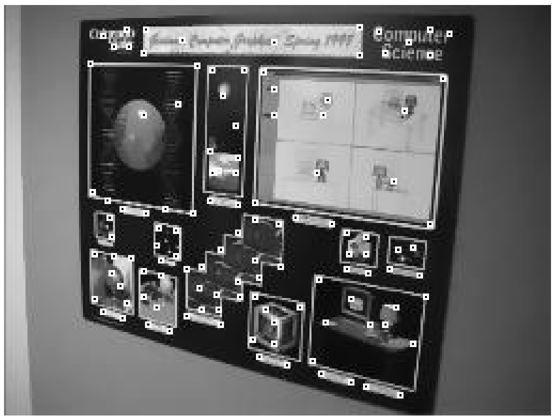
We use data drawn from digital images to illustrate the key feature local search algorithm. Real imagery provides a convenient way to construct challenging point matching problems. The problems are challenging because feature extractors may either miss points in an image or detect points unrelated to the structure being matched, introducing spurious points into both model and data sets. Because there are images underlying the point sets, we as external observers of algorithm performance can review the quality of a resulting match by transforming the image from which the model was drawn into the coordinate reference frame of the data image and then superimposing the two images. To obtain point sets from the images we converted each image to grey scale, and applied the Harris corner detector [26,27].

These problems represent homographies and object recognition problems. As discussed in the introduction, let us emphasize that we are concerned with the general point matching problem and not finding homographies or object recognition per se. Other approaches, using more information than simple point locations, may provide better solutions to finding homographies and recognizing objects. However, these approaches are correspondingly less general and often require information about the camera including the focal length. Alternatively they may place constraints on the relationship between the image pairs, for example making use of color or texture information. Our algorithms operate only on the locations of points in the point sets and so are applicable to a broader range of applications, for instance, the processing of star maps.

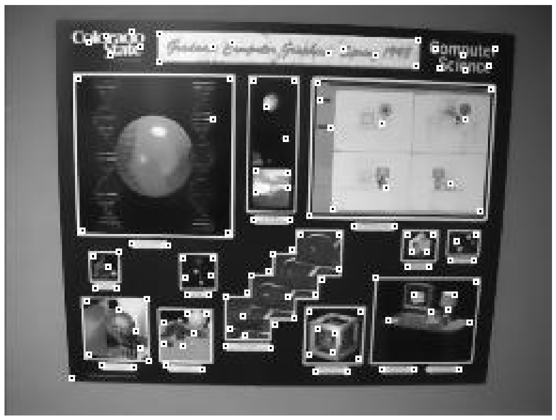
For each of the matching problems below we report run times in terms of seconds of processor time used. These timings where done on a single processor, 2.8 GHZ Pentium 4 with 1G of RAM. Local search innately lends itself to parallel and grid type processing. Although we will not discuss it further, we note that the implementation used here is capable of exploiting multiple processors. Doing so results in a small amount of overhead for spawning instances and synchronization, but otherwise yields a nearly linear improvement in performance.

6.2 Key Feature Local Search Results

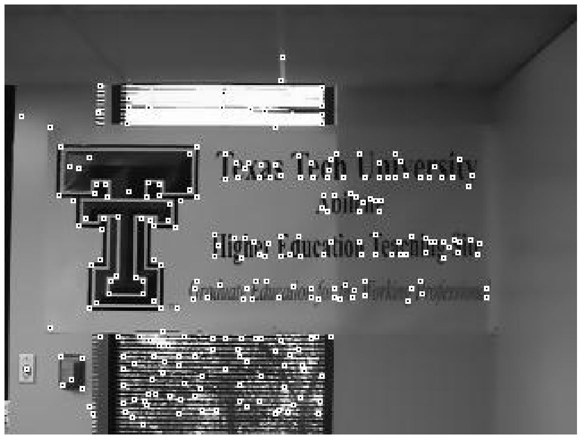
Figure 3 shows two data sets giving rise to four point matching problems. The first data set for which we present results shows two views of a poster hanging



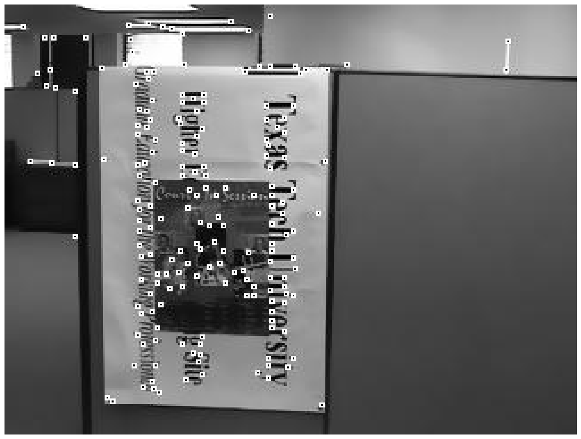
Poster 1 : 119 points



Poster 2 : 128 points



Banner 1 : 262 points



Banner 2 : 189 points

Fig. 3. Data Sets for Point Matching

on a wall. This data is interesting because it contains a large number of points, and the two views are at a significant angle relative to each other. The second set shows an advertising poster for Texas Tech University (TTU). The first image shows the complete banner, with a number of spurious points arising from the window. The second image shows only a portion of the banner, with some of it obscured by a another poster not present in the first image. The TTU image pair is the more difficult of the two in that there are things visible in one image that are not visible in the other, and this is true in both directions. Each pair of images gives rise to two point matching problems because we alternate which image is treated as the model and which is treated as the data.

Table 1 shows the results for running key feature local search on these problems. The column marked “KF Pos” gives the rank of the first key feature that lead local search to a correct match. The “KF Time” column gives the time required to create, evaluate, and sort all $24md$ key features on the list. For these problems the first 100 key features where used to seed trails of local

Table 1
Results for Key Feature Local Search

Model	Data	Pairs	KF Pos	KF Time	Time
Poster 1	Poster 2	106	9	2.48	99.05
Poster 2	Poster 1	111	20	2.48	88.69
Banner 1	Banner 2	116	95	13.91	512.18
Banner 2	Banner 1	94	13	14.20	472.45

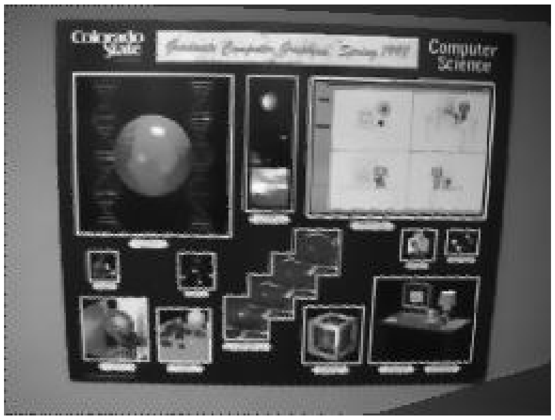
search. The final column gives the total time required to generate the key features and run all 100 trials. All times are given as seconds of processor time used.

The "Pairs" column gives the number of pairs found in each match, and the discrepancy between the number of pairings found when the model and data sets are reversed is worth remarking on. Ideally these numbers would agree. However, our objective function is not symmetric, and so it is not entirely surprising that some border line pairings which where chosen when matching from one set to another would be rejected when the roles of the model and data set are reversed.

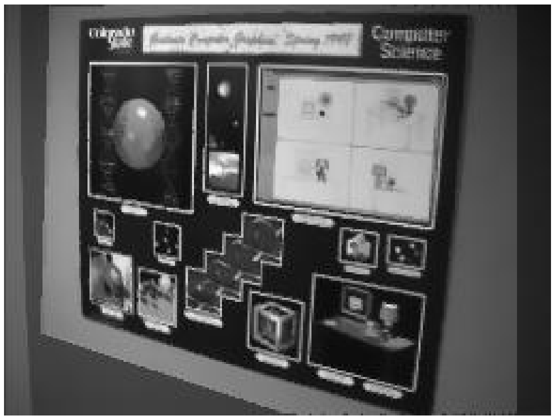
Figure 4 shows the results of the point matching. Here we have transformed each model by the optimal pose for the best match, and overlaid this transformed image with the image that gave rise to the data set. As can be seen, the correct correspondences for the poster and banner have been found, leading to a pose which correctly overlays the objects of interest.

Figure 5 shows a points extracted from single Christmas card used as a model, and a single point set containing points from a set of four Christmas cards, three of which have the same pattern and a fourth which is different. Both images where taken against a black background. The goal here is to locate all three instances of the matching card in the data. Finding all three instances is simple, we retain all local maximums found by applying local search to the key feature list, and take the first three that differ from each other by some sufficient number of pairings. Relative to any match to one of the cards, the points associated with the other two matching cards are treated as spurious.

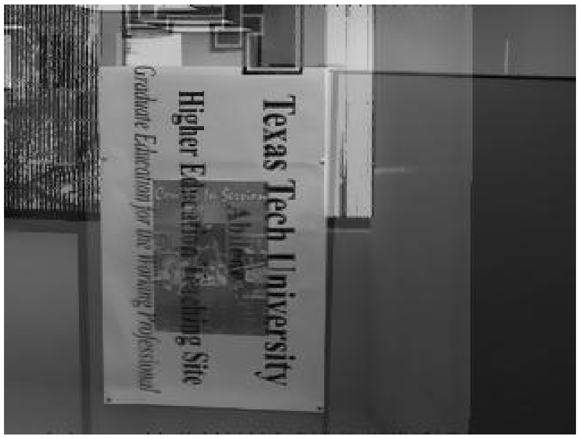
Local search correctly located all three instances of the card. The first instance of the card was located using the key feature ranked tenth on the sorted list and contained 27 pairs. The second instance was located using the third key feature and contained 25 pairs. The final instance was found using the thirteenth key feature on the ranked list, and contained 24 pairs. To generate all possible key features and run local search from the top one hundred features on the list required 6.04 seconds of processor time.



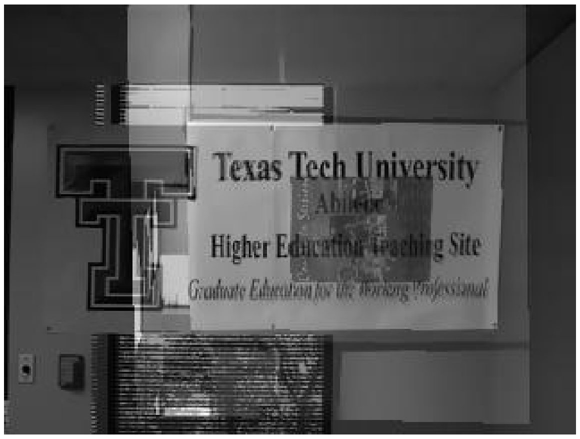
Poster 1 matched to Poster 2



Poster 2 matched to Poster 1


Texas Tech University
Higher Education Teaching Site
Graduate Education for the Working Professional

Banner 1 matched to Banner 2


Texas Tech University
Abilene
Higher Education Teaching Site
Graduate Education for the Working Professional

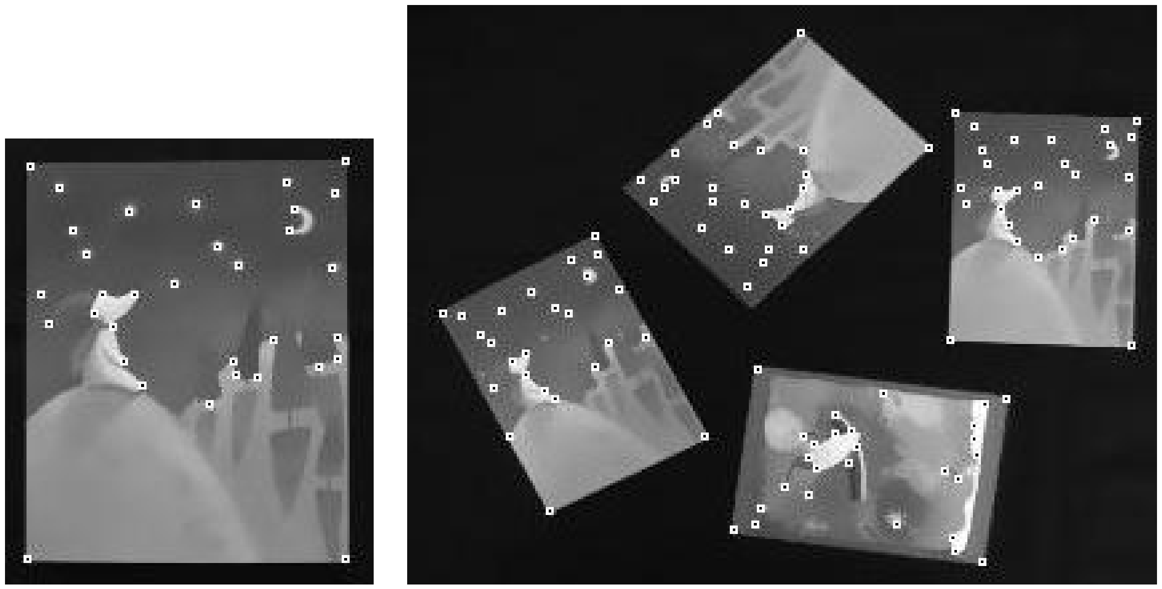
Banner 2 matched to Banner 1

Fig. 4. Data Sets for Point Matching

7 Discussion

Matching two dimensional point sets under projective transformations is much more difficult than point matching under lower dimensional transformations, for example rigid or similarity transforms. Few general algorithms have been proposed for this problem, with RANSAC being one of them [14]. We, as have others, have argued that the applicability of RANSAC is limited [11,16], and here we reviewed the analytical argument and have presented specific problems that are beyond the reach of RANSAC.

The key feature local search algorithm presented is able to solve these same example problems with relative ease, and we feel our algorithm represents a much more powerful and promising class of algorithm for projective point matching. Here it has been presented in its purest form as a tool for finding matches based upon geometry alone. Nothing has been assumed that might a priori limit the set of potentially matching points. That the algorithm performs



Christmas Card : 33 points Card Collection : 106 points

Fig. 5. Data Sets for Point Matching

well under these circumstances is evidence of its generality and power.

Some important practical tasks, including the image matching task used for illustration, naturally provide additional constraints, and incorporating these into the search process described above in order to constrain search can only improve performance. In our past work on line segment base model matching we showed how to incorporate constraints derived from approximate camera knowledge and properties derived from image gradients [12]. Our goal in this paper was to illustrate the power of a general purpose projective point matcher. A productive line of future work would be to develop a task specific incarnation of our algorithm specifically tailored to matching image pairs under the same general scenario captured by our examples.

It is important to acknowledge that others have proposed powerful iterative search algorithms for geometric feature matching, observing as we have that RANSAC is a weak algorithm against which to compare performance. For example, the softPOSIT [28] algorithm for matching three-dimensional model points against two-dimensional data sets shares much in common with our prior work on local search and the algorithm that has been presented here for projective matching. That is not to suggest the two approaches are by any means identical, they are not. Instead, what needs to be stressed is that iterative search through correspondence space guided by constant re-evaluation of the pose constraint, i.e. fitting of the model to the corresponding data, is a powerful idea, and one worthy of more attention and study. Sample code and data sets, including those presented here, can be found online at http://www.cs.ttu.edu/∼denton/pntmatch

References

[1] D. Skea, Barrodale, R. Kuwahara, R. Poeckert, A control point matching algorithm, Pattern Recognition 26 (2) (1993) 269–276.

[2] S. Chang, F. Cheng, W. Hsu, G. Wu, Fast algorithm for point pattern-matching: Invariant to translations, rotations and scale changes, Pattern Recognition 30 (2) (1997) 311–320.

[3] J. Starink, E. Backer, Finding point correspondences using simulated annealing, Pattern Recognition 28 (2) (1995) 231–240.

[4] P. David, D. DeMenthon, R. Duraiswami, H. Samet, softposit: Simultaneous pose and correspondence determination, in: ECCV (3), 2002, pp. 698–714.
URL `citeseer.ist.psu.edu/david02softposit.html`

[5] H. Baird, Model-based image matching using location, Ph.D. thesis, Princeton University (Oct. 1984).

[6] T. M. Breuel, Fast recognition using adaptive subdivisions of transformation space, in: CVPR, IEEE, 1992, pp. 445–451.

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

[8] M. Fischler, R. Bolles, Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography, Communications of the ACM 24 (6).

[9] D. A. Forsyth, J. Ponce, Computer Vision: A Modern Approach, Prentice hall, 2003.

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

[11] A. Lacey, N. Pinitkarn, N. A. Thacker, An evaluation of the performance of ransac algorithms for stero camera calibration, in: Proceedings of the 11th British Machine Vision Conference, Vol. 2, University of Bristol, Bristol, UK, 2000, pp. 646–55.

[12] R. J. Beveridge, Local search algorithms for geometric object recognition: Optimal correspondence and pose, Ph.D. thesis, University of Massachusetts (1993).

[13] D. Lowe, Object recognition from local scale-invariant features, in: ICCV99, 1999, pp. 1150–1157.

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

[15] J. A. Denton, Two dimensional projective point matching, Ph.D. thesis, Colorado State University, Ft. Collins, Colorado (2002).

[16] P. David, D. DeMenthon, R. Duraiswami, H. Samet, Evaluation of the softposit model to image registration algorithm, Tech. Rep. CAR-TR-974, University of Maryland (Feb. 2002).

[17] P. Torr, D. W. Murray, The development and comparison of robust methods for estimating the fundamental matrix, International Journal of Computer Vision 24 (3) (1997) 271–300.

[18] D. Kahl, A. Rosenfeld, A. Danker, Some experiments in point pattern matching, IEEE Transactions on Systems, Man, and Cybernetics 10 (2) (1980) 105–115.

[19] W. E. L. Grimson, D. P. Huttenlocher, On the sensitivity of the hough transform for object recognition, T-PAMI 12 (3) (1990) 255–74.

[20] P. H. S. Torr, A. Zisserman, Robust computation and parametrization of multiple view relations, in: U. Desai (Ed.), International Conference on Computer Vision, Narosa Publishing House, 1998, pp. 727–732.

[21] C. H. Papadimitriou, K. Steiglitz, Combinatorial Optimization Algorithms and Complexity, Prentice-Hall, Englewood Cliffs, NJ, 1982, Ch. Local Search, pp. 454–480.

[22] R. J. Beveridge, C. R. Graves, J. Steinborn, Comparing random starts local search with key feature matching, Tech. Rep. CS-97-117, Colorado State University, http://www.cs.colostate.edu/ (1997).

[23] D. Whitley, J. R. Beveridge, C. Graves, C. Guerra-Salcedo, Messy genetic algorithms for subset feature selection, in: Proceedings of the International Conference on Genetic Algorithms, 1997, pp. 586–575.

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

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

[26] M. G. Forsterner, A fast operator for detection and precise location of distinct points, corners and centers of circular features, in: ISPRS Intercommission Workshop, 1987, pp. 149–155.

[27] C. Harris, M. Stephens, A combined corner and edge detector, in: Fourth Alvey Vision Confrence, 1998, pp. 147–151.

[28] P. David, D. DeMenthon, R. Duraiswami, H. Samet, softposit: Simultaneous pose and correspondence determination, IJCV 59 (3) (2004) 259–284.