Optimal Matching of Planar Models in 3D Scenes

David W. Jacobs
MIT Artificial Intelligence Laboratory
545 Technology Square,
Cambridge, MA 02139

Abstract
This paper considers the problem of matching a model consisting of the point features of a flat object to point features found in an image that contains the object in an arbitrary three-dimensional pose. Once three points are matched, it is possible to determine the pose of the object. Assuming bounded sensing error, we present a solution to the problem of determining the range of possible locations in the image at which any additional model points may appear. This solution leads to an algorithm for determining the largest possible matching between image and model features that includes this initial hypothesis. We have implemented a close approximation to this algorithm, which is $O(nm\epsilon^2)$, where $n$ is the number of image points, $m$ is the number of model points, and $\epsilon$ is the maximum sensing error. We compare this algorithm to existing methods, and show that it produces more accurate results.

Introduction
In model-based object recognition, we use a precise geometric model of an object to search for that object in an image of a scene. Typically, we assume that an object is modeled by features such as lines, corners, or curvature extrema, and that comparable features can be found in the image. We assume that some bounded amount of sensing error occurs in locating these image features. Therefore, our goal is to find a hypothetical location of the object that will maximize the number of model features that will project to within error bounds of the image features.

This paper addresses this problem in the following domain. We assume that features are points, and that models are of only flat two-dimensional objects, but that these objects may be viewed from any direction in a three-dimensional scene. We also assume orthographic projection with scale, which is equivalent to assuming that the model has undergone an affine transformation.

A common approach to this problem is to form small sets of matches between image and model features, and use these to determine a hypothetical pose of the model in the scene. This pose may then be used to determine the location of other model features in the image, which may then be sought [8], [15], [11]. It is known that when projection is treated as an affine transform, matching three point features suffices to determine the pose of the model[11]. In this paper we derive a simple characterization of the possible location in the image of additional model features, assuming a bounded amount of sensing error. We call this location the potential location of a point.

This analysis has several implications. First, it may be used for the theoretical analysis of recognition systems. To this end, previous authors have addressed this problem, but have found only loose bounds on the potential location[12],[17],[9]. Second, knowing the potential location of a point tells us where to look for it when attempting to verify a hypothesis, or when performing indexing in the manner of Lamdan, Schwartz, and Wolfson[14].

However, we focus on a third application of the potential location. Its simple formulation leads to an algorithm that finds the maximum number of image and model features that may be matched within error bounds, given an initial hypothesis. This algorithm is significantly influenced by the work of Cass[5] and Baird[1]. We discuss an implementation of this algorithm that is a slight approximation, and runs in $O(nm^2\epsilon^2)$ time, where there are $m$ model features, $n$ image features, and sensing error is bounded by $\epsilon$. Of course, by exploring all initial hypotheses we may find the optimal match between image and model features, at an additional cost of $O(n^3m^3)$. We discuss experiments that demonstrate that our algorithm produces more accurate results than the technique used in alignment[11].
The Potential Location

Consider four model points, \( m_1, m_2, m_3, m_4 \), and three image points, \( i_1, i_2, \) and \( i_3 \). We match the three image points to the corresponding model points (\( i_1 \) matched to \( m_1 \) etc...). In the error-free case, the location that \( m_4 \) projects to in the image is easily found. For this analysis, we assume that sensing error causes us to misjudge the location of image points by up to \( \varepsilon \) in any direction, although the analysis is easily extended to error bounds with different shapes. This means that the true pose of the model would align each model point with a location in the image somewhere in a circle of radius \( \varepsilon \) about the corresponding image point. We determine the potential set of locations of \( i_4 \), the image of the fourth model point, if the first three points are correctly matched.

First, we review some facts about affine transformations. An affine transformation of a point is expressed by:

\[
p' = Ap + v
\]

Where \( p \) is a two-dimensional point, \( A \) is a two by two matrix, \( v \) is a translational vector, and \( p' \) is the transformed point. The projection of a planar model oriented freely in three dimensional space, using orthographic projection plus scale, is equivalent to an affine transformation\[12\].

Next, we may use any three model points as a basis, and express the remaining model points in terms of this basis. Let \( u = m_2 - m_1 \) and \( v = m_3 - m_1 \). Then, for some \( \alpha \) and \( \beta \):

\[
m_4 = \alpha u + \beta v + m_1
\]

We call \((\alpha, \beta)\) the affine coordinates of \( m_4 \). Then the affine coordinates of \( m_4 \) are invariant under affine transformations. This is shown in \[14\]. This means that when viewed from any direction, the projection of \( m_4 \) will still have the same affine coordinates with respect to the projections of \( m_1, m_2, \) and \( m_3 \).

Let us describe the sensing error with the vectors \( -e_i \). This means that the "real" location that the \( i \)th model point would project to in the absence of sensing error is \( i_i + e_i \). Our assumptions about error are expressed as: \(||e_i|| \leq \varepsilon\). Let \( u' = i_2 - i_1, v' = i_3 - i_1, \) and \( p_4 = i_1 + \alpha u' + \beta v' \). That is, if we use the match between the first three image and model points to solve for an affine transform that perfectly aligns them, and apply this transform to \( m_4 \), we will get \( p_4 \).

Let \( p_4' \) stand for the "real" location of \( m_4 \) in the image, that is, the location at which we would find \( m_4 \) in the absence of all sensing error.

\[
p_4' = (i_1+e_1)+\alpha((i_2+e_2)-(i_1+e_1))+\beta((i_3+e_3)-(i_1+e_1))
\]

When we allow the \( e_i \) to range over all vectors with magnitude less than \( \varepsilon \), this defines a region of potential locations of \( p_4' \). This region is a circle centered about \( p_4 \), of radius \( \varepsilon((1-\alpha-\beta)+|\alpha|+|\beta|) \).

The potential locations of \( i_4 \) are the set of all points within \( \varepsilon \) of the potential locations of \( p_4' \). So we find that the potential locations of the fourth image point, such that there exists some orientation and bounded sensing error that aligns the image and model points, is a circle, centered at \( p_4 \), with radius \( \varepsilon((1-\alpha-\beta)+|\alpha|+|\beta|+1) \).

This has a surprising consequence: the size of the space of potential locations of a fourth model point will depend only on the affine coordinates of the fourth point. It will not depend on the appearance of the first three model points. That is, it will not depend on the viewing direction. Even if the model is viewed almost end-on, so that all three model points appear co-linear, or if the model is viewed at a small scale, so that all three model points are close together, the size of the potential locations of the fourth model point in the image will remain unchanged.

Consequences for Analysis

Partial solutions to this problem have previously been produced in order to analyze the performance of different recognition algorithms.

In alignment\[11\], the affine transform is found after matching three image and model points. This is used to find the pose of a three-dimensional model in the scene. Remaining model points are then projected into the image, and matching image points are searched for within a radius of \( 2\varepsilon \) of the projected model points.

To analyze the accuracy of this system, Huttenlocher [12] considered the case of planar objects, as we have. He was able to show bounds on the potential location in the image of model points in certain simple cases. These bounds depended on assumptions about the image points. Here we have shown exact bounds on the potential location of model points, and we show that these bounds do not depend on characteristics of the image points.

In geometric hashing\[14\], the fact that affine coordinates are invariant under projection for planar models is used for indexing. A hash table is used to find groups of four model points with affine coordinates that are similar to the affine coordinates of four image points. These matches are then used in a Hough-type voting scheme. To perform this indexing exactly in the presence of image error, it is useful to know what set of model affine coordinates are compatible with four im-
age points. A characterization of this set would also be useful for analyzing the accuracy of geometric hashing. [17] and [9] have placed bounds on this set under specialized assumptions. We are now in a position to characterize this set precisely.

Suppose we have four model points, \( i_1, i_2, i_3 \) and \( i_4 \). Let the affine coordinates of \( i_4 \) with respect to \( (i_1, i_2, i_3) \) be \((\alpha, \beta)\), i.e. \( i_4 = \alpha(i_2 - i_1) + \beta(i_3 - i_1) + i_1 \). If a fourth model point has affine coordinates \((\alpha', \beta')\) with respect to the three other model points, we would like to know whether the model could match the image. We know that the fourth model point will match any image point within \( \epsilon(1 - \alpha' - \beta' | + |\alpha'| + |\beta'| + 1) \) of \( p'_4 = i_1 + \alpha'(i_2 - i_1) + \beta'(i_3 - i_1) \). So the model and image can match if the distance from \( p'_4 \) to \( i_4 \) is less than or equal to this value. That is, iff:

\[
|| (i_1 + \alpha'(i_2 - i_1) + \beta'(i_3 - i_1)) \leq \epsilon(1 - \alpha' - \beta' | + |\alpha'| + |\beta'| + 1)
\]

This equation tells us the range of affine coordinates that a model quadruple may possess to match a particular image quadruple. [10] points out that this equation describes an ellipse in the affine coordinate space under most conditions, and uses this analysis to perform an error analysis of geometric hashing. The immediate consequence of this result is that geometric hashing could exactly compensate for error by having each image quadruple access an ellipse in the affine coordinate hash table.

**Optimal Matching**

We now present a method for finding the pose of the model that maximizes the number of matches between model and image points. Cass has solved this problem in the case of purely two dimensional matching, that is, when a two dimensional model is transformed with a two-dimensional rotation and translation[5], and is scaled[6]. To do this, Cass makes use of Baird's[1] insight that when a model point is matched to an image point, this places a constraint on the possible transformations that will align the points to within error bounds. Cass notes that these constraints divide the space of all transformations into cells. Each of these cells corresponds to a set of possible transformations of the model in which the same model points are within error bounds of the same image points. Cass shows that there are a polynomial number of these cells, and he and Breuel[2] present algorithms for enumerating them.

We take a similar approach, examining error space instead of transformation space. We assume that we have matched three image points to three model points, and then find the maximum number of additional matches that may be added to this initial hypothesis. By considering all possible initial matches of three image points to three model points we could find the best overall match of model and image points.

For simplicity, we now assume that this error is bounded by a square of width \( 2\epsilon \). That is, for \( i_1 = (i_1x, i_1y) \) and sensing error of \( e_1 = (e_1x, e_1y) \), the "real" location of the first image point, in the absence of sensing error is \( p'_1 = (p'_{1x}, p'_{1y}) \), and \(|i_1x - p'_{1x}| \leq \epsilon \) and \(|i_1y - p'_{1y}| \leq \epsilon \).

Having matched three points, the above sections tell us where to look in the image to find potential matches for a fourth point. Matching a fourth model and image point constrains the possible values of \( e_1 \), \( e_2 \) and \( e_3 \). Suppose we find an image point, \( i_4 \), that could match a fourth model point, \( m_4 \). We again let \( p'_4 \) stand for the "real" location of \( m_4 \) in the image, that is, the location it would project to with no sensing error. For \( i_4 \) to be a valid match with \( m_4 \) we must have: \(|p'_4 - i_4| \leq \epsilon \). From above, we know that:

\[
p'_4 = (i_1 + e_1x) + \alpha((i_2 + e_2x) - (i_1 + e_1x)) + \beta((i_3 + e_3x) - (i_1 + e_1x))
\]

This tells us that:

\[
|p'_4 - i_4 + e_2x\alpha + e_3x\beta + e_1x(1 - \alpha - \beta)| \leq \epsilon
\]

where we recall that: \( p_4 = i_1 + (i_2 - i_1)\alpha + (i_3 - i_1)\beta \). Similarly,

\[
|p'_4 - i_4 + e_2x\alpha + e_3x\beta + e_1x(1 - \alpha - \beta)| \leq \epsilon
\]

These two equations define a volume in error space, that is, the six-dimensional space of \((e_1 \times e_2 \times e_3)\). This volume is bounded by four hyperplanes. The volume of allowable error vectors that would align the image and model points, then, is bounded by the 16 hyperplanes:

\[
p_4 - i_4 + e_2x\alpha + e_3x\beta + e_1x(1 - \alpha - \beta) = \pm \epsilon
\]

\[
p_4 - i_4 + e_2x\alpha + e_3x\beta + e_1x(1 - \alpha - \beta) = \pm \epsilon
\]

and: \( e_{ix} = \pm e, e_{iy} = \pm e \) for \( i = 1, 2, 3 \).

So, given an initial hypothesis of three matches, each additional match between an image and model point defines a volume in the six dimensional space of \((e_1 \times e_2 \times e_3)\). Each point in this volume corresponds to a set of error vectors in the first three points that would align all four matches to within error bounds. If two of these volumes, produced by two matches, intersect, then there is a set of error vectors, and hence an affine transform, that would align both the matches to within error bounds. If we determine these volumes for all
feasible additional matches between image and model points, the place where the maximum number of these volumes intersect provides us with a set of error vectors that maximizes the number of matches between image and model points, given an initial hypothesis.

In the next section we will describe an algorithm that provides an approximate answer to this problem. Here, however, we note that an exact algorithm exists that solves the problem in polynomial time.

There are at most \( mn \) additional matches that we must consider adding to our basis matches. Each of these matches corresponds to a volume of feasible error vectors in the six dimensional error space. Each of these volumes is bounded by a constant number of hyperplanes. Collectively, the \( O(nm) \) hyperplanes associated with the additional matches divide the error space into cells. Given a point in each cell we may determine in \( O(nm) \) time the number of volumes that contain that cell. Algorithms are known for enumerating the cells formed by \( n \) hyperplanes in a \( d \) dimensional space in \( O(n^d) \) time. Therefore, we can find a point where the maximum number of volumes intersect in \( O(n^m m^7) \) time.

### An Approximate Algorithm

We have implemented an algorithm that uses a slight approximation to find the point where the maximum number of these volumes intersect. We use a six dimensional array of length \( 2c \) in each dimension to represent the error space. Each point in the array represents a hypercube in error space. For each match, we make an entry at each bucket that overlaps the volume in error space for which the matched points may align.

The overall algorithm proceeds as follows. We are given a set of \( 2d \) model points, \( 2d \) image points, and matches between three of these points. We determine the affine coordinates of the other model points, and for each model point, \( m_4 \), we look for potentially matching image points in a square of the appropriate width about \( p_4 \). To intersect the volumes in error space implied by these matches, we represent the six dimensional space of \( (e_{1x} \times e_{1y} \times e_{3x} \times e_{3y} \times e_{2x} \times e_{2y}) \), with a six dimensional array. That is, we divide the error space up into buckets with a dimension of one pixel. Then, for each additional match, we make an entry in this array at each bucket that intersects the volume of possible error values that would align these newly matched points. This entry consists of the identity of the matched points. In the end, each bucket tells us all matches of image to model points that can be aligned to within error bounds by some set of error vectors represented by that bucket. This is a slight approximation, since matches may be aligned by error vectors that are represented by the same bucket without being aligned by the same error vectors.

If we simply count the number of matches in each bucket, we will double count when two model points match the same image point for a particular transformation of the model, or when two image points can match the same model point. Ideally, we would like to find the largest match between image and model points in which each point is used only once. [13] has shown that we get a good approximation to this by taking the minimum of the number of different model points and the number of different image points matched for a specific transformation of the model. So we find the bucket that maximizes the minimum of the number of different image and model points matched in that bucket.

This algorithm considers at most \( mn \) matches. For each match, it fills at most \( (2c)^6 \) buckets in the table. Finding the best bucket involves examining at most \( mn \) matches in each of \( (2c)^6 \) buckets. So the algorithm is \( O(nm^{10}) \).

Previous approaches have sampled transformation space[4] or examined transformation space using a generalized Hough transform. The primary advantage of formulating our constraints as hyperplanes in error space instead of in transformation space is that error space is smaller, and may be more efficiently sampled.

### Experiments

We have run experiments to compare the performance of our algorithm to that of alignment. We have implemented an alignment algorithm for planar objects. In this algorithm, we use a match of three image and model points to determine the hypothetical location of all remaining model points. We then look in the image for matching points, searching a square of width \( 4c \) centered about the hypothetical location of the projected model point. As in our algorithm, we evaluate a hypothesis by taking the minimum of the number of image points matched and the number of model points matched. Except for this addition, and the use of square error bounds instead of circular ones, this algorithm is identical to the one presented in [11], although we apply it only to planar models.

To find point features in an image we locate the corners formed by making straight line approximations to the edges found in the image. We use the Canny edge detector[3] with a sigma of 1, and a split and merge algorithm for finding straight line approximations[16].

To create a model, we took a picture of a flat object in isolation. Figure 1 shows this image, and the lines and corners found. Figure 2 shows an image of this
Figure 1: On the left, the image used to create a model of a widget. On the right, line approximations to the edges found in the image. Circles show where corners are found. These are used as point features.

Figure 2: On the left, an image of the widget and a different, occluding widget. The viewpoint is different than the image used to form the model. On the right, the lines and corners found in this image.

Figure 3: The three features circled have been correctly matched. The pose of the model is determined from these matches, and the resulting location of the model is shown. This set of matches leads to quite a lot of error in the location of many model features. As a result, alignment found only 6 additional matches. Our algorithm found 22 additional matches from this initial hypothesis.

To show that our algorithm produces more accurate results than alignment, we need to know how many matches each algorithm produces when given an initial triple of matches that is correct, and when given a set of matches that is incorrect. To find correct matches, we matched twelve model points to image points by hand (although there were more than twelve correct matches in the image), and then selected random triples of these matches. To find incorrect matches, we selected random triples of matches that did not include any of these correct matches, and that did not match the same image or model point twice. For each initial hypothesis formed in this way, we could then determine the number of matches found by our algorithm and by alignment.

We generated 100 correct hypotheses. The average number of points matched by alignment was 17.8, and by our algorithm was 25.6. The standard deviation of the number of matches found was 6.8 for alignment, and 6.2 for our algorithm.

We generated 1055 incorrect hypotheses. The average number of points matched by alignment was 3.8, and by our algorithm was 4.1. The standard deviation of the number of matches found was 3.1 for alignment, and 3.3 for our algorithm.

To judge the overall effectiveness of the two approaches, we may compare the probability that a random correct hypothesis will produce fewer or the same number of matches than the number produced by a random incorrect hypothesis. This allows us to estimate the likelihood of each algorithm making a mistake. For alignment, this probability is .073, while for our algorithm it is .002.

We can see that while our algorithm found slightly more matches using incorrect hypotheses, it more than compensated by finding 7.8 more matches on average with a correct hypothesis. Figure 3 illustrates the greater reliability of our algorithm. It shows a correct hypothesis that produced considerable error in the location of the other model points. Using this hypothesis, alignment finds 6 additional matches, while our method finds 22.

Alignment performed less well than our algorithm because an insufficient number of matches were found using correct initial hypotheses. One might wonder if this could be fixed by using more generous error bounds. To test this, we ran alignment on the same set of test matches, using error bounds that were rectangles of width 8ε instead of 4ε. This produced an average of 26.8 additional matches when starting with a correct
hypothesis. However, these had a standard deviation of 8. Also, on average, starting with an incorrect hypothesis, 8.2 additional matches were found, with a standard deviation of 5.7. Overall these figures do not lead to improved performance. The probability that an incorrect hypothesis would look as good or better than a correct one was 0.072, approximately the same as with error bounds of 4ε.

Conclusion

In the case of affine projection we have found a simple way of characterizing the potential location of a fourth model point in an image, after we have matched three points. This leads us to a polynomial time algorithm that finds the object pose that aligns it with the maximum number of image features. We have implemented an approximation to this algorithm, and shown that it performs more accurately than an alignment technique.

Acknowledgements

This paper describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by an Office of Naval Research University Research Initiative grant under contract N00014-86-K-0685, and in part by the Advanced Projects Agency of the Department of Defense under Army contract number DACA76-85-C-0010 and under Office of Naval Research contract N00014-85-K-0124.

References