# Object Recognition Using Point Uncertainty Regions as Pose Uncertainty Regions

Ilan Shimshoni *,[1]

*University of Haifa, Department of Management Information Systems, Haifa 31905, Israel*

Aviva Sasporta

*Technion, Department of Industrial Engineering and Management, Haifa, 32000, Israel*

**Abstract**

In this paper, a recognition algorithm based on point features is presented. In this algorithm sets of hypothesized matches between model and image points are generated. From them the pose of the object is estimated and stored in a lookup table. When two similar poses are found the pose is assumed to be correct and the hypothesis is verified. The main contribution of this paper is that poses and their uncertainties are represented by the uncertainty regions of the projections of several 3D points which are circles in the image. These uncertainty regions are due to the measurement uncertainty of the image features which result in uncertainty in the recovered pose. When two poses are consistent, the pairs of uncertainty regions of the same 3D point will have a non-empty intersection. The algorithm exploits the fact that these uncertainty regions can be computed easily and accurately. The algorithm has been implemented and tested on real images.

*Key words:* Object recognition, Uncertainty Regions, Pose Estimation

* Corresponding Author
*Email address:* ishimshoni@mis.haifa.ac.il (Ilan Shimshoni).

[1] Ilan Shimshoni was supported in part by the Israeli Ministry of Science, Grant no. 2104. He was also supported by the fund for the promotion of research at the Technion.

## 1 Introduction and Related Work

Model-based object recognition systems usually belong to one of three classes of algorithms. The first and more common type has been studied extensively and systems based on this concept have been presented for example in [1–3]. In this type of algorithm, which is known as the *Alignment* method, a minimal set of image features is matched to model features. From this hypothesized match the pose of the object is computed. The correctness of the match is verified by projecting the model on the image plane and searching for image features in the neighborhood of the projected model-features. If sufficiently many projected model features are matched to image features the match is assumed to be correct and the object recognized. This process is repeated until a correct match is found and verified. This algorithm is an example of the Random Sample Consensus (RANSAC) paradigm [1]. If the number of model features is $m$ and the number of image features is $n$, then the number of possible hypotheses generated by such an algorithm is $O(m^3n^3)$. The verification stage requires performing range queries on the image looking for image features in the neighborhoods of projected model features. The complexity of the verification stage per hypothesized match is $O(m \log n)$. The number of hypotheses can be reduced by devising a smart method to match model features to image features. In [3], this was achieved by maintaining for each model feature several of its appearances and matching them to neighborhoods of features recovered in the image.

The second method searches for clusters in pose space [4–7]. In [4,5], each possible match between a model feature and an image feature yields a constraint on the pose, which is represented geometrically as a hyper-surface in the pose space. The pose space is then searched for poses which are compatible with the most number of matches. From the $mn$ possible matches only a small fraction of them (less than $m$) are correct. The pose space search is performed in a top-down manner. At first a hyper rectangle in pose space corresponding to all possible poses is created and all constraint surfaces that intersect the hyper rectangle are maintained. At each step the hyper rectangle is cut by a plane parallel to one of the axes and the test is repeated on the two sub-boxes. Boxes which do not satisfy enough constraints are discarded. The problem with this method is that only after many subdivisions of the box are performed, it is possible to detect that there is no pose within this sub-box that is consistent with enough matches. Therefore, a very expensive search process has to be performed scanning the search tree to a very deep level.

In [8], this limitation is overcome in the following way. Hypothesized poses are generated by matching several line features. A box in pose space around the computed pose is generated and a consistent pose is searched for using the method described above. As the size of the box is relatively small, incor-

rect hypotheses can be discarded fast and a pose which is consistent with a sufficient number of matches can be found efficiently.

The more standard method of pose clustering involves sets of matches from which a pose can be computed. In [6,7], for matches of size three, the pose is computed and pairs of such matches are searched for which yield the same pose. When care is taken to verify if two poses are indeed compatible, a correct hypothesis can be detected when only a small number of correct poses have been generated. In [7], a six dimensional (the dimension of the pose space) lookup table is maintained and similar poses are stored in the same entry in the table. When two poses are verified to be compatible, a complete pose verification is performed. In [6], a lookup table is not maintained and all pairs of poses and their uncertainty regions in pose space are tested for compatibility. This solution requires less space but more pose compatibility checks need to be performed. The algorithm described in [6] deals with the weak perspective projection model. Extending such a method for the full perspective model would require estimating the errors in pose space for this model as described in [9].

Another method known as pose clustering [10] generates many hypothesized poses and detects clusters in pose space under the assumption that these are clusters of poses generated from correct matches. The advantage of this method is that an accurate expensive analysis of the compatibility of pairs of poses does not have to be performed under the assumption that under any reasonable distance metric, clusters of correct poses will be formed. The disadvantage of this method is that the system is able to detect a cluster only when enough correct poses have been generated. To overcome this problem parallel pose clustering algorithms have been suggested. In [11], the hypothesized poses are created in parallel using different processors which update the data-structure of potential clusters in parallel.

We have reviewed methods which use matches of one feature and methods which use matches of three features. It is possible to also use matches of size two. The advantage of such a method is that less hypotheses exist. On the other hand such hypotheses do not yield a complete pose. In [12], this problem is solved by building a Hough transform for unknown rotation angles and translation values. Each hypothesis votes for all unknown parameter values consistent with it. The correct pose is detected by finding peaks in the Hough space.

Algorithms of the third type do not generate hypotheses from matches, verify or store them. Instead, they solve the matching problem and the pose estimation problem together. This is done by finding a pose which minimizes an error function. This technique was used for object recognition [13–15] and for structure from motion problems [16]. This technique replaces the combinato-

rial complexity of the other two methods by a function minimization problem which might not converge to its global minimum.

Our method belongs to the second type of algorithms. Our goal is to generate a small number of hypotheses and be able to detect a correct match after a minimal number of correct poses have been generated. In algorithms of the second type the main problem is how to efficiently find pairs of similar hypotheses, and how to easily assure that the difference in the recovered poses can be explained by the uncertainty in the feature points measured in the image. The advantage of these types of algorithms over the first type is that even though twice as many hypotheses have to be checked (until the correct pose is found twice), each hypothesis does not have to be verified but only has to be stored in a lookup table. This can be done more efficiently and the computational complexity per hypothesis is independent of the number of model and image points.

In designing an algorithm of this type three problems have to be addressed. First, given a hypothesized match the pose uncertainty region which is due to measurement errors in the image has to be computed. Second, when given two such hypotheses and their respective pose uncertainty regions, a method has to be developed to check if they are compatible (i.e. that their pose uncertainty regions have a non-empty intersection). And finally, if hypothesized poses are kept in a lookup table, how to design the lookup table such that it is easy to store new hypotheses and that the table remains low-dimensional. The dimension of the table has a dramatic effect on the space requirements of the algorithm. A six-dimensional table, for example, yields a table of size $h^6$ where $h$ is the resolution of each coordinate. Moreover, if the diameter of the uncertainty region of a pose is $D$ (out of $h$), the pose has to be stored into $D^6$ cells of the table increasing the space and time requirements of the algorithm.

The naive solution to this problem which can be considered is to estimate the pose uncertainty regions directly in the six-dimensional pose space and maintain them in a six-dimensional lookup table. The problem with this solution is that computing the pose uncertainty region in a six-dimensional non-linear space is complicated and non-elegant. Besides, a six-dimensional lookup table is too large and computing which cells in the lookup table intersect the pose uncertainty region is also complicated adding to the complexity of the algorithm.

We therefore turned to point uncertainty regions which were actually developed for RANSAC/Alignment type algorithms but are used here for a different purpose. We will therefore first review their use in these types of algorithms and then present the way that they are used here.

In [2,1], three image points are matched to three model points. This match is

used to compute the object's pose [17]. The model points are then projected to the image plane and in the neighborhood of each projected model point corresponding image points are searched for. The question that needs to be answered is: what is the correct size of neighborhood in which to search for an image point. This problem has been addressed in [18,19] for match sizes of three or more. It has been shown that the size of the uncertainty region in which to search is determined by the number of points matched, the uncertainty in the measurements of the image points, the 3D coordinates of the model points involved and the pose of the object.

In this paper, we will present a method which maintains all the poses and their uncertainty regions generated so far in a lookup table such that compatible poses will be stored in the same entry in the table. The main contribution of this method is the simple representation of the pose uncertainty regions. This representation can be easily computed and the compatibility between two such poses is also easily tested. In addition, the dimension of the lookup table does not have to be six but can be determined by the user depending on space availability and the tradeoff between the speed of updating a smaller table and the increase in the number of false positives generated as a result.

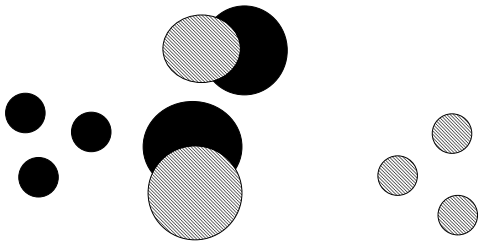


Fig. 1. Pose uncertainty regions represented by point uncertainty regions. Two pairs of image point triplets (hatched and black) are matched to model points yielding two hypothesized poses. The point uncertainty regions for two other 3D points are computed for both poses found by the two matches. These poses are compatible because the point uncertainty regions intersect.

The main idea, which is illustrated in Figure 1, is as follows. As mentioned above, methods presented in [18,19] estimate the uncertainty region in which a 3D point will be projected when given a match of three or more points. The uncertainty region found is circular and its radius is computed. The uncertainty in the projected position of the 3D point is due to the uncertainty in pose which is due to the uncertainty in the image point measurements used to compute the pose. Therefore, instead of trying to compute the pose uncertainty region we can use the point uncertainty regions to verify directly that poses are compatible since, when given two compatible poses, the uncertainty regions of the projections of 3D points will intersect in the image. Thus, a lookup table of the uncertainty circles of the projections of several 3D points is maintained (which do not have to be actual points in the model). These circles are maintained in a $2k$-dimensional table (where $k$ is the number of 3D

points projected). When a new hypothesis is generated, its circles are entered into the table and compared to hypotheses already present in the same entries. Testing if two hypotheses are compatible is simply done by testing if $k$ pairs of circles intersect. The hypotheses are generated in a probabilistic order exploiting the "probabilistic peaking effect" [20–23,6] improving the computational complexity of the algorithm.

The paper continues as follows. In Section 2, we describe the method to compute the point uncertainty region. A method to exploit these results for object recognition is described in Section 3. The probabilistic method for hypothesis ranking is described In Section 4. In Section 5, we present our implementation of the algorithm and certain design issues that we faced. Experimental results obtained on simulated and real data are presented in Section 6. We conclude in Section 7.

## 2 Point Uncertainty Regions

In this section we present a short overview on the topic of point uncertainty regions and how to estimate them. More details can be found in [18,19].

Given a match of three model points $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{m}_2)$ to three points from an image under the weak-perspective projection model $(\mathbf{i}_0, \mathbf{i}_1, \mathbf{i}_2)$, the pose of the object can be computed [17]. The pose is represented by two values $H_1$ and $H_2$ which give the tilt of the plane spanned by the three model points with respect to the image plane (See Figure 2). Given an additional model point $\mathbf{m}_3$ its coordinates can be expressed as a function of the coordinates of $\mathbf{m}_0$, $\mathbf{m}_1$, and $\mathbf{m}_2$ by solving the following vector equation for the "extended affine coordinates", $(\alpha, \beta, \gamma)$, of $\mathbf{m}_3$.

$$\begin{aligned}\mathbf{m}_3 &= \alpha(\mathbf{m}_1 - \mathbf{m}_0) + \beta(\mathbf{m}_2 - \mathbf{m}_0) \\ &+\gamma(\mathbf{m}_1 - \mathbf{m}_0) \times (\mathbf{m}_2 - \mathbf{m}_0) + \mathbf{m}_0.\end{aligned} \tag{1}$$

Substituting the representations of the three points in image coordinates into (1) yields that the image location $\mathbf{i}_3$ of $\mathbf{m}_3$ is

$$\begin{aligned}\mathbf{i}_3 = (&\alpha(x_1 - x_0) + \beta(x_2 - x_0) \\ &+\gamma((y_1 - y_0)H_2 - (y_2 - y_0)H_1) + x_0, \\ &\alpha(y_1 - y_0) + \beta(y_2 - y_0) \\ &+\gamma(-(x_1 - x_0)H_2 + (x_2 - x_0)H_1) + y_0),\end{aligned} \tag{2}$$

where $x_i$ and $y_i$ are the image coordinates of $i$-th point measured in the image. An error analysis of this equation yields that if we assume that the detected

image points $\mathbf{i}_0, \mathbf{i}_1, \mathbf{i}_2$ lie within a circle of radius $\epsilon$ from the projected model point, the position of $\mathbf{i}_3$ lies within a circle whose radius is parameterized by the extended affine coordinates and the pose parameters $H_1$ and $H_2$.

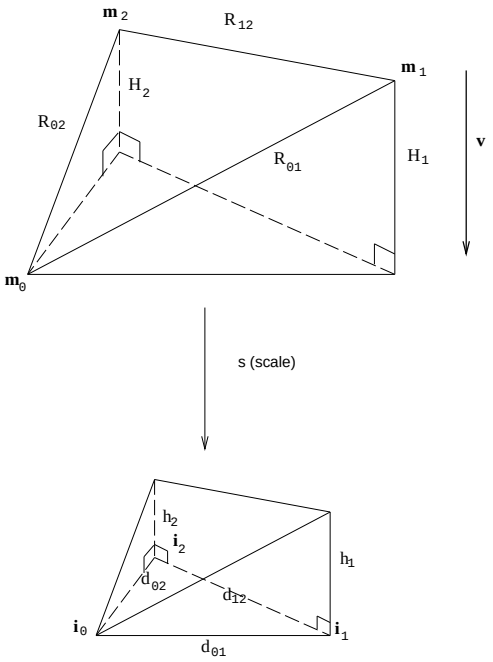



Fig. 2. Model points $\mathbf{m}_0$, $\mathbf{m}_1$, and $\mathbf{m}_2$ undergoing orthographic projection plus scale to produce image points $\mathbf{i}_0$, $\mathbf{i}_1$ and $\mathbf{i}_2$ (adapted from [17]).

This method has been extended for matches of more than three points. In this case, each sub-match of size three gives an estimate for the position of the projection of the model point and the size of the uncertainty region. These are dependent measurements of the position of the projected point. These estimations are combined taking into account the dependency between them yielding a much more accurate estimate of the position of the projection of the model point in the image.

## 3 Proposed Approach

One of the classical methods in Artificial Intelligence for solving problems is the "generate (hypothesis) and test" method. In our setting, hypotheses are generated by matching triplets of model points to image points and are verified by projecting model points on the image and searching for points in the image within the uncertainty regions. Here a different approach is proposed. Hypotheses are generated in an order sorted by a-priori probability (described in the next section). The poses which are consistent with these hypotheses are stored in a data-structure. Instead of trying to verify each hypothesis, a hypothesis is verified only when two hypotheses yield compatible poses. The

assumption is that random poses are compatible very rarely and therefore if two poses are compatible they are probably correct. When such a pair is found, a verification step of a standard alignment algorithm is used to test its correctness. Thus, when comparing the two types of algorithms, our algorithm generates on average more hypotheses but an alignment algorithm has to verify all of them whereas our algorithm only verifies a negligible amount of hypotheses.

The questions that have to be answered are: how to store these poses and their uncertainty regions, how to locate efficiently possible candidates for compatible poses, and how to test the compatibility of two poses.

To perform the above task, point uncertainty regions are used. Several 3D points $\mathbf{P}_1, \cdots, \mathbf{P}_k$ are chosen. These points should lie within the convex-hull of the object to be recognized. These points do not have to be actual feature points of the object and are not detected in the image, but as they lie within the convex hull of the object, their projection will lie inside the image of the object. Thus, the coordinates of a projection of such a point is bounded by the size of the image. This fact is used to bound the size of the lookup table.

When given a hypothesized match between three or more model points to image points the point uncertainty regions for these points are calculated and stored in a $2k-$dimensional matrix representing a Cartesian product of $k$ images. In each entry in the matrix there is a list of all hypotheses whose $k$ projected point uncertainty regions intersect a small $2k-$dimensional box. When entering a new hypothesis into the table, if the list into which the hypothesis is entered is not empty its uncertainty circles are compared to the circles of the hypotheses which are already in the list. This only requires to perform $k$ circle intersection tests. If all tests are successful we assume that the two poses are compatible and perform a more detailed verification of the hypothesis. Usually the list will be empty and even when not, with high probability not all $k$ circle pairs will have non-empty intersections.

The reason that this method is viable is because when there exists a pose which yielded both hypotheses then it must project the $k$ 3D points into points in the intersections of their uncertainty circles. Thus, the pair of hypotheses will be detected. The converse is not always true and depends on the number of points $k$. When $k$ is at least three and an intersection of all $k$ regions is found we can compute a compatible pose by choosing three of the $k$ 3D points, choose points in their respective intersection regions and compute the pose from the match between these points and their respective 3D points.

This does not mean that $k$ must be greater or equal three. Choosing a smaller $k$ only means that the algorithm might detect some false alarms which will have to be verified. The verification costs of these false alarms are offset by

the savings in time and space due to keeping a much smaller lookup matrix. The actual value of $k$ that has been chosen and the reasons for choosing it and other parameters of the algorithm are described in Section 5.

In conclusion, we have found a simple method to store and compare pose uncertainty regions using point uncertainty regions.

## 4 Probabilistic Match Ranking

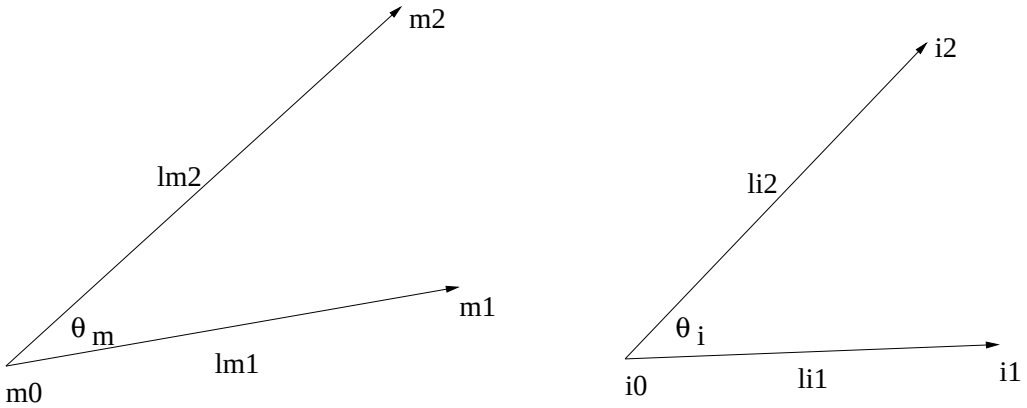



Fig. 3. Probabilistic peaking effect: Given a model point triplet and an image point triplet, the peaking effect states that measurements in the image such as angles ($\theta_i$) and ratios of lengths ($\rho_i = l_{i2}/l_{i1}$) are close to measurements performed on their corresponding model points ($\theta_m$ and $\rho_m = l_{m2}/l_{m1}$).

A very important factor in the computational cost of running the algorithm is the order in which hypotheses are generated. A large computational speedup can be achieved when more probable hypotheses are tested first. To achieve this goal the "probabilistic peaking effect" is exploited. This effect states that angles and ratios of lengths measured in the image will with high probability be close to the actual angles and ratios measured between their respective model points (Figure 3). This means that a high percentage of viewing directions (other components of the pose do not affect ratios and angles) on the viewing sphere, will yield values close to the real values. As the three points approach collinearity the effect increases and when they are collinear this effect becomes an invariant (i.e. collinearity and length ratios in this case are invariant to pose).

Thus, the peaking-effect is a double-edged sword. When approaching collinearity the measured values are quasi-invariant. This means that values close to the ones measured in the model can be measured in a very large segment of the viewing-sphere. Therefore, the pose can not be determined accurately. The practical effect of this is that the point uncertainty circles become very large and thus are not useful for matching poses because they are compatible

with many hypotheses. Therefore model and image triplets which are nearly collinear are discarded by the algorithm in the first stage.

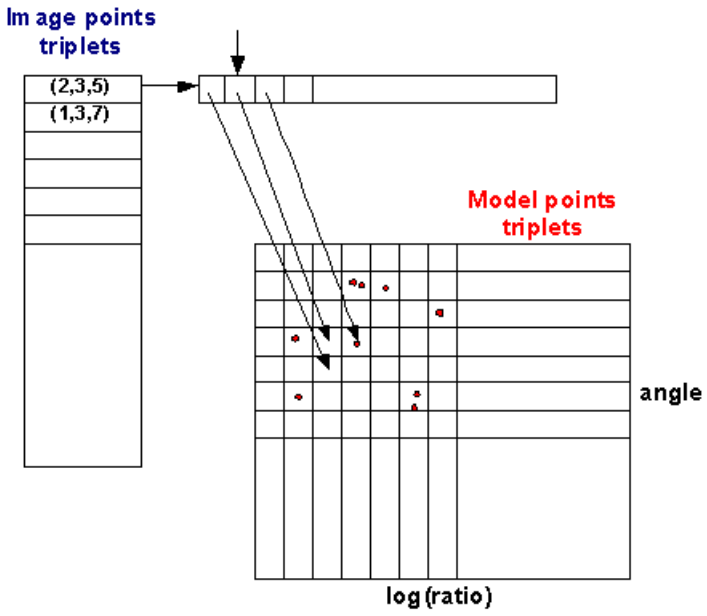



Fig. 4. Probabilistic Hypothesis Ranking: Image points triplets are stored in a heap. Each triplet has a list of cells in the angle×log(ratio) table ordered by probability in which to search for model matches. The model triplets are stored in such a table.

For all other image and model triplets an efficient method has to be developed that will generate the hypotheses ordered by probability. The algorithm has to deal with the fact that only a small fraction of the hypotheses will be generated by the algorithm before it terminates successfully. Thus, for each image point triplet at each step of the algorithm only the model triplet with the highest probability has to be known.

The algorithm consists of two stages: an off-line stage and an on-line stage. In the off-line stage for an angle and ratio of lengths measured in the image, a lookup table which estimates the probability that the angle and ratio of lengths measured in the model will be in a certain range is computed. Such a lookup table can not be built for each angle measured in the image. Therefore a table is computed for 20 angle values $\alpha_l$ equally spaced between $0 - \pi$ and the ratio value of one. Each entry in the table estimates the probability

$$Pr(\beta_i \leq \beta_m \leq \beta_{i+1}, \rho_j \leq \rho_m \leq \rho_{j+1} | \alpha_{im} = \alpha_l, \rho_{im} = 1),$$

where the values of $\beta_i$ and $\rho_j$ tessellate the angle×log(ratio) space into equally sized rectangles. Using Bayes law the prior probability that the match is correct can be deduced. If the real ratio of lengths is not 1 then the ratio values

of the boundaries of the regions are simply multiplied by the correct ratio. All model triplets are stored off-line in another table indexed according to their angles and ratios. At runtime, each image triplet is attached to a list of regions in the angle-ratio space and their probabilities, sorted by probability and a pointer which points to the first entry in the list (the one with the highest probability). Thus, when an image triplet requires model triplets with certain angle ratio values they can be found by accessing the relevant entry in model triplet table. The image triplets are kept in a heap where the image triplet with the region with the highest probability is on top.

Using these data-structures which are illustrated in Figure 4, the algorithm described in Algorithm 1 generates hypotheses in probabilistic order.

**Algorithm 1** Recognition algorithm

```
1: while heap not empty do
2:    Take the image triplet from the top of the heap.
3:    Extract from the array of region coordinates the model table entry with
      the highest probability.
4:    Access the entry in the model table.
5:    for all model triplet in the entry in the table do
6:       Generate a hypothesis between the image triplet and the model
         triplet.
7:       Compute the point uncertainty region for the k points.
8:       Store the hypothesis in the entries in the pose uncertainty table which
         intersect the k circles.
9:       for all hypotheses already in those entries in the table do
10:         Check if their k circles intersect the k circles of the new hypothesis.
11:         if found then
12:            verify the correctness of the pose
13:            if verified then
14:               exit algorithm successfully.
15:            end if
16:         end if
17:      end for
18:   end for
19:   Move the pointer which points to the region which needs to be tested
      to the next region on the list.
20:   Update the heap such that the highest unprocessed (image triplet,model
      table region) is on top.
21: end while
```

Probabilistic ordering has been used in [23] to generate hypotheses for an "alignment" algorithm. There the goal was to find one correct hypothesis, whereas in the algorithm presented here at least two correct poses have to be found. This yields a difference between the probabilistic orderings of the

two algorithms. In [23], an image triplet is chosen and all high probability model matches with that image triplet are tested, ordered by probability. Thus, if there is a match between the image triplet and a model triplet, it will be detected if it has a high enough probability. The hypothesis will then be verified and the program terminates. Whenever a hypothesis is tested and found to be incorrect, the probability of the other hypotheses with the same image triplet increases and therefore it is correct to continue to test other hypothesized matches of the same image triplet. Here at each step the model-image match with the highest probability is generated. Only when the second correct hypothesis has been found the program terminates. Thus, continuing to generate other matches with the selected image triplet is not necessarily the best strategy. Instead at each step the most probable hypothesis is generated.

## 5 Algorithm and Implementation

We have implemented the algorithm and tested it on simulated and real images. In building such an implementation two issues have to be addressed in order to yield an efficient algorithm. First, the number of 3D points $k$ has to be chosen and second, the structure of the lookup table has to be determined.

### *5.1 Choosing k*

We first studied the problem of choosing $k$ the number of 3D points $\mathbf{P}_i$ whose uncertainty circles will be used to find consistent poses. The tradeoff here is between the amount of computation that has to be performed for each hypothesis and the number of false positive hypothesis pairs which will be detected by the algorithm and will have to be verified.

| $k$ | Expected Probability | Actual Probability | $\frac{p_{k-1}}{p_k}$ |
|---|---|---|---|
| 1 | $8.5e^{-4}$ | $8.5e^{-4}$ | |
| 2 | $7.3e^{-7}$ | $4.5e^{-5}$ | 18.9 |
| 3 | $6.2e^{-10}$ | $1.5e^{-5}$ | 3.0 |
| 4 | $5.3e^{-13}$ | $6.5e^{-6}$ | 2.3 |

Table 1
Given a pair of random poses what is the probability that $k$ pairs of uncertainty circles will intersect. The second column is under the assumption that these events are independent of each other ($p_k = p_1^k$). The third column is the results of tests run showing that this assumption is very inaccurate.

To test this the following experiment whose results are presented in Table 1 was run. Random pairs of poses were generated and the percentage of them that yielded intersections in the uncertainty regions of one, two or more projected 3D points was recorded. Had each circle intersection test been performed on different model points been independent, it could be expected that the probability that $k$ intersections would be found would be $p_1^k$ where $p_1$ is the probability that the first pair of circles intersected. This did not happen because these events are dependent. Even so the results show that the probability that two random poses will cause even two pairs of uncertainty regions to intersect at random is very low. The contribution of the next 3D points to reducing the probability for a random match is small due to the dependence between the points. Thus, the main contribution for reducing the numbers of hypotheses in the same entry is achieved by the first two points and the others have a limited effect. Therefore, more than two points should be used by the algorithm only if the computational cost needed to maintain them is small.

### *5.2 Lookup table design*

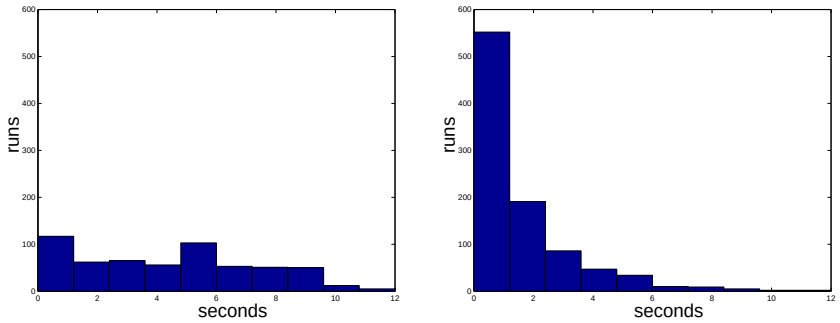
Another question which has to be addressed is how to construct the lookup table. Assuming that the value for $k$ was chosen to be 4, there are several options for constructing the lookup table. One eight-dimensional table could be built, or two four-dimensional tables or even four two-dimensional tables. The higher the dimension of the table, the more space is required to maintain it. In addition, when each hypothesis is stored in the table it will have to be stored in more entries of the table because an uncertainty region in $2k$ dimensions whose diameter is $D$ cells wide will intersect $O(D^{2k})$ cells in the table which increases the space requirements and processing time. On the other hand less false alarms will be detected.

When using two four-dimensional tables a pair of hypotheses is considered valid only when they are in the same entry of the first table and in the same entry of the second table (and then pass the circle intersection tests). As mentioned above, even a four-dimensional table (two points) has a very low false alarm rate. Thus, it was decided to build only one four-dimensional table and maintain for each hypothesis also the point uncertainty circles for two additional points which are used to verify the consistency of two poses only when they are consistent on the first two points. The reason for that was that the computational cost and space requirements for maintaining the second table were much higher than the cost in the rare cases when a pair of hypotheses passed the intersection test of the first two points and the pose had to be verified using the other two points.

## 6 Experimental Results

The algorithm was implemented and experiments were conducted to verify its applicability and test its various components.

In the first experiment the running of algorithm with and without the probabilistic ordering of hypotheses was compared. A thousand runs of the algorithm were performed for both cases. As can be seen from the runtime histograms shown in Figure 5, the probabilistic ordering has a major impact on reducing the runtime of the algorithm. Under the random ordering of hypotheses, the runtime of the algorithm is distributed fairly uniformly between 0-12 seconds whereas when the probabilistic ordering is used, the runtime is highly weighted towards 0-2 seconds. Keeping in mind that generating each hypothesis is more costly under the probabilistic ordering makes these results even more impressive.




(a) (b)

Fig. 5. Comparison of running the algorithm without (a) probabilistic ordering and with (b). Probabilistic ordering improves considerably the running time and reduces the number of hypotheses generated before a correct match is found.

In the second experiment the algorithm was tested on images of several objects. The first input to the algorithm was an image of a clock shown in Figure 6. The model has 30 points. The Susan corner detector [24] was applied to the image yielding 26 corners. 16 of them were projections of model points. Of the first 30 pairs of hypotheses found, 14 of them were close to the correct pose.

A surprising feature of the algorithm is presented in Table 2. In the experiment we assigned to each model point its correct match in the image. However, in some cases, hypothesis pairs yielding quite accurate estimates of the pose are generated from incorrect matches. This happens when the matched image point is close to the correct image point. In the table the first 30 results obtained by the algorithm are shown. The number of correct and incorrect matches for each of the two hypotheses is shown in the first column. The number of correct and incorrect poses with that match combination is shown in the other two columns.

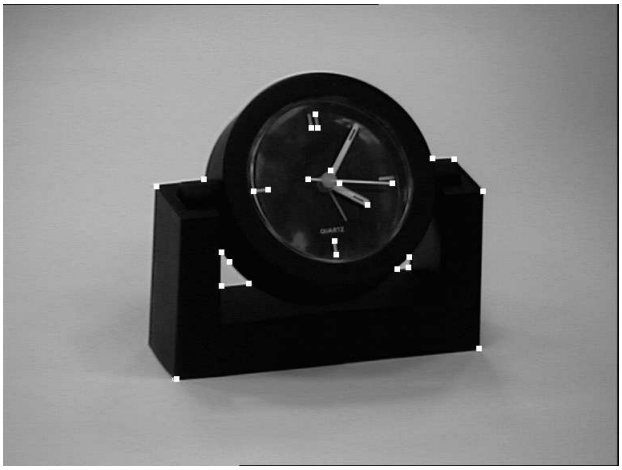


Fig. 6. Experiment setup. The image of the clock used in the experiment. The results of the corner detector which were the input to the algorithm are overlayed over the original image.

| Number of Correct Matches | Correct | Incorrect |
|---|---|---|
| (0,0) | - | 9 |
| (0,1) | - | 3 |
| (0,2) | - | 1 |
| (0,3) | - | - |
| (1,1) | - | 1 |
| (1,2) | - | 1 |
| (1,3) | - | - |
| (2,2) | 7 | 1 |
| (2,3) | 5 | - |
| (3,3) | 2 | - |
| | 14 | 16 |

Table 2
The first 30 results obtained in the first experiment: The table shows how many of the points in the matches were correct and if the resulting pose is "correct" (i.e. close to the correct pose). The results show that even if some of the matches are incorrect but the points in the image are close to the correct positions a correct pose is found.

In Figure 7 we plotted the positions of the projections of the four model points which were used in the experiment for correct pose pairs. As can be seen they form clusters due to the fact that the results are all very close to the correct pose. Once this estimate for the pose has been detected and verified it can be

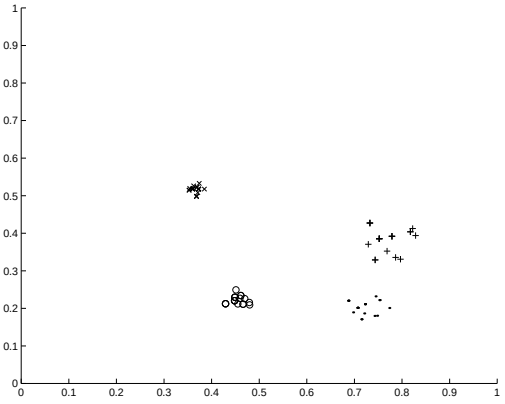


Fig. 7. The positions of the projections of the four model points used by the algorithm which were found by the pairs of matches which were close to the correct pose. Note that the points are clustered meaning that even when the matches are not entirely correct the correct pose is found.

optimized using the other image points which are projections of model points.

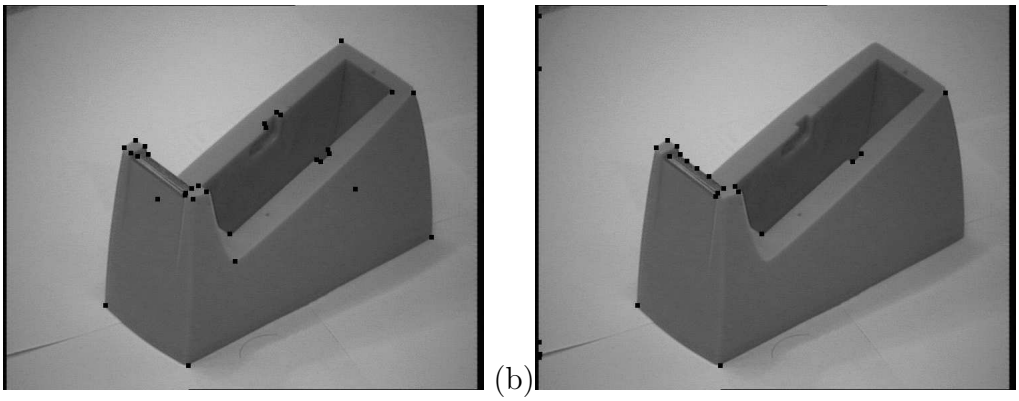



Fig. 8. Experimental setup. (a) The 30 model points overlayed over the image (including occluded model points). (b) The results of the corner detector which were the input to the algorithm.

In another experiment an image of a tape dispenser was used. The object has 28 model points which are overlayed over the image as seen in Figure 8(a). The corner detector recovered 26 corners (Figure 8(b)) of which 14 were projections of the model points. In this case the phenomenon of mismatched points is even more prevalent. This is because quite a few of the image features form clusters. Thus, even when the "wrong" image point is matched to a model point, the pose recovered is quite accurate. As can be seen in Table 3, correct poses were found even when all the image points were misclassified. This only means that matching a model point to an image point close to its correct projection is not actually a mismatch.

In a final experiment an image of a stapler, shown in Figure 9, was correctly identified by the algorithm. The model consists of 26 points and 24 points were detected in the image. 16 of those points correspond to model points. This

| Number of Correct Matches | Correct | Incorrect |
|---|---|---|
| (0,0) | 3 | 3 |
| (0,1) | - | 4 |
| (0,2) | - | - |
| (0,3) | - | - |
| (1,1) | 3 | 3 |
| (1,2) | 9 | 2 |
| (1,3) | 1 | - |
| (2,2) | - | - |
| (2,3) | - | - |
| (3,3) | 2 | - |
| | 18 | 12 |

Table 3
The first 30 results obtained in the second experiment: The table shows how many of the points in the matches were correct and if the resulting pose is "correct" (i.e. close to the correct pose). The results show that even if some of the matches are incorrect but the points in the image are close to the correct positions a correct pose is found.

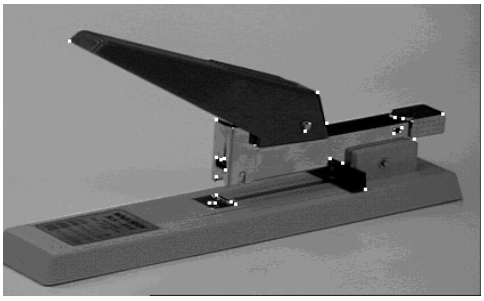


Fig. 9.
An image of a stapler with the features detected in the image overlayed over it.

image was used to test the robustness of the algorithm. In the experiment, additional (besides the original 8) 0-20 randomly selected image points were given to the algorithm. Even with these additional image points the correct hypothesis pair was always found first. The running times of the algorithm with 8-28 outlier image points are shown in Figure 10. The running time of the algorithm on the original data set (8 outliers) was 0.57 seconds on a Pentium 3 1000MHz laptop. Running times for various numbers of outliers are plotted in green. The increase in runtime is due to the increase in the number of possible

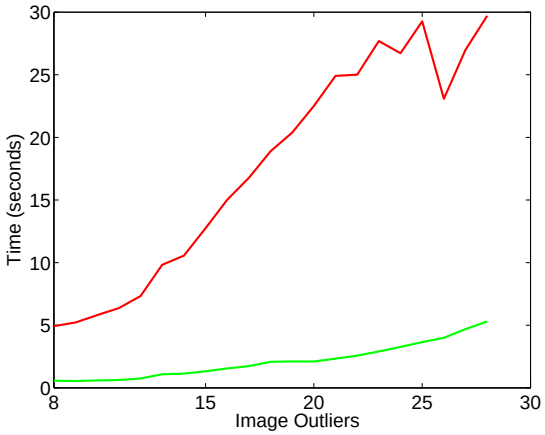



Fig. 10.
The running times of the algorithm for the stapler image with 8-28 incorrect image points added to it. The green curve plots the running time until the first (correct) pose pair is found and the red curve the running time until the next (incorrect) pose pair is detected.

hypotheses. The red plot shows the time required to find the second hypothesis pair which is incorrect. The huge difference in time is due to the difference in the probability to generate a random pair of consistent hypotheses and a pair of correct consistent hypotheses. This can be exploited for designing a multi-model recognition system as is demonstrated in the following test.

| | Time | Hypothesis No. |
|---|---|---|
| Stapler | 0.55 | 11216 |
| Clock | 20.50 | 372160 |
| Tape Dispenser | 13.50 | 276869 |

Table 4
Running times for correct and incorrect image data to the stapler model. The table shows the running times in seconds and the number of hypotheses generated until the first consistent hypothesis pair is found.

The algorithm was run using the stapler model and using three sets of 24 image points taken from the three images. The results are presented in Table 4. The running time differences between the correct and incorrect image data until a first consistent pair was found is enormous. Figure 11 shows the prior probabilities of the basic pose hypotheses in the three runs. Note that the distributions are quite similar. This is not surprising because the fraction of correct hypotheses within the hypotheses being tested is negligible. Thus, the distributions of the probabilities are similar even though the number of correct hypotheses with high prior probability is relatively high.

The conclusion from this experiment is that the simplest way to build a recognition system with several models is to run the algorithm in parallel on all the

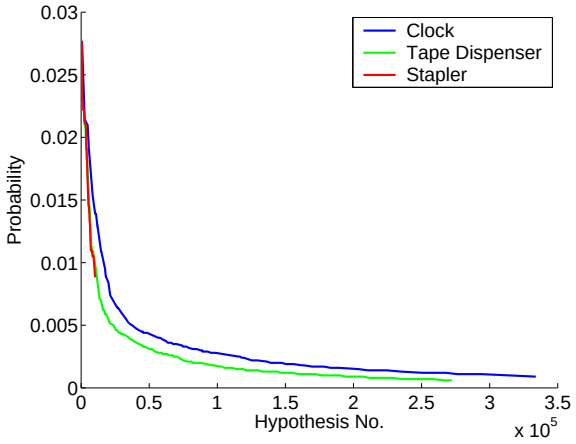



Fig. 11.
Prior probabilities for the three runs. Note the similarities in the probability densities of the correct(Stapler) and incorrect runs.

different models. The correct model will be found quite quickly. An alternative which could be considered is to build one program which deals with all models. However, as the distribution of the probabilities of the basic hypotheses is similar for correct and incorrect models, the running times will be basically the same as the more simpler implementation which runs on single models in parallel.

## 7 Conclusion

In this paper we have presented a recognition algorithm which searches for pairs of consistent matches between model and image points which yield the same pose. Poses are stored in a table represented by the projections of several 3D points onto the image and their uncertainty regions which represent the uncertainty in the pose. If all pairs of uncertainty regions intersect the two matches are declared consistent and yield very similar poses. This algorithm exploits the fact that we are able to compute uncertainty regions of the projections of 3D points simply and accurately and when sufficiently many pairs of regions have a non-empty intersection the two hypotheses are consistent.

This algorithm shows that point uncertainty regions are a very powerful tool to represent pose uncertainty regions in a simple and efficient manner. Projections of 3D points can also be used to represent poses in a traditional pose clustering system such as [10].

The probabilistic ordering of the hypotheses has a dramatic effect on the running time of the algorithm by processing the correct hypotheses earlier. One of the consequences of the probabilistic ordering is that the algorithm

can be scaled up easily to deal with more than one model by simply running the algorithm in parallel, one process per model yielding equivalent results to running one multi-model algorithm. The main problem which still remains is that the running time of the algorithm is linear in the number of models. Therefore, any existing method to prune the number of possible models (e.g., appearance based) can be applied to speedup the algorithm.

In the experiments we showed that the algorithm can deal with large amounts of clutter. Still clutter has a detrimental effect on the performance of the algorithm. Therefore, segmentation and grouping procedures can help in reducing the runtime of the algorithm. Moreover, algorithms which can yield better prior probabilities for matches between image to model features such as the one described in [3] can also speedup the algorithm.

## References


[1] M. Fischler, R. Bolles, Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography, Comm. of the ACM 24 (6) (1981) 381–395.

[2] D. Huttenlocher, S. Ullman, Recognizing 3D solid objects by alignment with an image, Int. J. of Comp. Vision 5 (2) (1990) 195–212.

[3] V. Lepetit, J. Pilet, P. Fua, Point matching as a classification problem for fast and robust object pose estimation, in: Proc. IEEE Conf. Comp. Vision Patt. Recog., 2004, pp. II: 244–250.

[4] T. Cass, Feature matching for object localization in the presence of uncertainty, in: Proc. Int. Conf. Comp. Vision, 1990, pp. 360–364.

[5] T. Breuel, Fast recognition using adaptive subdivision of transformation space, in: Proc. IEEE Conf. Comp. Vision Patt. Recog., 1992, pp. 445–451.

[6] I. Shimshoni, J. Ponce, Probabilistic 3D object recognition, Int. J. of Comp. Vision 36 (1) (2000) 51–70.

[7] D. Thompson, J. Mundy, Three-dimensional model matching from an unconstrained viewpoint, in: IEEE Int. Conf. on Robotics and Automation, Raleigh, NC, 1987, pp. 208–220.

[8] F. Jurie, Solution of the simultaneous pose and correspondence problem using gaussian error model, Comp. Vis. Im. Understanding 73 (3) (1999) 357–373.

[9] V. Kyrki, D. Kragic, H. Christensen, Measurement errors in visual servoing, in: IEEE Int. Conf. on Robotics and Automation, 2004, pp. 1861–1867.

[10] G. Stockman, Object recognition and localization via pose clustering, Comp. Vis. Graph. Im. Proc. 40 (3) (1987) 361–387.

[11] W. Austin, A. Wallace, Object location by parallel pose clustering, Comp. Vis. Im. Understanding 72 (3) (1998) 304–327.

[12] G. Hausler, D. Ritter, Feature-based object recognition and localization in 3d-space, using a single video image, Comp. Vis. Im. Understanding 73 (1) (1999) 64–81.

[13] J. Hornegger, H. Niemann, Statistical learning, localization, and identification of objects, in: ICCV, 1995, pp. 914–919.

[14] S. Gold, A. Rangarajan, C. Lu, S. Pappu, E. Mjolsness, New algorithms for 2d and 3d point matching: Pose estimation and correspondence, Pattern Recognition 31 (8) (1998) 1019–1031.

[15] P. David, D. Dementhon, R. Duraiswami, H. Samet, Softposit: Simultaneous pose and correspondence determination, Int. J. of Comp. Vision 59 (3) (2004) 259–284.

[16] F. Dellaert, S. Seitz, C. Thorpe, S. Thrun, Structure from motion without correspondence, in: Proc. IEEE Conf. Comp. Vision Patt. Recog., 2000, pp. II: 557–564.

[17] T. D. Alter, 3D pose from 3 points using weak-perspective, IEEE Trans. Patt. Anal. Mach. Intell. 16 (8) (1994) 802–808.

[18] T. D. Alter, D. W. Jacobs, Error propagation in full 3D-from-2D object recognition, in: Proc. IEEE Conf. Comp. Vision Patt. Recog., Seattle, Washington, 1994, pp. 892–898.

[19] I. Shimshoni, On estimating the uncertainty in the location of image points in 3D recognition from match sets of different sizes, Comp. Vis. Im. Understanding 74 (3) (1999) 163–173.

[20] T. Binford, T. Levitt, W. Mann, Bayesian inference in model-based machine vision, in: Workshop on Uncertainty in Artificial Intelligence, 1987.

[21] J. Ben-Arie, The probabilistic peaking effect of viewed angles and distances with application to 3-D object recognition, IEEE Trans. Patt. Anal. Mach. Intell. 12 (8) (1990) 760–774.

[22] J. B. Burns, R. S. Weiss, E. M. Riseman, View variation of point-set and line-segment features, IEEE Trans. Patt. Anal. Mach. Intell. 15 (1) (1993) 51–68.

[23] C. F. Olsen, Probabilistic indexing for object recognition, IEEE Trans. Patt. Anal. Mach. Intell. 17 (5) (1995) 518–521.

[24] S. Smith, J. Brady, SUSAN - a new approach to low level image processing, Int. Journal of Computer Vision 23 (1) (1997) 45–78.