ROR: Rejection of Outliers by Rotations

Amit Adam, *Student Member*, *IEEE*,
Ehud Rivlin, *Member*, *IEEE*, and
Ilan Shimshoni, *Member*, *IEEE*

Abstract—We address the problem of rejecting false matches of points between two perspective views. The two views are taken from two arbitrary, unknown positions and orientations. Even the best algorithms for image matching make some mistakes and output some false matches. We present an algorithm for identification of the false matches between the views. The algorithm exploits the possibility of rotating one of the images to achieve some common behavior of the correct matches. Those matches that deviate from this common behavior turn out to be false matches. Our algorithm does not, in any way, use the image characteristics of the matched features. In particular, it avoids problems that cause the false matches in the first place. The algorithm works even in cases where the percentage of false matches is as high as 85 percent. The algorithm may be run as a postprocessing step on output from any point matching algorithm. Use of the algorithm may significantly improve the ratio of correct matches to incorrect matches. For robust estimation algorithms which are later employed, this is a very desirable quality since it reduces significantly their computational cost. We present the algorithm, identify the conditions under which it works, and present results of testing it on both synthetic and real images. The code for the algorithm is available through the World Wide Web.

Index Terms—Correspondence problem, feature matching, false matches, outliers, outlier rejection, robust estimation.

1 Introduction

Correspondence of points in two images is a fundamental problem in computer vision and has been the subject of intensive research for over two decades (see, for example, chapter 16 in [7]). This problem is sufficiently hard that even recent matching algorithms (for some examples, see [10], [14], [9], [16]) may still make errors. There are two types of errors. The first is to miss a match—i.e., not to match the images of a feature which appears in both views. As long as this does not happen prohibitively too often, this type of error is not alarming. The second type of error in which a false match is found—i.e., the pixels which image two different scene points are matched—is more severe. For instance, in motion estimation, even one false match may affect the results of non-robust algorithms (for example, the least-squares-based algorithms [8]) as to make them useless. For this reason, in recent years, work has been devoted to estimation based on robust methods. In these works, even if there are false matches in the input, a meaningful result may still be obtained. One of the examples for these kind of algorithms is the work by Zhang et al. [16]. In this work, the fundamental matrix is robustly estimated by minimizing the Least Median of Squares (LMedS) criterion. The search for the proper matrix is carried out by randomly choosing subsets of eight matching pairs. Other examples of robust algorithms may be found in [15], [13], [12].

In this paper, we address the issue of reducing the number of false matches. The percentage of correct matches out of the total number of matches directly affects the computational cost of many robust algorithms. In many of these algorithms, subsets of the matching pairs are chosen and computations are carried out on these subsets (for example, RANSAC in [10], choice of subsets of eight matches in [16]). In order to receive the correct answer, some of the subsets chosen should contain only correct matches. The higher the percentage of correct matches, the lower the number of subsets one has to choose in order to ensure (with a certain probability) obtaining some "pure" subsets.

We present an algorithm which accepts as input a list of matches of which many may be false (even up to 70-85 percent) and outputs a subset of these matches in which the number of false matches is much lower. In other words, the algorithm throws away matches in a such a way that false matches are rejected with a high probability while correct matches are rejected with a low probability. We do this by actively rotating one of the views, an action which "shakes" away the outliers.

Our algorithm is different from other robust approaches in several ways. First, our algorithm does not carry out computations using the outliers (false matches) and then rejects them based on the results obtained (as is done in the RANSAC paradigm, for example). In other words, our algorithm is application-independent. Second, our algorithm's complexity does not increase with the percentage of outliers in the input. In approaches where subsets of features are sampled, the number of samples required increases dramatically with the percentage of outliers. In our algorithm, the complexity does not depend on the percentage of outliers. Last, our algorithm may be applied even in cases where the percentage of outliers is as high as 85 percent. This percentage may be above the breakpoint of many other robust algorithms.

We emphasize that *we do not assume knowledge of the relative rotation and translation between the two input views*. The only fact we use in our algorithm is that the point matches which are correct are perspective images of the same point in space.

The ideal use for our algorithm is as a postprocessing stage after any automatic matching algorithm. This postprocessing of the list of matches is likely to throw away a large portion of the wrong matches. This allows the user to apply a wider range of robust techniques to further process the matches (for the specific application in mind) and at a lower computational cost. As will be demonstrated, we have used ROR on input from a very naive SSD matcher and have raised the percentage of correct matches from 20-30 percent to around 70-80 percent.

2 The ROR Algorithm

2.1 Problem Definition and Terminology

Given two perspective views of a static scene, we associate a world coordinate system with each view in the standard fashion: the origin is at the center of projection and the z axis is the viewing direction. The two coordinate systems are related by a rotation R and a translation $\vec{t}$ as usual: $\hat{P} = R(P - \vec{t})$. We assume the focal length f of the camera is known.

The correspondence algorithm outputs a set of corresponding points on the two screens

$$S = \{l_i = (p_i, \hat{p}_i) = (x_i, y_i, \hat{x}_i, \hat{y}_i) | i = 1, \ldots, n\}.$$

We will sometimes refer to the pairs l_i as segments and think of the line segments between the point p_i and the point $\hat{p}_i$ (as if the two points were on the same screen). Some of the matches in the scene

- *A. Adam is with the Department of Mathematics, Technion—Israel Institute of Technology, Haifa 32000 Israel. E-mail: amita@cs.technion.ac.il.*
- *E. Rivlin is with the Department of Computer Science, Technion—Israel Institute of Technology, Haifa 32000 Israel. E-mail: ehudr@cs.technion.ac.il.*
- *I. Shimshoni is with the Department of Industrial Engineering and Management, Technion—Israel Institute of Technology, Haifa 32000 Israel. E-mail: ilans@ie.technion.ac.il.*

Manuscript received 10 Nov. 1999; revised 20 July 2000; accepted 27 Sept. 2000.
Recommended for acceptance by P. Meer.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 110916.

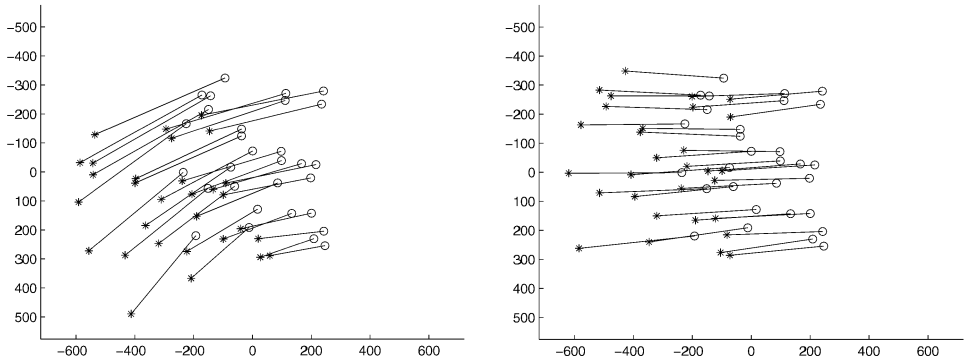


Fig. 1. The points p_i and $\hat{p}_i$ joined by lines. (a) Original images A and B. (b) Image A and rotated image $\tilde{B}$.

are false or outliers—i.e., p_i and $\hat{p}_i$ are not the images of the same point in space.

We will shortly use rotations as a method to generate new scenes from the given scene S. By this we mean the following: It is well-known that given a certain image, one may reproduce the effect of changing the 3D orientation of the camera. Let us denote the two views by view A and view B. From these two views, we compute the segments $s_i = \hat{p}_i - p_i$. Let us now rotate view B by $\tilde{R}$ to obtain the view $\tilde{B}$ with the points $\tilde{p}_i$. We may now compute the segments $\tilde{s}_i = \tilde{p}_i - p_i$. Thus, we obtained a rotated scene $\tilde{S} = \{\tilde{s}_i\}$ from the given scene $S = \{s_i\}$.

2.2 The ROR Algorithm

The basic idea for rejecting false matches is simple and may be demonstrated by looking at Figs. 1a and 1b. In these figures, we plotted the segments $\{s_i\}$ by plotting the points $\{p_i\}$ and $\{\hat{p}_i\}$ on the same screen and joining them by a line. Fig. 1a depicts point matches between two images A and B which are related by a certain translation and rotation of the camera. Fig. 1b depicts matches between image A and image $\tilde{B}$ which was obtained from image B by rotation. Clearly, one can see that in this case, the segments are all pointing at approximately the same direction. A segment that would point in a different direction is likely to be a false match or an outlier.

We exploit the following idea: **Given two images A and B with point correspondences, one may rotate the image B to obtain an image $\tilde{B}$ such that the point correspondences between A and $\tilde{B}$ will create segments which point at the same direction. Incorrect matches between the two images will lead to segments which do not point at that direction. Thus, we may identify the incorrect matches by identifying deviation of the segment direction from the most frequent direction.**

We now present the ROR (rejection of outliers by rotations) algorithm. We will explain the different steps of the algorithm in the following section.

ROR Algorithm

1. Start with correspondence segments $l_i = (p_i, \hat{p}_i)$ between image A and image B.
2. Perform K random rotations of image B. For the kth rotation compute the set of segments $S_k = \{s_i^k = (p_i, \hat{p}_i^k)\}$. For each set S_k compute the following:
   a. Find the mode of the distribution of segment directions. Denote it by m_k.
   b. Let d_i^k denote the angle between the direction the segment s_i^k points at, and the frequent direction m_k. Save this distance from the mode.
   c. Find the minimal window around the mode m_k which contains some percentage q of the directions of segments in the set S_k. Denote the window width by w_k.
3. Order the K rotations according to increasing widths w_k. Define as "good" rotations only the first G rotations in that order. Let $\mathcal{G}$ denote the indices of the "good" rotations.
4. For each feature i, compute the average distance of the segment containing the feature from the mode:

$$\bar{d}_i = \frac{1}{G}\sum_{k\varepsilon\mathcal{G}} d_i^k$$

5. Compute the mode T of the average distances $\bar{d}_i$.
6. Increase T by a small constant α and use it as a threshold on the average distances $\bar{d}_i$. The correct matches are those segments s_i such that $\bar{d}_i \leq T + \alpha$ for some fixed α.

The computation of the mode of the distribution of segment directions is done by using the Mean Shift Mode Estimator [6], [3] and a technique described in [4] for avoiding local maxima.

3 Geometric Interpretation of the ROR Algorithm

3.1 3D Interpretation of the Directions of Segments

In this section, we will explain the reasoning behind the algorithm which was presented in the previous section. We first relate the directions that the line segments point at to some characteristic of 3D feature vectors. Then, we reason by using the 3D vectors.

3.1.1 Working with 3D Feature Vectors

Let $s_i = \hat{p}_i - p_i$ denote the ith segment. By direct computation,

$$s_i = \begin{pmatrix} f\frac{\hat{X}}{\hat{Z}} - f\frac{X}{Z} \\ f\frac{\hat{Y}}{\hat{Z}} - f\frac{Y}{Z} \end{pmatrix} = \frac{f}{Z\hat{Z}} \begin{pmatrix} Z\hat{X} - X\hat{Z} \\ Z\hat{Y} - Y\hat{Z} \end{pmatrix}. \quad (1)$$

Let $\vec{\mu}_i$ denote the 3D vector

$$\vec{\mu}_i = P_i \times \hat{P}_i = (-(Z\hat{Y} - Y\hat{Z}), Z\hat{X} - X\hat{Z}, X\hat{Y} - Y\hat{X})^t. \quad (2)$$

We can see that the projection of $\vec{\mu}_i$ on the xy plane is orthogonal to the vector s_i. From this, we immediately obtain:

Claim: Suppose we are given two corresponding pairs $(p_i, \hat{p}_i)$ and $(p_j, \hat{p}_j)$. These pairs give rise to the segments $s_i = p_i - \hat{p}_i$ and $s_j = p_j - \hat{p}_j$. Then, the angle between the vectors s_i, s_j is equal to the angle between the xy projections of $\vec{\mu}_i$ and $\vec{\mu}_j$.

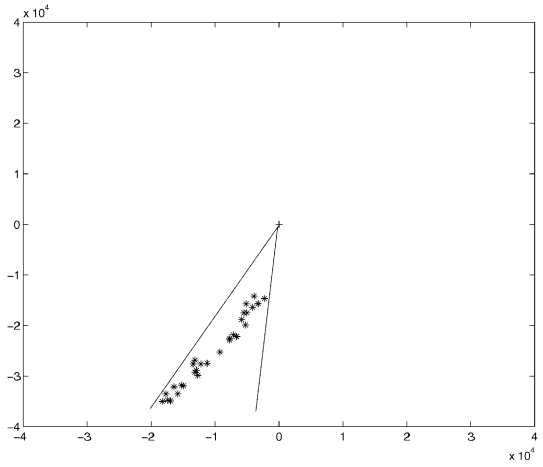


Fig. 2. The angular width of the set C as viewed from the origin.

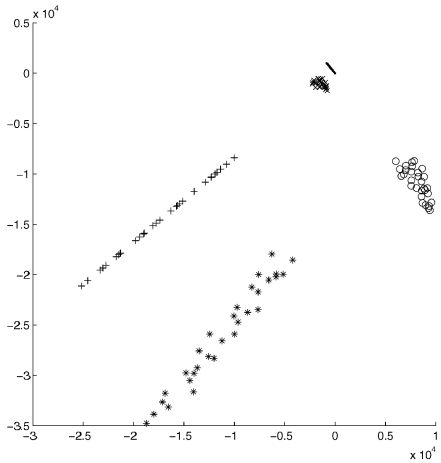


Fig. 3. The four components of μ_i and their sum. $\circ$ is the projection of g_1 term. $+$ is the projection of g_2 term. $\times$ is the projection of g_3 term. $\bullet$ is the projection of g_4 term. $*$ is the point in C (sum of the four terms).

Let C denote the projection of the set $\{\vec{\mu}_i\}$ on the xy plane. C is a "cloud" of points. We may now derive the following conclusion (see Fig. 2).

Conclusion: The maximal angle between any two segments s_i and s_j representing correct matches, is equal to the angle at which the set C is viewed from the origin.

We now want to express the points in the set C as the sum of certain expressions. Assume the rotation R acts on a vector by rotating it around the axis $\hat{k}$ at an angle θ. Then, by Rodriguez's formula [5], we have

$$R\vec{v} = \cos\theta\vec{v} + \sin\theta(\hat{k}\times\vec{v}) + (1-\cos\theta)(\hat{k}\cdot\vec{v})\hat{k}. \tag{3}$$

We apply this formula to $\vec{v} = P - t$ and after some manipulations obtain

$$\vec{\mu} = P\times\hat{P} = P\times R(P-t) \tag{4}$$

$$= -\cos\theta(P\times t) + \sin\theta(P\cdot(P-t))\hat{k} - \sin\theta(P\cdot\hat{k})(P-t) + \tag{5}$$

$$(1-\cos\theta)(\hat{k}\cdot(P-t))(P\times\hat{k}). \tag{6}$$

Let us denote the four terms in (6) by

$$\begin{aligned} g_1 &= -\cos\theta(P\times t)\\ g_2 &= \sin\theta(P\cdot(P-t))\hat{k}\\ g_3 &= -\sin\theta(P\cdot\hat{k})(P-t)\\ g_4 &= (1-\cos\theta)(\hat{k}\cdot(P-t))(P\times\hat{k}). \end{aligned} \tag{7}$$

Each vector $\vec{\mu}_i$ is the sum of these four vectors. The x, y projection of $\vec{\mu}_i$ is the ith point in the set C. This point is the sum of the x, y projections of each of the four terms.

3.1.2 Small Angular Width of C

We will now reason why, in some cases, the set C is likely to occupy a small angular region when viewed from the origin. By the conclusion above, in these cases, all the inlier segments point in the same direction. This characteristic behavior of the inliers allows us to differentiate them with respect to the outliers.

The set C is the xy projection of the 3D points μ_i as in (6). The vectors $\hat{k}$ and t are fixed and the point P changes ($P = P_i$). Let us now suppose that $\hat{k}$ is a vector in the xy plane—$\hat{k} = (k_x, k_y, 0)^t$. Let us also assume the points P_i have a dominant Z component. This is a reasonable assumption for cameras which are not with an exceptionally wide field of view. Fig. 3 shows the xy projections of the four terms $g_1, \ldots, g_4$ defined in (7), for a typical set $\{\mu_i\}$.

The long straight line (marked by pluses) is the projection of g_2 : all the points are in the direction of the xy components of $\hat{k}$—i.e., (k_x, k_y). The scalars multiplying $\hat{k}$ are $\sin\theta(P_i\cdot(P_i - t))$. As the points P_i vary, these scalars are of order $\mathcal{O}(Z^2)$ and, thus, have large variability, leading to a long line.

Let us now look at the first term:

$$g_1 = -\cos\theta(P\times t) = -\cos\theta\begin{pmatrix} t_zY - t_yZ\\ -t_zX + t_xZ\\ t_yX - t_xY\end{pmatrix}. \tag{8}$$

If we assume $Z \gg X, Y$ and that t_z does not dominate t_x, t_y, we obtain

$$(g_1)_{1,2} \approx -\cos\theta Z\begin{pmatrix} -t_y\\ t_x\end{pmatrix}. \tag{9}$$

Indeed, we may see in the figure that the term g_1 gave rise to a second, shorter line (marked by circles) which is in the fixed direction $(-t_y, t_x)^t$. The scalars multiplying that vector are of order $\mathcal{O}(Z)$ and, hence, the variability is not as large as the in the previous term, resulting in a shorter line.

The projections of the third term are shown by xs and are seen to be near the origin, i.e., small in magnitude. This is a result of the assumption that $\hat{k}$ has no Z component and that the dominant component in the P_is is in the Z direction: the scalar $P\cdot\hat{k}$ is a factor reducing the magnitude of g_3.

The fourth term g_4 is

$$g_4 = (1-\cos\theta)(\hat{k}\cdot(P-t))(P\times\hat{k}) \tag{10}$$

$$= (1-\cos\theta)(\hat{k}\cdot(P-t))\begin{pmatrix} k_zY - k_yZ\\ -k_zX + k_xZ\\ k_yX - k_xY\end{pmatrix} \tag{11}$$

and under the assumption $k_z = 0$ its projection on the xy plane is

$$(g_4)_{1,2} \approx \mathcal{O}(XZ + YZ)\begin{pmatrix} -k_y\\ k_x\end{pmatrix}. \tag{12}$$

Hence, it is a line orthogonal to the projection of g_2, but much shorter. This line may be seen in the figure (marked by dots).

The set C (marked by asterisks) is the vector sum of the four terms described above and, hence, is dominated by the sum of the

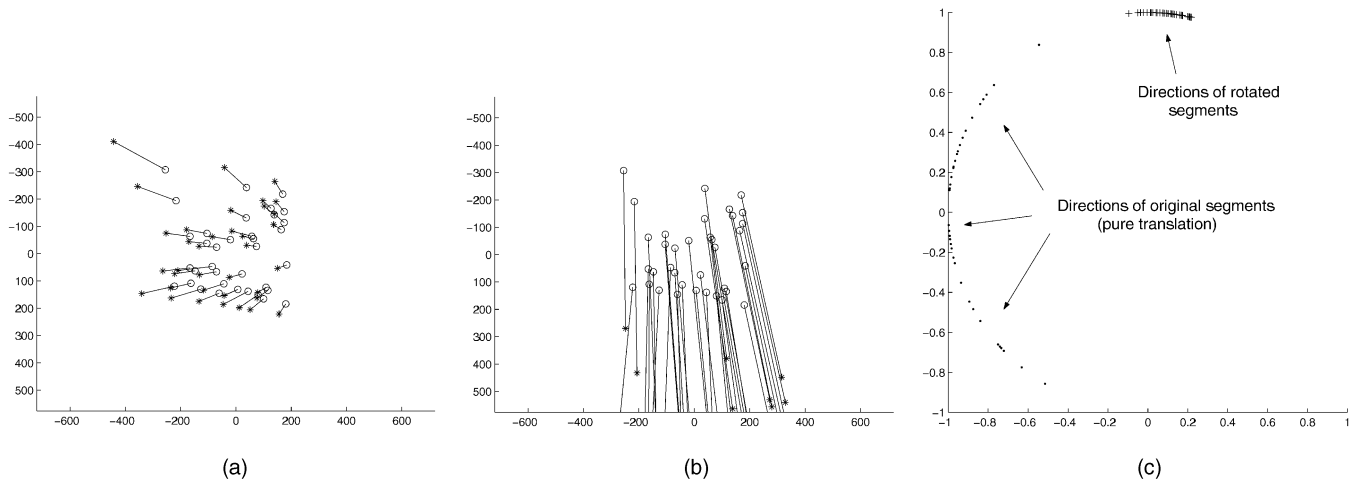


Fig. 4. An example showing that the "good" rotations are not de-rotations. (a) Segments from original (pure translation) pair of images. (b) Segments after applying one of the "good" rotations to image B. (c) Angular ranges of segments from the original and rotated pairs.

two terms g_1, g_2 (see Fig. 3). As we saw, the term g_2 is the dominant term and gives rise to a straight line passing through the origin. The direction of this line depends only on the rotation and is equal to $(k_x, k_y)^t$. The term g_1 gives rise to a set that is approximately a straight line, too, although this line is shorter. We saw that the direction of this line depends only on the translation and is $(-t_y, t_x)^t$. If the directions of these two lines happen to be equal, the set C will be virtually contained in a line passing through the origin and, hence, will have a nearly zero angular width. Note that the translation is fixed and we have no control over it. However, we may apply different rotations and, thus, have some control of the direction of the longer line resulting from the term g_2 (see Fig. 3).

In addition to specifying the optimal rotation axis, we should also specify the rotation angle around this axis. The choice of an optimal angle is affected by a large number of factors, including the translation vector and the depths of the points. However, it turns out that the range of suitable rotation angles is not very small and, hence, even random search for it will yield reasonable results.

3.2 Applying Different Rotations

We have seen that some unknown rotation should be applied to the second image in order to achieve the desired common behavior of the correct matches. This unknown rotation is found by applying K random rotations $\tilde{R}_k$ to the second image. Some of these rotations (by pure chance) will give a rotation with the desired characteristics. This is Step 2 in the algorithm. The main question is how to identify these "good" rotations. The approach we use is to identify these rotations by the characteristic which makes these rotations "good." We know that in the rotations which are suitable for our purpose, the inliers all point at the same direction. Thus, we expect a peak in the histogram of directions. The sharper the peak, the better we expect the rotation to be. In Step 2a of the algorithm, we find the peak of the histogram of directions. Step 2c then checks how sharp the peak is. In Step 3 of the algorithm, we choose those rotations in which the peak was the sharpest.

Having identified the proper rotations, we now compute for each segment $(p_i, \hat{p}_i)$ its average distance from the peak of the histogram. To be more precise, each segment $(p_i, \hat{p}_i)$ gives rise to a rotated segment $(p_i, \tilde{p}_i^k)$ for the kth rotation. We measure in Step 2b how far the direction of this segment is from the frequent direction m_k. Then in Step 4, we average these measurements, but we take into account only the measurements from the "good" rotations. For inliers, we expect all these measurements to be small, and for outliers, we expect these measurements to be more uniformly distributed with large values.

Based on the averages of these measurements, we want to decide which are the inliers by thresholding. The averages of inliers are all small and close to each other. On the other hand, the averages of outliers are larger and more widely dispersed. Thus,

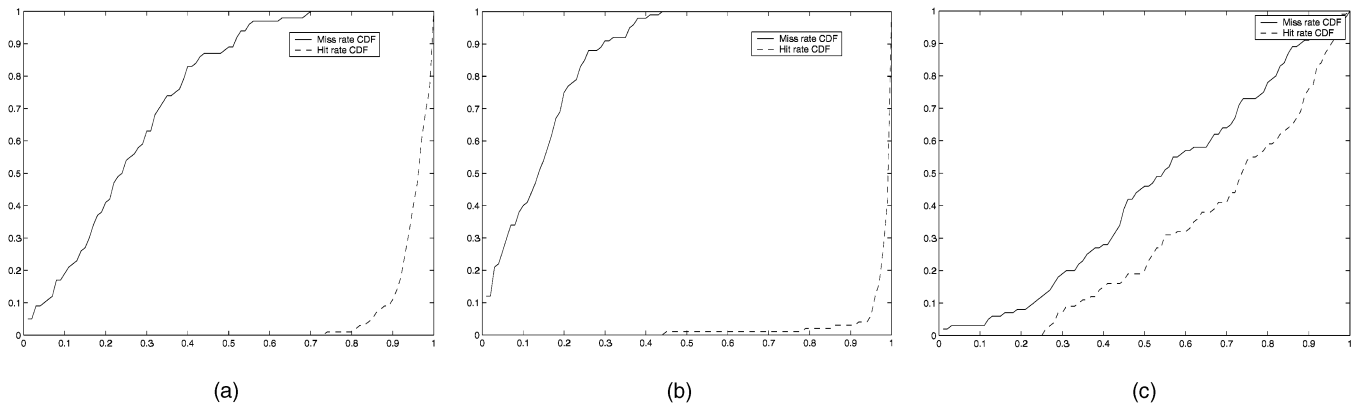


Fig. 5. Cumulative Distribution Functions (CDFs) of incorrect rejection (miss) and correct rejection (hit) probability distributions. (a) Basic scenario. (b) Scenario with large depths. Notice improvement of performance. (c) Scenario with close 3D points. The algorithm fails.

Fig. 6. Some of the real image pairs on which ROR was tested.

when we look at the distribution of the values these averages obtain, we see a peak corresponding to the inlier average value. We choose the threshold according to this peak value as done in Steps 5 and 6 of the algorithm.

3.3 Additional Remarks

At this point, we would like to make some additional remarks.

As is evident, our algorithm relies on common behavior of the inliers which creates peaks in certain distributions. In order for this to happen, we need some "critical mass" of inliers in the original population. Fortunately, it turns out that this "critical mass" does not have to be very high: even a low percentage of inliers such as 15-30 percent is sufficient, as will be shown in the next section.

Note that the complexity of our algorithm is determined by the complexity of finding the proper rotation. *This complexity is constant with respect to the percentage of outliers*. This is a fundamental difference with respect to other approaches such as RANSAC, where the complexity rises significantly with the percentage of outliers. In cases where the percentage of outliers is high, RANSAC may not be applicable because of its complexity, while our approach may still work.

Last, we would like to emphasize that the rotation we are searching for is not the unknown rotation between the two original views. In other words, we are not performing rotation compensation. Fig. 4 shows an example in which the two views are related by pure translation. In this case, no rotation compensation is required at all. Still, by applying the "good" rotation found by the algorithm, we obtain approximately parallel segments.

4 Results

The ROR algorithm was tested both on synthetic and on real images. We briefly describe the results obtained on synthetic images and then present results with real images. A more complete description of the simulation results may be found in [2].

Each synthetic image pair contained 30-45 inliers and 100-140 outliers, thus reaching approximately 70-80 percent contamination of outliers. For each scenario defined below, we generated 100 image pairs on which the algorithm was tested.

The basic scenario was the one in which the 3D points had x, y coordinates in the range $(-50, 50)$ and the depth was in the range $(150, 300)$ (the focal length is 1). This corresponds to a half field of view of approximately 18 degrees, which is reasonable. The rotation between the two screens was obtained by choosing randomly the roll, pitch, and yaw angles to be between 0 and 20 degrees. The z component of the translation was in the range $(15, 45)$ and was multiplied by -1 with probability 0.5. The x, y components of the translation vector were computed from the z component and the focus of expansion which was chosen randomly in the square $(-0.3, 0.3) \times (-0.3, 0.3)$.

We will consider two probability distributions for each scenario. The first is the probability that an inlier will be mistakenly classified as an outlier. The second is the probability that an outlier will indeed be identified and rejected. Fig. 5a shows the Cumulative Distribution Function(CDF) of these two distributions, as obtained from 100 pairs from the basic scenario. One may

TABLE 1
ROR Performance on Real Images

Image Pair	Total Number of Matches	Correct Matches	False Matches	Percentage of Inliers	Inlier Rejected (Misses)	Outliers Rejected (Hits)
1	378	77	301	20%	14 (**18%**)	260 (**86%**)
2	238	54	184	23%	6 (**11%**)	164 (**89%**)
3	130	34	96	26%	0 (**0%**)	86 (**90%**)
4	128	39	89	30%	3 (**8%**)	79 (**89%**)
5	129	30	99	23%	1 (**3%**)	91 (**92%**)
6	274	45	229	16%	11 (**24%**)	192 (**84%**)
7	215	23	192	11%	2 (**9%**)	110 (**57%**)
8	149	38	111	26%	10 (**26%**)	94 (**85%**)
9	168	24	144	14%	3 (**13%**)	103 (**72%**)
10	186	61	125	33%	1 (**2%**)	118 (**94%**)
11	191	38	153	20%	6 (**16%**)	127 (**83%**)

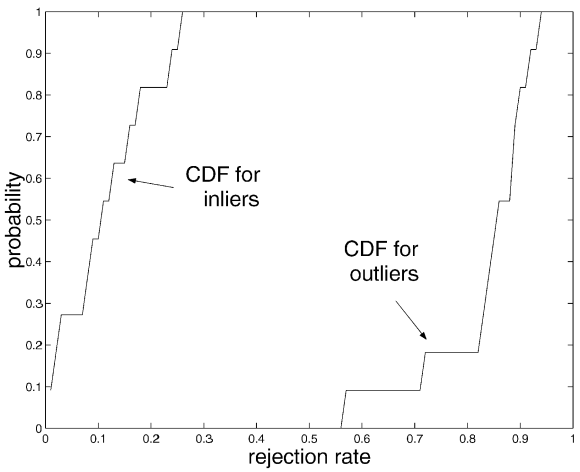


Fig. 7. CDFs for the rejection rate of outliers and inliers. These are empirical distributions based on the results obtained on 11 real image pairs.

see that the probability that no more than 30 percent of the inliers will be mistakenly rejected is around 70 percent. On the other hand, the probability that less than 90 percent of the outliers will be rejected is only around 15 percent. From the graphs, we see that in this scenario we have a very high probability that nearly all the outliers will be rejected and a good chance that 50 percent or more of the inliers will not be rejected.

In Section 3, we have discussed the need to assume that the 3D points have a dominant Z component. Figs. 5b and 5c show the results we obtained from two scenarios which differ in this respect. Fig. 5b shows the results for a scenario in which we changed the depths of the points to be in the range $(300, 450)$ instead of the range $(150, 300)$ as in the basic scenario. The results are clearly better than in the basic scenario. Fig. 5c shows that when the depth is in the range $(50, 100)$ then the algorithm does not work well.

In other scenarios, we have checked the effects of variability in depths, direction of translation, and magnitude of translation. Because of space limitations, we refer the reader to [2] for a description of these results.

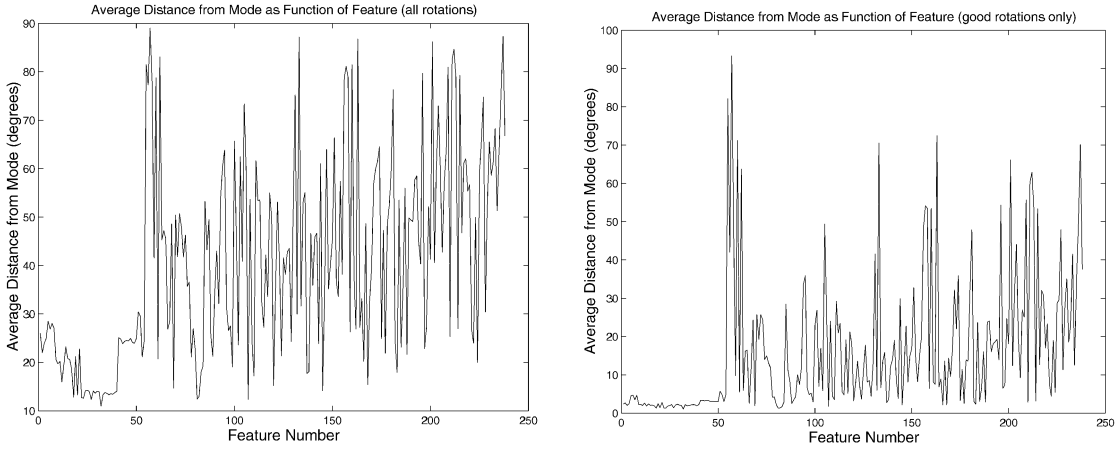
The results we have just described were obtained by running the algorithm with the following values for its parameters (see Section 2.2). The number of random rotations was $K = 1000$. From them, we have chosen the best $G = 50$ rotations. The width of the window in the MSM algorithm is $\pm 7.5\,\mathrm{deg}$. The threshold T was increased by $\alpha = 1\,\mathrm{deg}$. The percentage q which is used in Step 2c was 33 percent.

Real Images: Our algorithm was tested on several image pairs taken in our lab. At first, we ran a very simple matching algorithm. The algorithm extracts features using the SUSAN [11] algorithm, and then tries to match these features using a simple SSD cue. The results of this simple matching algorithm are not of high quality: only 15-30 percent of its matches are correct. Fig. 6 shows some of the image pairs on which the algorithm was tested.

Table 1 summarizes the results of running ROR on the real images and the matches found by the SSD matcher. Most of the outliers were rejected and the rejection rate of the inliers was almost always low. Notice that the outlier rejection rate was usually above 80 percent, while the inlier rejection rate was usually less than 20 percent. Fig. 7 summarizes these results by showing the empirical cumulative distribution functions (CDFs) of the rejection rates for outliers and for inliers. As may be seen, the probability distributions describing the rejection rates for outliers and for inliers, are clearly different. For example, we may see that the chance that more than 80 percent of the outliers will be rejected is roughly 80 percent, while the chance that less than 20 percent of the inliers will be rejected is over 80 percent.

We remark that the algorithm currently uses random search. In order to reduce the variability in results, we employed 10 runs (of 1,000 rotations each), and took the majority vote for each match. On a SUN Sparc this took 5 to 10 seconds (depending on the number of features).

As an example of the inner workings of the algorithm, we present data from one of the image pairs. We have ordered the 238 features such that the 54 correct matches will be at the beginning, and the false matches will follow. Fig. 8a shows the average distance of each segment direction (d_i^k in the earlier notation) from the mode direction, where the average is taken over all $K = 1,000$ rotations. It may be seen that there is a difference between the inlier segments and the outliers: compare the graph up to $x = 54$ and beyond. After we choose the $G = 50$ "good" rotations, this difference becomes much clearer. This may be seen in Fig. 8b which is a plot of the average distance $\bar{d}_i$, where the average is taken only on good rotations.



(a) (b)

Fig. 8. Average distance of each segment's direction from the mode. (a) Average over all rotations. (b) Average over "good" rotations only.

5 Conclusions

In this work, we have presented a novel method for rejecting false matches between two images. We have exploited the fact that we may perform (by software) a rotation of the camera and by doing that we have built many image pairs with matching points. We have shown that in some of these image pairs the correct matching points create line segments that point in approximately the same direction. By using the Mean Shift Mode algorithm, we were able to find those segments that do not point in the mode direction and classify them as false matches.

Our algorithm rejects mostly outliers and (unfortunately) a small number of correct matches. We have shown that under rather wide conditions, the algorithm succeeds and rejects a very significant portion of the outliers, while keeping most of the inliers. Since the algorithm does not manage to reject 100 percent of the outliers, robust estimation algorithms may still be needed for the application in mind.

We therefore suggest using the ROR algorithm as a preprocessor before employing further robust estimators:

- Use of the ROR may lower the percentage of outliers from an unacceptable percentage to a percentage below the breakpoint of the robust estimator to be used later on. Thus, by using ROR, we may clear the way for use of robust estimators in situations in which these estimators were previously useless.
- By reducing the percentage of outliers, ROR may significantly reduce the time required by robust algorithms, which are based on random sampling.

We have made the code for the algorithm available for research purposes [1].

Our current work addresses the following issues: The first is the rotation of the first image in addition to the second image. This idea is expected to further widen the scope of scenarios under which the algorithm is useful. We are also interested in understanding which are those outliers that are not rejected and what characterizes the inliers that do get rejected. Another interesting line of work is to use information gathered by our algorithm to gain some knowledge about the camera motion between the two original views. For example, should the mode direction approximate the perpendicular direction to the focus of expansion?

Acknowledgments

The authors would like to thank Moti Gornshtein for the C implementation of the algorithm. A provisional application for a patent on these algorithms has been filed in the United States.

References

[1] A. Adam, "ROR Implementation," available through http://www.cs.technion.ac.il/~amita/ 2000.
[2] A. Adam, "Vision-Based Navigation Packages," PhD thesis, Technion—Israel Inst. of Technology, 2000.
[3] Y. Cheng, "Mean Shift, Mode Seeking, and Clustering," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 17, no. 8, pp. 790-799, Aug. 1995.
[4] D. Comaniciu and P. Meer, "Distribution Free Decomposition of Multivariate Data," *Pattern Analysis and Applications,* vol. 2, pp. 22-30, 1999.
[5] J.J. Craig, *Introduction to Robotics,* second ed. Addison-Wesley, 1989.
[6] K. Fukunaga and L.D. Hostetler, "The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition," *IEEE Trans. Information Theory,* vol. 21, pp. 32-40, 1975.
[7] R. Haralick and L. Shapiro, *Computer and Robot Vision.* Addison-Wesley, 1992.
[8] R.I. Hartley, "In Defense of the 8 Point Algorithm," *Proc. IEEE Int'l Conf. Computer Vision (ICCV),* pp. 1064-1070, 1995.
[9] M.S. Lew and T.S. Huang, "Optimal Multiscale Matching," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR),* pp. 88-93, 1999.
[10] P. Pritchett and A. Zisserman, "Wide Baseline Stereo Matching," *Proc. IEEE Int'l Conf. Computer Vision (ICCV),* pp. 755-760, 1998.
[11] S. Smith and M. Brady, "Susan—A New Approach to Low Level Image Processing," *Int'l J. Computer Vision,* vol. 23, no. 1, pp. 45-78, 1997.
[12] P. Torr and A. Zisserman, "Robust Computation and Paramterization of Multiple View Relations," *Proc. IEEE Int'l Conf. Computer Vision (ICCV),* pp. 727-732, 1998.
[13] P.H.S. Torr and D.W. Murray, "The Development and Comparison of Robust Methods for Estimating the Fundamental Matrix," *Int'l J. Computer Vision,* vol. 24, no. 3, pp. 271-300, 1997.
[14] G.Q. Wei and G. Hirzinger, "Intensity and Feature-Based Stereo Matching by Disparity Parameterization," *Proc. IEEE Int'l Conf. Computer Vision (ICCV),* pp. 1035-1040, 1998.
[15] Z. Zhang, "Detrmining the Epipolar Geometry and Its Uncertainty: A Review," *Int'l J. Computer Vision,* vol. 27, no. 2, pp. 161-195, 1998.
[16] Z. Zhang, R. Deriche, O. Faugeras, and Q.T. Luong, "A Robust Technique for Matching Two Uncalibrated Images through the Recovery of the Unknown Epipolar Geometry," *Artificial Intelligence,* vol. 78, pp. 87-119, 1995.