ROR: Rejection of Outliers by Rotations in Stereo Matching

Amit Adam
Dept. of Mathematics
amita@tx.technion.ac.il

Ehud Rivlin
Dept. of Computer Science
ehudr@cs.technion.ac.il

Ilan Shimshoni
Dept. of Industrial Engineering
ilans@ie.technion.ac.il

Technion – Israel Institute of Technology
Haifa 32000 – Israel

Abstract

We address the problem of rejecting false matches of points between two perspective views. 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 behaviour of the correct matches. Those matches that deviate from this common behaviour turn out to be false matches. The statistical tool we use is the Mean Shift Mode Estimator.

Our algorithm does not use in any way the image characteristics of the matched features. In particular it avoids problems that cause the false matches in the first place. The algorithm may be run as a post processing step on output from any point matching algorithm. Use of the algorithm may significantly improve the ratio of correct matches to incorrect matches. On real images our algorithm has improved the percentage of correct matches from an initial $20\% - 30\%$ *to a final* $70\% - 80\%$*. 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.

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 [4]). The correspondence problem is sufficiently hard that even recent matching algorithms (for some examples see [8, 12, 7, 14]) 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 [5]) 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. Examples of robust algorithms may be found in [14, 13, 11, 10].

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 [8], choice of subsets of 8 matches in [14]). 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%-80%), and outputs a set of matches with a much higher percentage of correct ones. We do this by actively rotating one of the views, an action which "shakes" away the outliers. In contrast to other works, 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).

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. We do not use in any way the image characteristics themselves such as intensity, rank statistics in a window, texture etc. Therefore our algorithm avoids all the reasons which cause the false matches in the first place. Our algorithm may be employed on input which was obtained by any matching algorithm, and we do not use in any way the cues that were used to match. In particular, we may employ this algorithm even on images which were obtained from different imaging devices (see for example [6]).

We suggest using our algorithm as a post processing stage after any automatic matching algorithm. As will be demonstrated, we have used it on a very naive SSD matcher and have raised the percentage of correct matches from 20%-30% to around 70%-80%.

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. Let $P = (X,Y,Z)^t$ and $\hat{P} = (\hat{X},\hat{Y},\hat{Z})^t$ denote the coordinates of a certain point with respect to these two coordinate systems. Let $R, \vec{t}$ denote the rotation and translation of the camera between the two views. P and $\hat{P}$ are related by

$$\hat{P} = R(P - t) \tag{1}$$

We assume we are working with a calibrated camera with focal length f. The images of the point on the two screens are (x, y) and $(\hat{x}, \hat{y})$ with

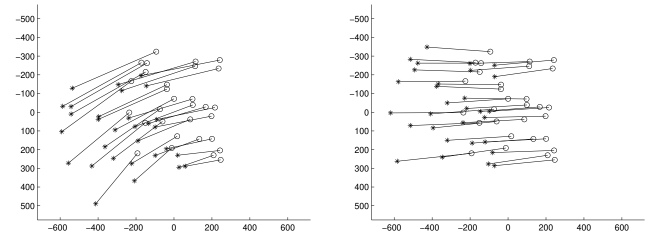
$$\begin{pmatrix} x \\ y \\ f \end{pmatrix} = \frac{f}{Z}\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}, \begin{pmatrix} \hat{x} \\ \hat{y} \\ f \end{pmatrix} = \frac{f}{\hat{Z}}\begin{pmatrix} \hat{X} \\ \hat{Y} \\ \hat{Z} \end{pmatrix} \tag{2}$$

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

$$S = \{l_i = (p_i, \hat{p}_i) = (x_i, y_i, \hat{x}_i, \hat{y}_i) | i = 1, \ldots, n\} \tag{3}$$

We will 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). The set S will be referred to as a scene of segments. Some of the matches in the scene are false - i.e. p_i and $\hat{p}_i$ are not the images of the same point in space. The segments which are false matches will also be called outliers, and the segments which are correct matches will be called inliers. Our aim is to identify the outlier segments or false matches, and reject them.

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. In other words, it is possible to compute the image that would have been obtained had the camera been rotated, with no change in its position. (This is possible because all points on a fixed ray originating from the center of projection, would be imaged in the rotated view at the same point: the depth of points is irrelevant to the displacement of their image under pure rotation).



(a) (b)

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

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. 1(a) and 1(b). 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. 1(a) depicts point matches between 2 images A and B which are related by a certain translation and rotation of the camera. Fig. 1(b) 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 in 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 in the same direction. Incorrect matches between the two images will lead to segments which do not point in that direction. Thus we may identify the incorrect matches by identifying deviation of the segment direction from the most frequent direction.**

This idea cannot be used immediately. The reason is that we cannot know in advance how to rotate image B in order to obtain line segments that are all pointing in the same direction. We therefore search the rotation space for those "good" rotations, and after they are identified, we use them to find the incorrect matches. The identification of the "good" rotations is based on the same quality which makes them "good" - namely having a sharp peak in the density of segment direction distribution.

We now present the ROR (rejection of outliers by rotations) algorithm. At this stage the reader may just obtain a general impression of the algorithm. We will explain the different steps of the algorithm in the following section.

ROR Algorithm

1. Start with correspondence segments $s_i = (p_i, \hat{p}_i)$ between image A and image B.
2. Perform K random rotations of image B. For the k'th 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 in, 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.

Remark: The mode computations are done by using the Mean Shift Mode Estimator. Lack of space prevents us from describing this algorithm. The reader is referred to [1, 3].

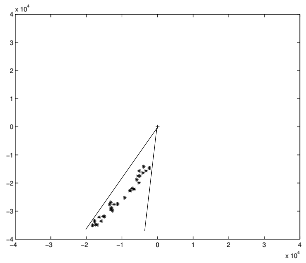


Figure 2: The angular width of the set C as viewed from the origin

3 Geometric Interpretation of the ROR Algorithm

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 in, to some characteristic of 3D feature vectors. Then we reason by using the 3D vectors.

3.1 Working with 3D Feature Vectors

Let $s_i = \hat{p}_i - p_i$ denote the i'th segment. By direct computation

$$s_i = \begin{pmatrix} f\frac{\hat{X}}{\hat{Z}} - f\frac{X}{Z} \\ f\frac{\hat{Y}}{\hat{Z}} - f\frac{Y}{Z} \end{pmatrix} = \frac{f}{Z\hat{Z}} \begin{pmatrix} Z\hat{X} - X\hat{Z} \\ Z\hat{Y} - Y\hat{Z} \end{pmatrix} \quad (4)$$

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

$$\vec{\mu}_i = P_i \times \hat{P}_i = (-(Z\hat{Y} - Y\hat{Z}), Z\hat{X} - X\hat{Z}, X\hat{Y} - Y\hat{X})^t \quad (5)$$

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 $\vec{\mu}_j$.

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 [2], we have

$$R\vec{v} = \cos\theta\vec{v} + \sin\theta(\hat{k} \times \vec{v}) + (1 - \cos\theta)(\hat{k} \cdot \vec{v})\hat{k} \quad (6)$$

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

$$\begin{aligned}\vec{\mu} &= P \times \hat{P} = P \times R(P-t) \\ &= -\cos\theta(P \times t) + \sin\theta(P \cdot (P-t))\hat{k} - \\ &\sin\theta(P \cdot \hat{k})(P-t) + (1-\cos\theta)(\hat{k} \cdot (P-t))(P \times \hat{k})\end{aligned} \quad (7)$$

Let us denote the four terms in (7) by

$$\begin{aligned} g_1 &= -\cos\theta(P \times t) \\ g_2 &= \sin\theta(P \cdot (P-t))\hat{k} \\ g_3 &= -\sin\theta(P \cdot \hat{k})(P-t) \\ g_4 &= (1-\cos\theta)(\hat{k} \cdot (P-t))(P \times \hat{k}) \end{aligned} \quad (8)$$

Each vector $\vec{\mu}_i$ is the sum of these 4 vectors. The x, y projection of $\vec{\mu}_i$ is the i'th point in the set C. This point is the sum of the x, y projections of each of the four terms.

3.2 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 behaviour of the inliers allows us to differentiate them from the outliers.

The set C is the xy projection of the 3D points μ_i as in (7). 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. Figure 3 shows the xy projections of the 4 terms $g_1, \ldots, g_4$ defined in (8), 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:

$$g_1 = -\cos\theta(P \times t) = -\cos\theta \begin{pmatrix} t_z Y - t_y Z \\ -t_z X + t_x Z \\ t_y X - t_x Y \end{pmatrix} \quad (9)$$

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

$$(g_1)_{1,2} \approx -\cos\theta Z \begin{pmatrix} -t_y \\ t_x \end{pmatrix} \quad (10)$$

Indeed we may see in the figure that the term g_1 gave rise to a second, shorter line (marked by circles) which

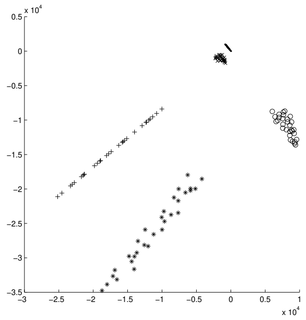


Figure 3: The four components of μ_i and their sum. $\circ$ - projection of g_1 term. $+$ - projection of g_2 term. $\times$ - projection of g_3 term. $\bullet$ - projection of g_4 term. $*$ - point in C (sum of the 4 terms)

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 x's 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_i's 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

$$\begin{aligned} g_4 &= (1-\cos\theta)(\hat{k} \cdot (P-t))(P \times \hat{k}) \\ &= (1-\cos\theta)(\hat{k} \cdot (P-t)) \begin{pmatrix} k_z Y - k_y Z \\ -k_z X + k_x Z \\ k_y X - k_x Y \end{pmatrix} \end{aligned}$$

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

$$(g_4)_{1,2} \approx \mathcal{O}(XZ + YZ) \begin{pmatrix} -k_y \\ k_x \end{pmatrix} \quad (11)$$

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 two terms g_1, g_2 (see figure 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).

To summarize, we saw that under some rotations the set C will occupy a small angular region when viewed from the origin, and this implies that all the inlier segments will point in the same direction. The rotations for which this happens are those in which the rotation axis is has a small z component and the first two components approximately satisfy

$$\begin{pmatrix} k_x \\ k_y \end{pmatrix} \parallel \begin{pmatrix} -t_y \\ t_x \end{pmatrix}$$

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. For reasons of space we cannot discuss these factors here. We just remark that the exact rotation angle is not very important and there is a rather wide (tens of degrees) range of appropriate rotation angles which will do.

3.3 Applying Different Rotations

As we just saw, we would like to apply a rotation to the second image such that the resulting axis of rotation will have some specific direction. However, we cannot specify which rotation to apply, for two reasons. The first reason is that the desired axis of rotation should be parallel to $(-t_y, t_x)^t$, but we have no knowledge of the translation vector. The second reason is the following. We apply a rotation $\tilde{R}$ to the second image B which is already rotated by R w.r.t. the first image A. Thus the overall rotation matrix is $\tilde{R}R$ and it is the axis of this rotation matrix which we want to be perpendicular to the translation vector (i.e. parallel to $(-t_y, t_x)^t$). However, even though we control $\tilde{R}$, the overall matrix is unknown since R is unknown.

We overcome this problem 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 in 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 k'th 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 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 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.

As is evident, our algorithm relies on common behaviour 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 $20\% - 30\%$ is sufficient as will be shown in the next section.

4 Results

The ROR algorithm was tested both on synthetic and on real images. Each synthetic image pair was created as follows. Between 30 to 45 points in 3D space were projected on two screens thus creating $30 - 45$ inlier segments. To these we added $100 - 140$ false matches, thus reaching approximately $70\% - 80\%$ contamination of outliers. For each scenario we wanted to test, we created 100 synthetic pairs. We will present the distribution of results from these 100 pairs.

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

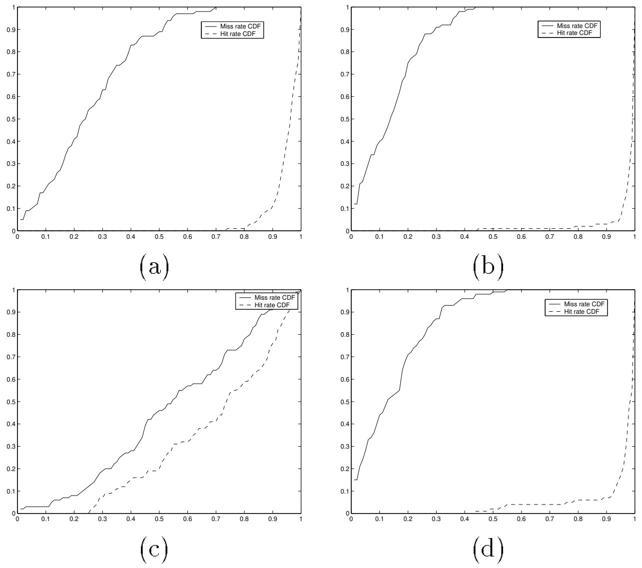


Figure 4: Cumulative Distribution Functions (CDF's) of incorrect rejection (miss) and correct rejection (hit) probability distributions. (a) Basic scenario. (b) Scenario with large depths. Notice improvement of performance. (c) Scenario with close 3D points. The algorithm fails. (d) Near constant depth scenario.

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. Figure 4(a) shows the Cumulative Distribution Function(CDF) of these two distributions, as obtained from 100 pairs from the basic scenario. One may see that the probability that no more than 30% of the inliers will be mistakenly rejected is around 70%. On the other hand, the probability that less than 90% of the outliers will be rejected is only around 15%. 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% 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. Figures 4(b) and 4(c) show the results we obtained from two scenarios which differ in this respect. Fig. 4(b) 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. 4(c) shows that when the depth is in the range $(50, 100)$ then the algorithm does not work well.

Image Pair	Total Number of Matches	Correct Matches	False Matches	Percentage of Inliers
1	238	54	184	23%
2	130	34	96	26%
3	128	39	89	30%
4	129	30	99	23%
5	378	77	301	20%

Table 1: Inlier/outlier statistics of the input to ROR

In the next scenario we checked whether variability in depth is crucial for the algorithm to perform. Fig. 4(d) shows the results for the scenario where the depth of the points was in the range $(300, 320)$ - hence low variability in depth. As may be seen, if the points are not too close to the camera, variability is not a necessary condition for the algorithm to succeed.

In other scenarios we have checked how the algorithm performs under small translations and under different directions of translation. The algorithm performed well in these scenarios. For reasons of space we omit these results.

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 [9] 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 20%−30% of its matches are correct. The left two columns in Fig. 5 show the image pairs with the corresponding pairs the SSD algorithm found.

The correspondences which were determined by the initial matching algorithm were checked manually. The number of true and false correspondences is summarized in Table 1. It may be seen that the percentage of inliers is rather low.

The ROR algorithm was run on these inputs from the initial SSD matcher. Most of the outliers were rejected and the rejection rate of the inliers was low except for one case. The results are summarized in Table 2. The two right columns of Fig. 5 show the output of the ROR algorithm on the image pairs.

Notice that the outlier rejection rate was around 90% in all cases. The inlier rejection rate was low (under 10%) for all cases except the last image pair. Notice that in this image pair we have many features which are rather close to the camera. As we have reasoned in section 3 and as we saw in the results from the synthetic scenes, the algorithm does not perform well in this case. However, even in this case the inlier percent-

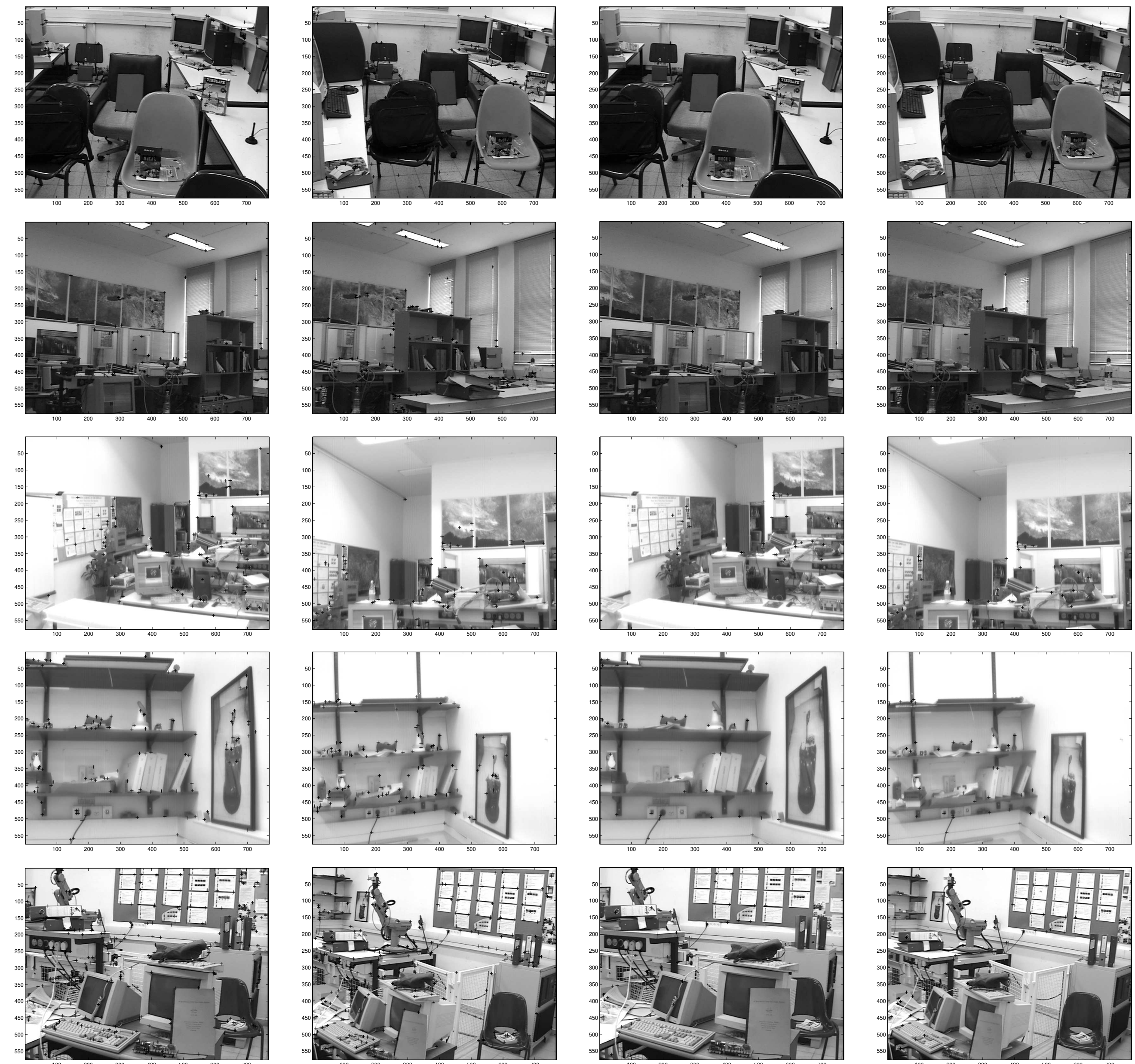
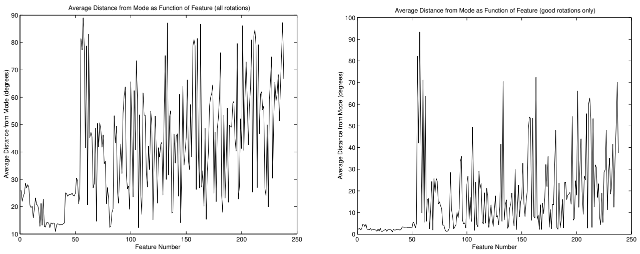


Figure 5: Real image pairs. The two left columns are the image pairs with the input correspondences to the ROR algorithm. The two right columns are the corresponding pairs which were selected by the ROR algorithm

Image Pair	Inliers Rejected (Misses)	Outliers Rejected (Hits)	New Inlier Percentage
1	6 (11%)	159 (86%)	66%
2	0 (0%)	86 (90%)	77%
3	4 (10%)	80 (90%)	80%
4	1 (3%)	91 (92%)	78%
5	37 (48%)	270 (90%)	56%

Table 2: ROR performance statistics (real images)



(a) (b)

Figure 6: Average distance of each segment's direction from the mode. (a) Average over all rotations. (b) Average over "good" rotations only.

age has risen from around 20% to around 55% which is still impressive.

As an example of the inner workings of the algorithm, we present data from the first image pair. We have ordered the 238 features such that the 54 correct matches will be at the beginning, and the false matches will follow. Fig. 6(a) 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 $K = 1000$ 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. 6(b) which is a plot of the average distance $\bar{d}_i$, where the average is taken only on good rotations.

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 rotate an image 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 improves significantly the ratio of correct matches to false matches. This feature may reduce significantly the computational cost of robust estimation algorithms which are usually employed on the set of matching features.

Our algorithm does not rely in any way on the image characteristics of the corresponding pairs. In particular it avoids the problems which cause false correspondences in the first place. The algorithm may be employed on output from any matching algorithm. It may also be employed on matches between images obtained from different sensors (IR image and visible wavelengths image for instance).

A number of improvements may be made in our algorithm. These include the automatic setting of window sizes used in the Mean Shift algorithm. In addition we would like to include automatic identification of failure of the algorithm in those cases where it is expected to fail (close scene, wide field of view). Another interesting line of work is to use information gathered by our algorithm to gain some knowledge about the camera motion between the two views. For example, the mode direction should approximate the perpendicular direction to the focus of expansion.

References

[1] Y. Cheng. Mean shift, mode seeking and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, August 1995.

[2] John J. Craig. *Introduction to Robotics*. Addison-Wesley, second edition, 1989.

[3] K. Fukunaga and L. D. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Trans. Information Theory*, 21:32–40, 1975.

[4] R. Haralick and L. Shapiro. *Computer and Robot Vision*. Addison-Wesley, 1992.

[5] R. I. Hartley. In defence of the 8 point algorithm. In *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, pages 1064–1070, 1995.

[6] M. Irani and P. Anandan. Robust multi-sensor image alignment. In *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, 1998.

[7] M. S. Lew and T. S. Huang. Optimal multi-scale matching. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 88–93, 1999.

[8] P. Pritchett and A. Zisserman. Wide baseline stereo matching. In *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, pages 755–760, 1998.

[9] S. Smith and M. Brady. Susan - a new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–78, 1997.

[10] P. Torr and A. Zisserman. Robust computation and paramterization of multiple view relations. In *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, pages 727–732, 1998.

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

[12] G. Q. Wei and G. Hirzinger. Intensity and feature based stereo matching by disparity parameterization. In *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, pages 1035–1040, 1998.

[13] Z. Zhang. Detrmining the epipolar geometry and its uncertainty: A review. *International Journal of Computer Vision*, 27(2):161–195, 1998.

[14] 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*, 78:87–119, 1995.