

Structure from Stereo Vision using Unsynchronized Cameras for Simultaneous Localization and Mapping

Marcus Svedman

Royal Institute of Technology
Stockholm, Sweden

Email: marcus.svedman@gmail.com

Luis Goncalves, Niklas Karlsson, Mario Munich, Paolo Pirjanian

Evolution Robotics, Inc.
Pasadena, California, USA

Email: {luis,niklas,mario,paolo}@evolution.com

Abstract—This paper presents a system for automatic reconstruction of 3D structure using two unsynchronized cameras. Three images are acquired sequentially from the left, right, and again from the left camera. A virtual image from the left camera synchronized with the right image is created by interpolating matching points of interest (SIFT features) in the two left images. Both geometric and probabilistic criteria are used to select the correct set of matching features amongst the three views. In an indoor environment, the method typically results in 3D structure with approximately 200 feature points, with a median 3D accuracy of 1.6 cm when the average depth is 3 m and the robot has moved 1-2 cm between each image acquisition.

Index Terms—Stereo Vision, 3D Structure, Outliers Detection, Unsynchronized Cameras, SLAM, Feature Correspondence.

I. INTRODUCTION

The degree of autonomy of a robot is very dependent on its ability to perform simultaneous localization and mapping (SLAM). An autonomous robot should be able to explore its environment without user intervention, build a reliable map, and be able to localize itself within that map. Many approaches to the SLAM problem can be found in the literature, but the most promising one in terms of cost, computation, and accuracy is the so-called *visual simultaneous localization and mapping* (vSLAMTM) [8], [9]. In vSLAM, localization is achieved based on recognizing a visual landmark that is nothing but a set of unique and identifiable feature points in 3D space. If a subset of these features are later observed, the vSLAM system can calculate the new robot's new pose [5].

Given the pose of the two cameras in the stereo pair and the correspondence between features in the left and right image, then the 3D coordinates of the feature point in space can be computed using triangulation. In a stereo vision system, the cameras' relative positions are well known from prior calibrations [1]. If the robot is moving, it is normally necessary to use expensive synchronized cameras so the relative pose difference between the right and left images remain the same for all the frames. For many consumer robotics applications however, only inexpensive, unsynchronized cameras like webcams can be used. If the cameras are unsynchronized and the robot moves, the relative pose between the robot and the environment changes slightly

between the left and right images, resulting in large errors in the final estimated 3D structure. This paper presents a solution for correcting this change of pose in order to obtain an accurate 3D structure. It also describes a fully functional stereo vision system that typically finds corresponding points in the two views with less than 1% outliers. The webcams used in our system had additional wide-angle lenses attached that presented a noticeable degree of distortion that was removed using a well-known calibration toolbox [1], [2], [7].

Different approaches have been presented to synchronize unsynchronized multi-view video streams, where the main emphasis so far has been put on finding the time difference between the different video streams, using for example a voting scheme for cost functions [13]. If objects move in the video stream, the fact that they move in the same way in all views can be used to calculate the time difference between views [14], [18]. If a whole sequence of a dynamic scene is observed, both pixel correspondences and synchronization can be post-processed and calculated [3], [4].

These approaches use a variety of visual cues and linear interpolation to find the time difference and synchronize the images within video streams. The algorithm in this paper, on the other hand, assumes that the time difference already is known and concentrates on the problem of finding a correct set of matching features, and creating accurate 3D structure with few outliers. If the time difference cannot be measured directly, the method proposed by Zhou and Tao in [19] using four feature points correspondences on a line would give a good estimation.

Our algorithm finds feature point correspondences between the two views using the *Scale Invariant Feature Transform* (SIFT) proposed by David Lowe [10], which is the most robust feature descriptor available [11]. The SIFT algorithm makes it possible to extract a large number of unique feature points from an image, and provides a way of finding probable feature correspondences in other images. Our system uses the feature matching algorithm proposed by Lowe and applies a number of constraints in order to find the most likely feature matches. A final outliers rejection step is used to validate the extracted correspondences between the features. Our algorithm assumes that odometry measurements are available,

and that accurate timestamps of each image are recorded.

This paper is organized as follows. Section II describes the stereo matching procedure, including the feature selection, correspondence calculation, and the interpolation process used to synchronize the images. Section III presents different algorithms to find and remove outliers from the set of proposed feature matches. Section IV evaluates the performance of interpolation and the quality of the triangulated 3D structure. Sections V and VI drafts some conclusions and discusses further improvements to the system.

II. STEREO MATCHING

A. Overview

In this paper, a 3D structure is created using three images. Two consecutive images are acquired with the left camera (L1, L3). One image is acquired with the right camera (R2) somewhere in the time lap between the two left images (see Figure 1). In order to create accurate 3D structure, features from an image synchronized with image R2 are needed. This image (L2) is not available, but L1 and L3 can be used to interpolate the missing information. This is the key idea in the proposed stereo matching algorithm, which can be summarized in the following steps:

- 1) Extract features in all three images and save the features in L1 and L3 in searchable databases.
- 2) For every feature f_i found in R2, do:
- 3) Look in image L1 for features similar to f_i . For every found feature $f_{i,j}^1$ that fulfills certain constraints (if any exist), do:
- 4) Look in image L3 for features similar to $f_{i,j}^1$. For every found feature $f_{i,j,k}^3$ that fulfills certain constraints (if any exist), do:
- 5) Calculate the interpolated coordinates $f_{i,j,k}^2$ between $f_{i,j}^1$ and $f_{i,j,k}^3$ that matches the timestamp of R2.
- 6) Repeat the steps 3-5 to find additional feature matches for f_i , using the reversed search direction $R2 \rightarrow L3 \rightarrow L1$.
- 7) Calculate the quality of the match for the features $f_{i,*}^2$ that have been kept.
- 8) Save the best feature match along with the quality of the match.
- 9) If a previously processed feature match $f_n, n < i$ in R2, matched with the same interpolated feature above, keep the match with the best match quality.

The algorithm assumes that the coordinates of the corresponding features in L1 and L3 change linearly between these two frames. This assumption is a good approximation if the time between the two images is small and the robot moves smoothly.

B. Finding Synchronized Feature Matches

If the rotation and translation (difference in pose) between two views is known, like for example image R2 and L2, a

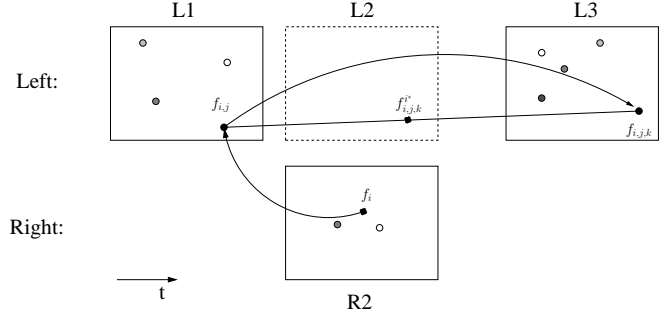


Fig. 1. Schematic description of the algorithm. If a match is found between the three images (L1, L3, and R2), the feature coordinates in image L1 and L3 are interpolated to match the timestamp of R2.

feature in image R2 has its corresponding feature, according to epipolar geometry, somewhere along its corresponding epipolar line in L2 [6].

If the epipolar lines can be calculated, the matching features are easier to find. It is easy to calculate the epipolar line in L2 for every feature in R2 since the pose difference is known from prior calibration. The pose difference between R2 and L1, and between R2 and L3, are not as obvious when the robot is moving. In our system, wheel odometers are available that makes it possible to estimate the motion of the robot between each image. The epipolar lines, however uncertain, can thus be calculated in L1 and L3. The epipolar lines may not be very accurate, especially if the odometers have a slow sample rate, but still gives a rough estimation of where the epipolar lines lie.

The SIFT feature extraction algorithm provides the location, scale, and orientation as well as a 128 dimensional feature descriptor for each feature [10]. Features can be compared by computing the SIFT distance, which is the Euclidian distance between the 128 dimensional feature descriptors. Similar features have similar feature descriptors, which corresponds to a small SIFT distance between them.

SIFT features are first extracted from the three images and the features in L1 and L3 are stored in two searchable databases. For every feature f_i in image R2, we initially look for the most similar features $f_{i,j}^1$ in image L1 by searching for features with small SIFT distances to f_i .

The epipolar line in image L1 is calculated and all the features $f_{i,j}^1$ that lie reasonably close to the epipolar line could be the correct match. The distance to the epipolar line is calculated for each feature, and if the distance is smaller than a certain threshold, the feature is kept. The choice of threshold depends on the uncertainty of the epipolar line in L1. It is preferable to select a large threshold in order to assure that the right match always is accepted although a higher threshold also allows more false matches to be kept.

If none of the possible features $f_{i,j}^1$ were close enough to the epipolar line, then f_i is assumed to not have any matches. All accepted features $f_{i,j}^1$ found in image L1 are used to search for a match in image L3. For every accepted

feature $f_{i,j}^1$ in image L1, the most similar features $f_{i,j,k}^3$ in image L3 are extracted. Just as in image L1, the epipolar line in L3 is calculated, and if any of the matches found in image L3 are close to the epipolar line they are used for interpolation. If none of the found features in L3 lie close to the epipolar line, then no match is found for f_i and the next feature f_{i+1} in the right image R2 is processed. It is assumed that the exact timestamp of each image is known. If the robot has been moving smoothly, and the time difference between image L1 and L3 is small, then the changes in feature coordinates between the two images can be approximated with a linear function. The algorithm estimates the position of the feature at the timestamp of image R2 by linearly interpolating the coordinates of $f_{i,j}^1$ and $f_{i,j,k}^3$. All possible matches that fulfilled the epipolar constraints are now evaluated using a set of different constraints to decide which one, if any, is the correct match.

The algorithm presented in this section finds matches from image R2 to image L1, and then from image L1 to image L3. In reality, it is more robust to do it both directions; i.e., in the order $R2 \rightarrow L1 \rightarrow L3$ as well as $R2 \rightarrow L3 \rightarrow L1$. Sometimes the same feature correspondence is found by both matching directions, making one of the computations completely unnecessary. However, it is our experience that some of the correct feature correspondences are obtained by only one of the two matching procedures.

C. Computing Quality of Match

All the candidate feature matches that fulfill the epipolar constraints are evaluated using a second set of constraints based on the following eight parameters:

- Distance to epipolar line from $f_{i,j,k}^2$.
- SIFT distance between f_i & $f_{i,j}^1$, and between $f_{i,j}^1$ & $f_{i,j,k}^3$.
- Difference in SIFT scale between f_i & $f_{i,j}^1$, and between $f_{i,j}^1$ & $f_{i,j,k}^3$.
- Difference in SIFT orientation between f_i & $f_{i,j}^1$, and between $f_{i,j}^1$ & $f_{i,j,k}^3$.
- Coordinate disparity between $f_{i,j}^1$ & $f_{i,j,k}^3$.

The SIFT scale corresponds to the size of the SIFT feature [10]. Since the scene in front of the robot is at almost the same distance in the three images, all corresponding feature pairs should have similar scale in the different images. The SIFT feature extraction algorithm also extracts the prominent gradient direction of each feature, known as the SIFT orientation, so that the same feature can be recognized even if it has been rotated. The difference in SIFT orientation between matching features is expected to be small between the two left images. The difference in SIFT orientation between left and right view should be larger but is typically small too, and is also used as a matching parameter.

If the robot primarily moves forward, as it does in our case, and the robot has not moved far between the left images

(high image frame rate or slow motion), the coordinates of the matching features in L1 and L3 should be similar. If the robot has rotated, the difference between matching feature coordinates is much larger and should not be used as a matching parameter.

The above parameters are calculated and thresholded for each possible match pair to reject unlikely matches. The threshold values have been obtained using ground truth data. A set of 1500 candidate feature matches from different environments were manually classified into the three categories *correct*, *false* or *uncertain*. The thresholds for the different parameters are found numerically by maximizing

$$D = \frac{P(\text{correct positive}) - P(\text{false positive})}{\frac{\text{No. correct feat. passing all thresholds}}{\text{Tot. no. correct features}} - \frac{\text{No. false feat. passing all thresholds}}{\text{Tot. no. false features}}} \quad (1)$$

The maximization is performed using the *Nelder-Mead* simplex method [12]. Since this function has many degrees of freedom, it has many local maxima and is difficult to maximize. Instead of trying over and over using random initial guesses hoping to achieve the global maxima, the optimization follows an incremental approach in which the best solution retrieved up to date is refined in each step. The best thresholds obtained in the previous tries are used as initial guesses in the new search, but with a small amount of noise added to the values. This way, the thresholds are perturbed enough to make the initial guess jump out of the local maxima and hopefully land in an even better local maxima. Once the optimization process converges, the whole process is repeated a few times from different initial guesses to increase the probability of actually finding the global maxima. The different calculated thresholds are accepting between 95 and 99 percent of all the correct matches while rejecting between 50 and 95 percent of the false matches. Even though each threshold only is rejecting a relatively small number of all the outliers, and many thresholds reject the same features, most outliers are caught by at least one of the different thresholds.

If several candidate matches pass the thresholding phase, the best match has to be picked. The probability of being a correct match given eight match parameters should be calculated, so that the match with the greatest probability can be chosen. In this application, there is a need to quantify which feature match is the most probable out of the possible ones. The different parameters have experimentally been shown to be uncorrelated by plotting them as function of each other and manually checking for correlations. A good numerical approximation of the probability of a match given the parameters, assuming that the parameters $d_i \dots d_8$ are independent, is given by

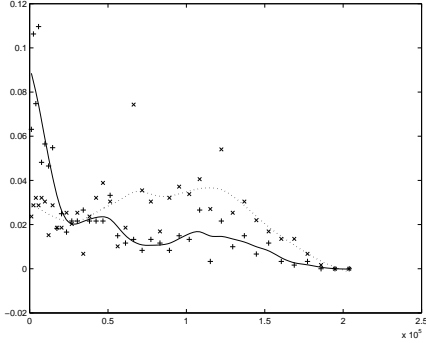


Fig. 2. Estimation of $P(d_i|\text{match})$ and $P(d_i|\text{non-match})$. The plot shows the normalized histograms for the SIFT distance parameter between $f_{i,j}^1$ and $f_{i,j,k}^3$. The crosses and pluses correspond to the histogram values of $P(d_i|\text{non-match})$ and $P(d_i|\text{match})$ respectively. The dotted and solid lines are their smoothed spline functions.

$$\begin{aligned}
 P(\text{match}|d_1\dots d_8) &= \frac{P(d_1\dots d_8|\text{match})P(\text{match})}{P(d_1\dots d_8)} \\
 &= \prod_{i=1}^8 \left(\frac{P(d_i|\text{match})}{P(d_i)} \right) P(\text{match}) \\
 &= \frac{\prod_{i=1}^8 P(\text{match}|d_i)}{P(\text{match})^7} \quad (2)
 \end{aligned}$$

Since we are only interested in the best match relative to all candidates, the scaling factor $P(\text{match})$ can be neglected. The manually extracted dataset shows the distribution of the different parameters for the matches and non-matches for the different thresholds. By calculating normalized histograms of the different individual parameters, an estimate of their individual *probability mass function* is obtained. The different parameter distributions have different shapes and cannot be described particularly well with any of the commonly used probability distributions. In our case, we align smooth curves to the noisy histograms using cubic smoothing splines [17]. This way, reasonable approximations to $P(d_i|\text{match})$ and $P(d_i|\text{non-match})$ are achieved as shown in Figure 2. The desired probabilities $P(\text{match}|d_i)$ are given by

$$P(\text{match}|d_i) = \frac{P(d_i|\text{match})}{P(d_i|\text{match}) + P(d_i|\text{non-match})} \quad (3)$$

using Bayes Rule, and assuming that $P(\text{match}) = P(\text{non-match}) = 0.5$ since the histograms are normalized.

The estimate of $P(\text{match}|d_i)$ is estimated by evaluating the values at the two normalized spline curves for the particular d_i , and then calculating the ratio between curves according to Equation 3. The final probability of match, given by Equation 2, is calculated for all candidate matches that passed all the initial thresholds.

III. OUTLIERS REJECTION

A. Removing Uncommon Coordinate Disparities

The coordinate disparity is the difference between the x-, and y-coordinates in two images. It should be small in the

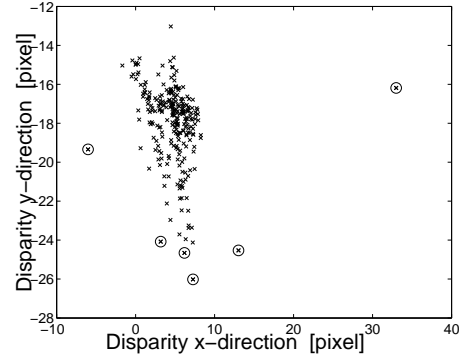


Fig. 3. Coordinate disparity between features in images L2 and R2. Good features tend to cluster in disparity space. The encircled crosses are features rejected by the algorithm.

two left images. However, the disparity between the left and right images might not be as small. Not much can be said about the disparity between individual features in generic terms aside from the fact that the disparity of all features change in similar ways depending on the distance to the objects. In most cases, depending on how the cameras have been mounted and on the structure of the environment, the disparity is typically distributed in a small densely populated area. If a few pairs of features have a disparity different from the majority, then they are likely to be outliers, and should thus be removed. Figure 3 shows the distribution of disparity values between features in images L2 and R2 (the encircled crosses are outliers rejected by the algorithm).

Outliers are found using *Singular Value Decomposition* (SVD) of the disparity difference vectors [15]. The *principal component directions*, or the directions of the largest variations in the data are retrieved, and the standard deviations, σ , in those directions are calculated. Feature pairs with disparities that differ more than 4σ from the dataset's median disparity are likely outliers and removed.

B. Removing Impossible 3D Coordinates

After the 3D structure has been calculated, the first obvious test is to remove impossible features like all features located behind the robot or closer to the robot than physically possible. Features that are far away are likely to be outliers or have a high uncertainty anyway, so features further away than a certain threshold from the robot are also removed.

C. 3D Structure Estimation and Re-projection onto the Image

When the triangulation was calculated, the algorithm computed the optical rays passing through the feature points in L2 & R2 and finds a 3D point where the rays intersect. Since the optical rays do not really intersect because of small errors, the algorithm instead calculates where the rays are closest to intersecting and estimates the triangulated 3D point as the point half way between the rays.

Once the triangulated 3D structure is obtained, the algorithm re-projects the 3D structure back onto image R2. It is then easy to compare the ground truth feature coordinates

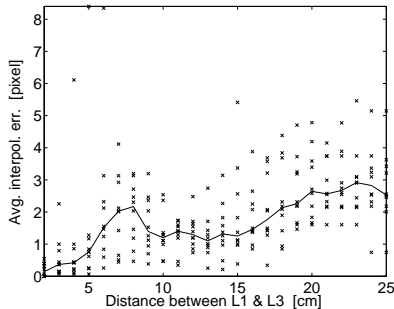


Fig. 4. Interpolation error. Image L2 is acquired and the true coordinates in L2 are compared with the interpolated coordinates using L1 and L3. The figure shows the error between interpolated coordinates and the true coordinates for ten datasets at different distances between L1 and L3. The solid line is the averaged value.

and the re-projected feature coordinates. Features with a re-projection error greater than 4σ are assumed to be outliers and removed.

D. Removing Uncommon 3D Coordinates

Uncommon coordinate disparities were assumed outliers. The same idea is also used on the actual 3D coordinates to remove outliers. The idea is that the 3D features usually are limited to certain volume in 3D space. In an indoor environment, all features are within the walls of the room, or in a 3D tube if a corridor is observed. The principal components are calculated using SVD, just as in Section III-A, and features that differ more than 4σ away from the median in the principal component directions are removed from the dataset.

IV. EXPERIMENTS

A. Interpolation Error in L2

The triangulation is very sensitive to errors in both calibration and coordinate positions. It is therefore of great importance to study how much error that is introduced by the interpolation of coordinates. In general, the further away features have moved between the images, the higher the uncertainty of the interpolated feature position. One major factor in this error is the distance that the robot has traveled between the two left images L1 and L3. If the robot only has moved a few centimeters, the scene has not changed much between the images and the error is small. If the robot has traveled a longer distance, the linear interpolation is less accurate.

Another major factor in the interpolation error is the distance to the objects in the scene. If objects are far away, the images would not differ substantially if the robot moves straight forward. If objects are close to the robot, the corresponding features would move quite a bit between the images. If the robot in addition is turning between the images, then all features move even more.

The interpolation error is obtained by comparing the true coordinates of a real synchronized image with the interpolated coordinates. Three images (L1, L2, and L3) are

acquired with one camera to estimate the importance of the interpolation error. Features are extracted in the three images in order to find correspondence in all images using the three-way-matching method in Goncalves et al. [5].

For every matching feature, the coordinate between L1 and L3 is interpolated so that it matches the timestamp of L2. The interpolation error is computed as the Euclidean distance between the true coordinate in L2 and the interpolated coordinate. The median error from all the features in one dataset correspond to one cross in Figure 4. The figure shows the median errors from ten different datasets at different distances between images L1 and L3. It can be seen that the interpolation error is approximately linear with the distance between the two views.

B. 3D Structure

A good way to test the accuracy of the 3D structure is to generate a structure that is well known and evaluate how well the estimated 3D structure fits the actual structure. The reconstruction of a feature-rich perpendicular wall seen from above is shown in Figure 5. In this experiment Logitech 3000 webcams have been used with an image size of 320×240 pixels. The three images in the stereo system are captured after 0, 1 and 3 cm of motion using a robot that has the cameras separated 30 cm. The encircled features in the right figure are features classified as outliers in the outliers detection phase (more existing outliers were found and removed but not shown because they are located outside the frame of the figure).

A simple way to calculate the accuracy of the stereo algorithm is to quantify how well the features align to two lines, and how perpendicular the lines are. The perpendicular wall in figure 5 is captured 16 times from slightly different angles (total of 2900 features). The feature points have an average distance of 2.4 cm from their corresponding lines and the median distance is 1.6 cm. Furthermore, 95% of the features are closer than 6.9 cm, and only 1% is farther away than 16 cm.

The ground truth angle measured between the walls is 89.5 ± 0.3 degrees, calculated by measuring the length of the walls and the diagonal, and then applying the cosine rule. The angles on the 16 datasets have a mean angle 88.5 degrees and a median angle 88.2 degrees. The standard deviation of the angle calculations in the datasets is 1.7 degrees, so the whole range of the ground truth angle falls within one standard deviation from both the mean and median values.

It can be seen in the left figure that some features are outliers and some are not outliers but still a few cm away from the wall. It is obvious that small errors always exist in the triangulation parameters. The right figure shows the sensitivity of the 3D structure to small errors for the two systems. Small ellipses (that look like lines because they are so thin) are drawn around each 3D point and represent the 3D uncertainty volume at 2σ seen from above. The sensitivity is

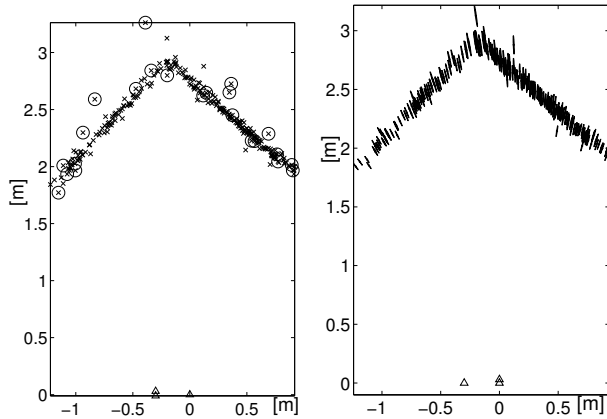


Fig. 5. Reconstructed 3D structure of a perpendicular wall seen from above. The crosses in the left image are all the found features, and the encircled stereo features are classified as outliers in the outliers rejection phase. Camera positions are presented as triangles. The right image shows triangulation sensitivity to small errors in image coordinates. The area within each ellipse (look like lines) corresponds to a 2σ uncertainty area or possible locations of the true 3D coordinate.

calculated by numerically estimating the covariance of the error in the triangulated 3D point $V(X)$, using $V(X) = JV(\psi)J^T$, if J is the Jacobian with respect to the different triangulation parameters ψ . The triangulation parameters ψ are assumed to be uncorrelated, and the small variance in pose is neglected. The coordinate variances are estimated as the mean square error between the re-projected image coordinates and the true coordinates used in the triangulation. The ellipses are drawn around a 2σ error.

V. CONCLUSION

The purpose of this system was to enhance the accuracy of the 3D structure in a vSLAM system using two cheap unsynchronized cameras. A method for stereo matching between the views is presented, as well as techniques for powerful outliers rejection that easily can be applied on any stereo matching problem.

Assuming that the system is able to acquire images reasonably fast, the linear interpolation of coordinates is a good approximation. If the interpolation error is small and the cameras are well calibrated, then the 3D structure normally has an accuracy of a few cm for a 3D structure located up to a couple of meters distance. A lot of effort was put into finding the correct stereo matches, and removing outliers. Several of the outliers detection algorithms are based on finding features that are different from the majority. This has been shown to be powerful and efficient, and typically results in less than one percent outliers.

VI. FUTURE WORK

The matching algorithm could be optimized. In this implementation, similar matching features are sought in L1 and L3 several times. The algorithm searches for good matches by both searching from L1 \rightarrow L3 and from L3 \rightarrow L1, which often results in the same candidate matches. This

search method can be modified to reduce processing time. A faster matching method is presented in Svedman's Master's thesis [16].

Sometimes a scene contains repetitive patterns, such as wall or mat patterns. The feature databases will in these cases contain a large set of similar features. Some features, on the other hand, are very unique and are rarely confused with others. In this implementation, a constant number of similar features are extracted from the databases. A good idea would be to allow an adaptive number of features to be extracted depending on how many similar features that exist in the database. In fact, Lowe has already suggested to measure the ratio between the SIFT distance to the most similar feature and the second most similar feature [10].

REFERENCES

- [1] Jean-Yves Bouguet. Camera calibration toolbox for Matlab. Website. <http://www.vision.caltech.edu/bouguetj/>.
- [2] D. C. Brown. Decentering distortion of lenses. In *Photometric Engineering*, volume 32, pages 444–462, 1966.
- [3] Y. Caspi and M. Irani. A step towards sequence-to-sequence alignment. In *Proc. 17th IEEE Conf. On Computer Vision and Pattern Recognition, Santa Barbara, California*, pages 8–15, 1998.
- [4] Y. Caspi and M. Irani. Alignment of non-overlapping sequences. In *Proc. 8th Int'l Conf. on Computer Vision*, pages 76–83, Vancouver, 2001.
- [5] L. Goncalves, E. Di Bernardo, D. Benson, M. Svedman, J. Ostrowski, N. Karlsson, and P. Pirjanian. A visual frontend for simultaneous localization and mapping. In *Proc. of Int. Conf. on Robotics and Automation (ICRA)*, 2005.
- [6] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049, 2000.
- [7] J. Heikkilä and O. Silvén. A four-step camera calibration procedure with implicit image correction. *Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn.*, 1997.
- [8] N. Karlsson, E. Di Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, and M.E. Munich. The vSLAM algorithm for robust localization and mapping. In *Proc. of Int. Conf. on Robotics and Automation (ICRA)*, 2005.
- [9] N. Karlsson, L. Goncalves, M.E. Munich, and P. Pirjanian. The vSLAM algorithm for navigation in natural environments. *Korean Robotics Society Review*, 2(1):51–67, 2005.
- [10] David Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
- [11] K. Mikolajczk and C Schmid. A performance evaluation of local descriptors. *IEEE Conference on Computer Vision and Pattern Recognition*, June 2003.
- [12] J. A. Nelder and J Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.
- [13] D. W. Pooley, M. J. Brooks, A. J. van den Hengel, and W. Chojnacki. A voting scheme for estimating the synchrony of moving-camera videos. In *Proc. Int'l Conf. on Image Processing*, Barcelona, Spain, 2003.
- [14] I. Reid and A. Zisserman. Goal-directed video metrology. In R. Cipolla and B. Buxton, editors, *Proceedings of the 4th European Conference on Computer Vision, LNCS 1065, Cambridge*, volume II, pages 647–658. Springer, April 1996.
- [15] G. Strang. *Linear Algebra and it's Applications*. Harcourt Brace Jovanovich, Publishers, 1986.
- [16] M. Svedman. 3-D structure from stereo vision using unsynchronized cameras. Master's thesis, Royal Institute of Technology (KTH), Stockholm, 2005.
- [17] G. Wahba. *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics, 1990.
- [18] L. Wolf and A. Zomet. Correspondence-free synchronization and reconstruction in a non-rigid scene. In *Proc. of Workshop on Vision and Modeling of Dyanmic Scenes*, Copenhagen, 2002.

- [19] C. Zhou and H. Tao. Dynamic depth recovery from unsynchronized video streams. In *Proc. 21st IEEE Conf. On Computer Vision and Pattern Recognition*, volume II, pages 351–358, Madison, Wisconsin, 2003.