Combined feature based and shape based visual tracker for robot navigation

M. Deans, C. Kunz, R. Sargent, E. Park, and L. Pedersen
QSS Group / Autonomy and Robotics Area
NASA Ames Research Center
Mailstop 269-3, Moffett Field, CA 94035, USA
{mdeans,ckunz,rsargent,epark,lpedersen}@arc.nasa.gov

ABSTRACT

We have developed a combined feature based and shape based visual tracking system designed to enable a planetary rover to visually track and servo to specific points chosen by a user with centimeter precision. The feature based tracker uses invariant feature detection and matching across a stereo pair, as well as matching pairs before and after robot movement in order to compute an incremental 6-DOF motion at each tracker update. This tracking method is subject to drift over time, which can be compensated by the shape based method. The shape based tracking method consists of 3D model registration, which recovers 6-DOF motion given sufficient shape and proper initialization. By integrating complementary algorithms, the combined tracker leverages the efficiency and robustness of feature based methods with the precision and accuracy of model registration. In this paper, we present the algorithms and their integration into a combined visual tracking system.

TABLE OF CONTENTS

1 INTRODUCTION
2 A TALE OF TWO TRACKERS
3 RESULTS
4 CONCLUSIONS

1. INTRODUCTION

Goal level, single cycle activity commanding for planetary rovers requires a high degree of robotic autonomy. The 2009 Mars Science Laboratory (MSL) rover will be required to navigate to a scientifically relevant feature from a distance of 10 meters away and place a contact instrument within 1 cm of the specified location. This capability will require the rover to track specified points with centimeter precision and servo directly to them. Because features are selected for scientific relevance, they are not necessarily those features which best facilitate appearanc- based visual tracking.

We have developed a combined feature based and shape based visual tracking system that leverages the benefits of each method in a complementary manner.

In order to handle large errors in platform motion prediction, reduce tracking frequency, and handle targets which do not facilitate unambiguous appearance based matching, the system uses one tracker which makes use of invariant feature detection and matching. The feature detector finds large populations of features around the feature of interest. By matching features across a stereo pair, as well as matching pairs before and after robot motion, the tracker can quickly compute a 6-DOF motion. In a static environment this 6-DOF transformation describes the motion of the tracked point. RANSAC is used to provide robustness to errors during feature detection and matching. Because the recovered motion is incremental, compounding the transformations found by the tracker leads to target drift over time. This is compensated by making use of a shape based tracker.

The shape based tracker employs 3D terrain model registration based on nonlinear optimization. Model registration can provide a strong cue for recovering 6-DOF motion if the models have sufficient shape and the optimization is initialized sufficiently close to the solution. By using the output of the feature based tracker to initialize the registration, we are able to align the current target view to the original view, thereby eliminating drift incurred by compounding the incremental motions recovered by the feature based tracker.

The combined tracking system is capable of tracking user-specified points for robotic navigation with centimeter level accuracy over distances on the order of ten meters. This tracking system is a critical component of the integrated single cycle instrument placement work demonstrated at NASA Ames Research Center.

2. A TALE OF TWO TRACKERS

Our robot navigation and instrument placement system uses two vision based tracking methods to provide fast, accurate incremental updates to the target location estimate during robot navigation as well as high precision error correction when the rover reaches an intended science target. Both of these tracking methods make the assumption that the target and the scene around it do not change, i.e. that the world is physically static. Lighting conditions may change, since our system may operate autonomously for a few hours.

The first method is an appearance based method that uses the SIFT algorithm[1] at its core. The SIFT algorithm is used to reliably find interesting points in stereo cameras and to find

0-7803-8870-4/05/$20.00/© 2005 IEEE
IEEEAC paper # 1297

putative matches between image pairs before and after motion. Stereo cameras provide 3D point locations for matched interest points in stereo views and matched 3D points are used to recover a rigid transformation aligning the points. Robust estimation makes the method tolerant of outliers and mismatches. Ultimately each target might be tracked using hundreds or thousands of matched points, providing a high degree of redundancy and accuracy.

Given two meshes generated from two different views, the second tracker method makes use of a virtual range sensor in order to determine depth at each correspondence point between the two meshes. By minimizing the difference between the rendered depth at each point, we can extract a rigid transformation that aligns the two models, thereby allowing us to determine the coordinate transformation between views.

Our combined tracker takes advantage of both of these methods to provide an integrated visual tracking infrastructure which is fast and robust during a traverse, and can provide bounded error at the end of a target approach.

Feature based tracker

Many feature based trackers operate by matching a chosen template to an area of interest in successive images. The search is often done using an exhaustive correlation or convolution, which can be expensive when precise predictions are not available or large camera motions must be tolerated. These trackers may offer the user the flexibility to specify the template, but the specified template may not be amenable to tracking due to low visual texture or changing appearance during motion. In addition, if the tracker only keeps track of one nominal target point, it is brittle in the event of a mismatch, and vulnerable to occlusions or changing viewpoints or other physical constraints.

The appearance based tracking algorithm used in our system uses large numbers of image features matched across stereo pairs. Feature matching is done using the SIFT algorithm[1]. The SIFT algorithm consists of two steps. The first step is the extraction of *interest points* from an image. Interest points are local maxima in scale space, found by searching for points in a Laplacian image pyramid[2] with higher values than neighbors in x, y and the scale dimension. The interest operator used by SIFT is invariant to rotation, translation, and spatial scale[1]. Once these interest points are found, a local orientation is estimated. The local image patch is then used to compute a feature vector, or *descriptor*, which is computed using some edge statistics in the neighborhood of the interest point, where the neighborhood is defined by the location, orientation, and scale recovered by the interest operator. This means that a large number of interest points are identified in image pairs under Euclidean or approximately Euclidean transformations in the images. The descriptors also tend to be fairly robust, so that a nearest neighbor search in feature space tends to find a large number of matched points in two images.

Our 3D SIFT based tracker uses features extraced from four images–two stereo pairs–to recover the incremental motion of the tracked target in the robot coordinate frame. We refer to one stereo image pair as $\{L_i, R_i\}$. From these images the SIFT algorithm extracts and matches features $\{l_i, r_i\}$ between the images, providing matched pairs of image points $z_i = (l_i^T, r_i^T)^T$. The 3D location x_i corresponding to the matched pair z_i is recovered through stereo,

$$x_i = f(z_i) \tag{1}$$

By arbitrary choice, the SIFT descriptor for the left image point l_i is taken as the descriptor for the 3D point x_i.

When the next image pair $\{L_{i+1}, R_{i+1}\}$ is acquired, matched features $z_{i+1} = (l_{i+1}^T, r_{i+1}^T)^T$ are found and 3D points $x_{i+1} = f(z_{i+1})$ are recovered. Using the SIFT descriptors from x_i and x_{i+1}, putative matches can be found between the 3D points extracted before and after robot motion.

Once a set of putative 3D point matches are found, we estimate the 6-DOF transformation from one view to the next using Horn's method[3] and RANSAC[4]. Horn's method will find the least mean square rotation and translation between a set of matched points in closed form[3]. However, because Horn's method minimizes a second order cost function, errors (outliers) in SIFT matching either between image pairs or between the 3D points can cause arbitrarily large errors in the recovered transformation parameters. To identify and eliminate outliers we use the robust estimation algorithm RANSAC[4] to find the transformation that is consistent with the largest number of points, or inliers.

Inliers are defined as those putative matches $\{x_i^{(j)}, x_{i+1}^{(j)}\}$ such that

$$||x_{i+1}^{(j)} - T_i^{i+1} x_i^{(j)}|| < \tau \tag{2}$$

where τ is a threshold. Currently we use $\tau = 3$ cm and repeat the RANSAC loop for $M = 100$ times, which takes negligible time since Horn's method is very fast. RANSAC returns the transformation with the largest consensus, and the list of matches in the consensus set. To further improve the estimate we use the consensus set to re-estimate the transform with all inliers. If the set of inliers changes we continue to re-estimate the transform until the consensus set converges. We denote the resulting transformation by T_i^{i+1*} and the inlier set by $\mathcal{J}$. The steps above are also shown in Figure 1.

Once the rigid transformation T_i^{i+1*} is computed, the tracked feature location is simply updated by applying the transformation to the target location

$$x_0^{i+1} = T_i^{i+1*} x_0^i \tag{3}$$

Note that mismatches may occur in two different steps in the tracking algorithm. Mismatches between l_i and r_i will lead to erroneous coordinates for x_i. Mismatches between points $x_i^{(j)}$ and $x_{i+1}^{(j)}$ will lead to 3D point pairs which are not consistent with a single rigid body motion. Both of these kinds

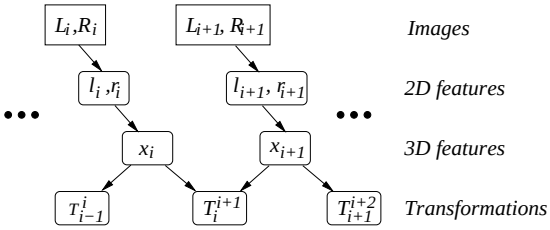


Figure 1. SIFT based tracker diagram

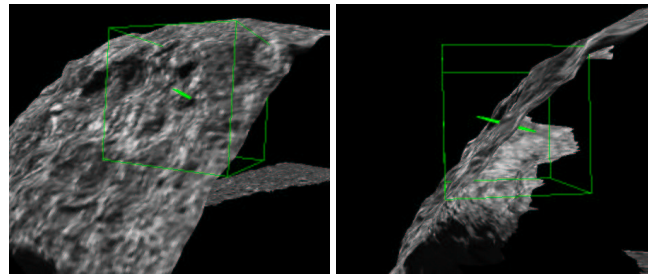


Figure 2. Uncertainty in the 3D coordinates of the initial target selection due to stereo errors

of outliers are handled by the robust absolute orientation. No explicit outlier rejection is needed in the 3D feature extraction prior to solving for absolute orientation.

Uncertainty

As the feature based tracker tracks a science target, two measures are used to estimate the performance of the system. The first is the uncertainty in the target location represented by a 3×3 covariance matrix over the XYZ location of the tracked feature, which is useful for geometric reasoning about the precision of the target location estimate for camera pointing and target handoff. The second is a single number representing a qualitative, overall confidence measure for the tracker which is useful for planning and execution purposes and detecting tracking failures.

The tracker uncertainty takes into consideration the initial target location specification as well as compounding the uncertainties in all of the tracker updates. The initial target location uncertainty is computed assuming a half-pixel standard deviation in the user specified location in the reference camera image, as well as an uncorrelated half-pixel standard deviation in the stereo disparity matching in the other stereo camera. The initial location and its covariance matrix are found by taking the unscented transform[5] of equation (1) above with $z_0 = (l_0^T, r_0^T)^T$ and

$$P_{zz} = \sigma^2 \mathbf{I}_{4\times 4} \tag{4}$$

with $\sigma = 1/2$ to yield the 3D location x_0 and 3×3 covariance matrix P_{xx}.

At each tracker update, the RANSAC method above is used to find the set of inlying matches that can be used to compute the dominant rigid transformation. However, Horn's method returns only a point estimate, without any information about the uncertainty in the estimate. In order to compute the covariance of the estimator, we use bootstrap[6]. Analytic approaches to propagating uncertainties through Jacobians of the norm minimized by Horn's method do exist[7], but the non-parametric approach we use is theoretically sound, trivial to implement, and makes significant reuse of the estimator code already implemented.

Bootstrap is a Monte Carlo method. To compute a bootstrap estimate of the covariance of the absolute orientation estimate, we generate a population of matched point sets from the inlier set $\mathcal{J}$ by sampling with replacement to yield B bootstrap sets $\mathcal{J}^b$. For each set of matches $\mathcal{J}^b$, we compute the transform T_i^b using Horn's method and recover the transform parameters θ^b. Our current implementation recovers translation and Euler angles[1], but other rotation representations are equally feasible. From the population of estimates θ^b, an empirical covariance is computed,

$$P_{\theta\theta} = \frac{1}{B}\sum_{b=1}^{B}(\theta^b - \theta_i^*)(\theta^b - \theta_i^*)^T \tag{5}$$

where θ_i^* is the transform parameters corresponding to the optimal estimate T_i^{i+1*}. The covariance matrix for the updated location of the feature is then given by an unscented transform on the update

$$x_0^{i+1} = T(\theta^i)x_0^i \tag{6}$$

with x_0^i and P_{xx}^i from the previous tracker update, and θ_i^* and $P_{\theta\theta}^i$ from Horn's method, RANSAC, and bootstrap. The notation $T(\theta^i)$ indicates the rigid transformation parameterized by the rotation and translation parameters θ^i. Note that because tracking is done in an incremental fashion, the covariance P_{xx} is monotonically growing during a tracking run, i.e. the tracker accurately models the fact that incremental updates with small errors will compound into a larger drift over time. Our system typically has incremental errors on the order of millimeters and milliradians per tracker update, so that a single target approach accumulates only centimeters of error. The 3D model registration step described below is designed to recover from this potential drift.

In addition to the geometric uncertainty in target location represented by the covariance matrix P_{xx}^i, the tracker maintains a single confidence value as a qualitative measure of how well the target is being tracked. This number is computed assuming a simple function of the number of inliers found by

[1] Euler angles can present problems due to singularities, and may not be amenable to representation by a Gaussian (mean and covariance). However, this work is applied to a surface rover with limited roll and pitch angles, avoiding the singularities in the representation, and the absolute orientation estimates tend to be highly overconstrained and yield very small covariance matrices, so the representation only needs to be accurate over a small neighborhood of the parameter space.

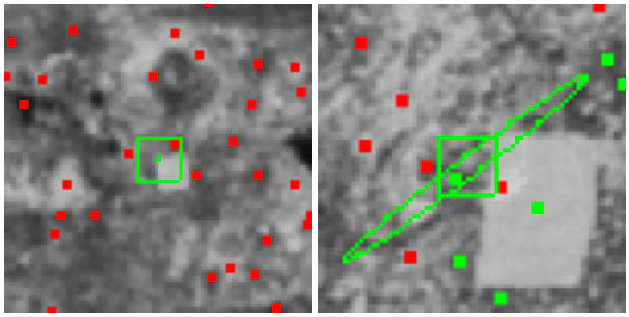


Figure 3. Tracker result with confidence interval shown as an ellipse around the tracked point. The tracker was initialized with the upper left corner of a fiducial on a rock. Note that the fiducial was not explicitly used in the tracking, but placed to facilitate performance evaluation.

RANSAC above, that is the confidence $\mathcal{C}$ is given by

$$\mathcal{C} = 1/(1 + e^{a(|\mathcal{J}|-N)}) \tag{7}$$

where through experimentation we have set $N = 30$, and $a = 0.1$. This confidence measure reflects the fact that if fewer than N inliers are found, then the tracker uncertainty should be low while if significantly more inliers are found then the tracker confidence should be high.

These confidence measures are somewhat ad hoc, but are only intended to capture some useful qualitative information about tracker performance. The confidence is used by the onboard executive to determine when the risk of losing a target is high enough to warrant a change in activity, e.g. to approach a different target with higher expected utility. In our experiments the tracker tends to either find a large number (hundreds) of matches or very few, and the overall system typically does what the rover operators would expect by identifying tracking failures and aborting when necessary.

Starting with the target location x_0^i, covariance matrix P_{xx}, point locations x_i and descriptors d_i from the previous time step, the tracker update proceeds as follows:

1. Find matching SIFT features l_{i+1} and r_{i+1}
2. Recover point locations x_{i+1}
3. Find putative matches between x_i and x_{i+1}
4. Repeat M times:
   (a) Choose 3 putative matches at random
   (b) Find rigid transformation T_i^{i+1}
   (c) Find the number of inliers (consensus)
5. For the best consensus set $\mathcal{J}$,
   (a) Compute θ^* using matches $\mathcal{J}$
   (b) Find inliers $\mathcal{J}$ under transform $T(\theta^*)$
   (c) If inlier set changes, repeat
6. Compute confidence $\mathcal{C}$ based on $|\mathcal{J}|$ using (7)
7. Compute $P_{\theta\theta}$ using Bootstrap
8. Compute x_0^{i+1} and P_{xx}^{i+1} using (6)
9. Return x_0^{i+1}, P_{xx}^{i+1} and $\mathcal{C}$

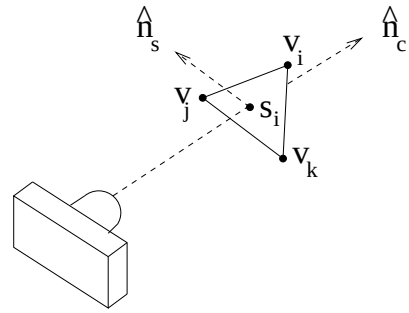


Figure 4. Each pixel in the range image is predicted by rendering the corresponding mesh facet into a virtual range sensor.

3D shape based tracker

The 3D shape based tracker uses terrain model registration to recover 6-DOF motion from stereo cameras. Tracking is performed by registering successively acquired terrain models of the target area to the initially acquired model of the target. By using an initial target template throughout the tracking cycle, successful registration to the current view at each step provides an an estimate of the goal location that does not drift over time.

For every pixel in the left camera image for which a correspondence is found in the right camera image, our stereo algorithm estimates the depth to that point. These depth estimates are combined to produce a 3D model of the surface. If two models of a surface are made from different locations, the rigid transformation that aligns the two models can be used to determine the coordinate transformation between views.

The surface models are represented by triangulated meshes with verticies $\mathbf{v}$ and $\mathbf{v}'$. If the two 3D models contain some region of overlap, there is a rigid transformation that aligns the overlapping regions. We represent the rigid transformation using the parameter vector $\mathbf{p} = (x, y, z, \alpha, \beta, \gamma)^T$ corresponding to 3 translational and 3 rotational degrees of freedom. These parameters define a transformation matrix $\mathbf{T_p}$. If $\mathbf{p}$ is the parameter describing the transformation between surfaces $\mathbf{v}$ and $\mathbf{v}'$, then for every pair of corresponding points $\mathbf{v}_i$ and $\mathbf{v}'_i$ the relationship

$$\mathbf{v}'_i - \mathbf{T_p}\mathbf{v}_i = 0 \tag{8}$$

holds. With real observations this equality will not hold exactly.

Our mesh registration approach projects these two models into a virtual range sensor view and minimizes the difference between the rendered depths at each point. The rendering takes $O(n)$ operations, where n is the number of pixels in the virtual range sensor. For each triangle on the mesh $\mathbf{v}'$, the vertices $\mathbf{v}'_i$, $\mathbf{v}'_j$, and $\mathbf{v}'_k$ are projected onto the image plane. For every pixel inside that triangle, the location of the intersection of the camera ray $\mathbf{c_z}$ and the facet of the mesh is a

point $\mathbf{s}'_i$, given by

$$\mathbf{s}'_i = \alpha_i \mathbf{v}'_i + \alpha_j \mathbf{v}'_j + \alpha_k \mathbf{v}'_k \tag{9}$$

with $\alpha_i + \alpha_j + \alpha_k = 1$. The depth to the intersection point is the z coordinate in the camera frame,

$$z_i = \hat{\mathbf{n}}_c \cdot \mathbf{s}'_i \tag{10}$$

The vector of all depths z_i is denoted $\mathbf{z}$. The surface model $\mathbf{v}'$ does not move during registration, so $\mathbf{z}$ is a constant.

The depth to the point $\mathbf{v}_i$ changes with transformation $\mathbf{p}$.

$$\begin{aligned} \mathbf{s}_i &= \mathbf{T_p}(\alpha_i \mathbf{v}_i + \alpha_j \mathbf{v}_j + \alpha_k \mathbf{v}_k) \\ h_i(\mathbf{p}) &= \hat{\mathbf{n}}_c \cdot \mathbf{s}_i \end{aligned} \tag{11}$$

We define a robust objective function which is the sum of the absolute deviations between the projected depths:

$$J(\mathbf{p}) = \sum |h_i(\mathbf{p}) - \mathbf{z}_i| \tag{12}$$

Because the $J(\mathbf{p})$ has a local minima, we first perform a coarse correlation search in order to narrow down the location of the best solution. Our initial estimate of $\mathbf{p}$, $\mathbf{p_0}$ comes from the stereo SIFT-based tracker described earlier. Consider that $\mathbf{p}$ is decomposed into rotational component $\mathbf{r}$ and a translational component $\mathbf{t}$. Furthermore, consider that $\mathbf{t}$ is decomposed into:

$$\mathbf{t} = \mathbf{x}\mathbf{c_x} + \mathbf{y}\mathbf{c_y} + \mathbf{z}\mathbf{c_z} \tag{13}$$

where $\mathbf{c_x}$ and $\mathbf{c_y}$ are in the plane of the virtual range sensor, and $\mathbf{c_z}$ is the pointing direction of the sensor. Because a search over the 6 dimensions of $\mathbf{p}$ is expensive, we make a few approximations.

For small changes in $\mathbf{t}$, $h_i(x, y, z + \Delta z, \mathbf{r}) \sim= \mathbf{h_i}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{r}) + \mathbf{\Delta z}$. In other words, a change in transformation along the z-axis of the virtual range sensor by some distance Δz changes h_i by approximately the same amount. Our initial estimate of $\mathbf{r}$ is approximately correct.

These two approximations allow us to perform the correlation search across only two dimensions; the x-axis and y-axis of the virtual range sensor.

For every Δx and Δy searched, the transformation $\mathbf{p}$ is computed by translating initial estimate $\mathbf{p}_0$ by Δx and Δy and by translating in the directions of the x-axis and y-axis of the virtual range sensor. The correction Δz to z_0 which minimizes the objective function $J(x_0 + \Delta x, y_0 + \Delta y, z_0 + \Delta z, \mathbf{r_0})$ is calculated as follows:

$$\Delta z = median(h_i(x_0 + \Delta x, y_0 + \Delta y, z_0, \mathbf{r_0})) \tag{14}$$

As described above, the correlation search uses approximate knowledge of the three orientation parameters to search only over the sensor x and y coordinates, solving for the average difference in z. Once the correlation search finds an approximate solution, we optimize over all 6 rigid transformation parameters using Nelder Mead[8], which is a general local nonlinear optimization method. Nelder Mead only requires a cost function, not any derivative information, so the cost function in equation (12) is used directly. In order to avoid problems with early termination[8], we restart the Nelder Mead optimization twice after it converges. Figure 5 shows an example result of the depth error after Nelder Mead converges.

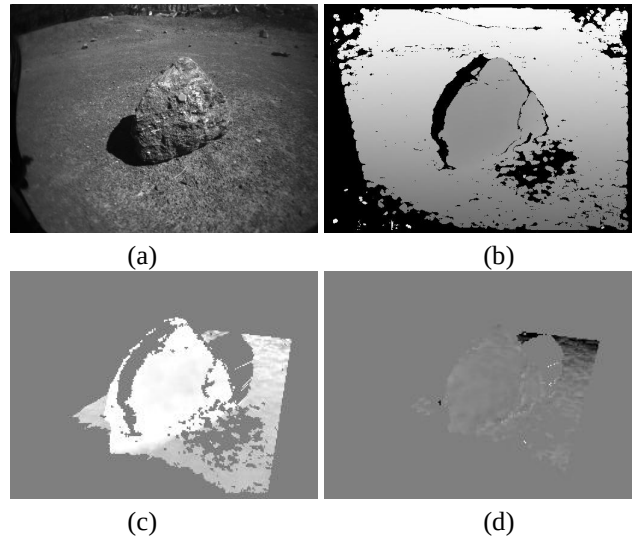


Figure 5. Registration result: (a) hazcam image (b) range image (c) depth error after range image correlation (d) depth error after nonlinear optimization

3. RESULTS

We ran the combined tracker through a simple test scenario on the K9 rover[9] in the NASA Ames Marscape. The test scenario was repeated over the course of September 22nd and 23rd, 2004. In the scenario, an operator selects three targets such that the straight-line distance from the rover's arm workspace at the starting position to the targets are approximately 5, 7.5, and 10 meters. The rover is then commanded to navigate to each of the targets in turn, while tracking their location with the combined tracker. The rover avoids obstacles using the CLARAty[10] navigator package, which is based on the Morphin algorithm[11].

After the rover arrives at each of the rocks, it is commanded to move its arm such that the CHAMP[12] camera contacts the rock as close as possible to the tracked target location. The instrument placement code analyzes the scene before the arm is moved, to determine the closest point on the rock that is safe for the CHAMP to touch, and plans a collision-free path for the arm.

Tables 1 and 3 show the results for these two days of testing. For each target, we record the elapsed time of the traverse

(which can be large if several obstacles have to be driven around), the accuracy of the target as tracked by the feature based tracker relative to 3-D models generated by the same camera pair as is used in the tracking, and the accuracy of the 3-D shape-based tracker used for handoff from one pair of cameras to another. The placement accuracy is also recorded, though the placement error can be arbitrarily large since the system places a higher priority on safety than on accuracy of placement. Placement figures are not available for September 22nd; a motor failure in one of the arm joints prevented successful placement.

The only failure in tracking occured in the feature based tracker for the second rock on September 23rd. In this case, the tracker failed just as the rover approached the rock and introduced a large cast shadow into the scene. Once the tracker was unable to find a transformation between subsequent images, it stopped updating the target location and fell back to dead reckoning. After the navigation was finished, the shape-based tracker was able to recover the target with accuracy comparable to the other experiments.

Target	1 (5m)	2 (7.5m)	3 (10m)
Time to reach target	21 mins	+42 mins	+17 mins
Tracker accuracy	0.68 cm	0.29 cm	1.3 cm
Hand-off accuracy	0.5 cm	2.7 cm	1.7 cm
Placement accuracy	N/A	N/A	N/A

Table 1. 9/22/2004 Performance

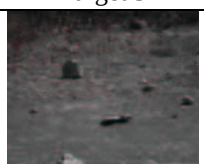
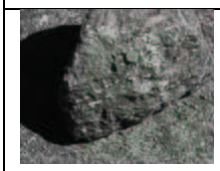
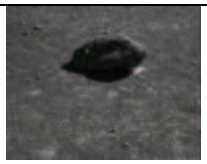
Target 1	Target 2	Target 3
		
		

Table 2. 9/22/2004 Tracker

Target	1 (5m)	2 (7.5m)	3 (10m)
Time to reach target	25 mins	+27 mins	+23 mins
Tracker accuracy	∼0.3 cm	failed	1.7 cm
Hand-off accuracy	1.3 cm	∼1.6 cm	3.2 cm
Placement accuracy	∼6.3 cm	∼11 cm	∼3 cm

Table 3. 9/23/2004 Performance

4. Conclusions

We started this work in an effort to increase the reliability of our previous system, which was based largely on the shape based method alone. We found that the shape based method

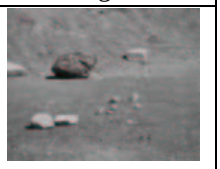
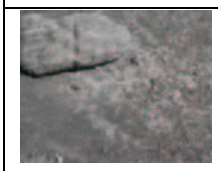
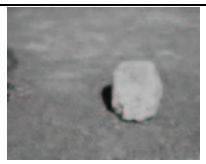
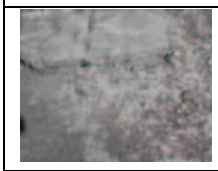
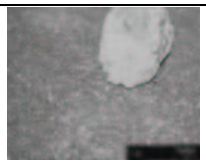
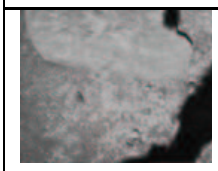
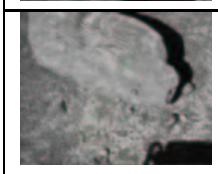
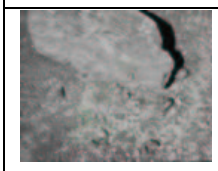
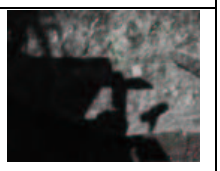
Target 1	Target 2	Target3
		
		
		
		
		
		

Table 4. 9/23/2004 Tracker

was quite reliable so long as the initial estimate of the target location was not incorrect by more than approximately half the width of the rock being tracked. Unfortunately, dead reckoning errors often led to our initial estimates being beyond this error.

Once we started developing the feature tracker based on SIFT, we found that it was reliable enough to use as the primary tracker in our navigation system, because the cumulative error over a traverse of less than 10 meters was typically well within the half-rock tolerance that the shape based tracker generally requires. We decided therefore to use the shape based tracker only as the last step, to hand the target off from the long-range cameras used for approach to the front cameras used for manipulation and instrument placement. Using the shape based tracker as the last step ensures that the rover is indeed using the same point on the designated rock for instrument placement as was initially chosen by the operators, since this point is chosen relative to another 3-D mesh of the rock, rather than relative to the rover or to another arbitrary coordinate frame. Since this change in usage, we've found

the system to be quite reliable. The two components have complementary strengths that yield a robust tracking system.

Since the experiments outlined in section 3, we have demonstrated the system operating several times, often tracking as many as five targets as the rover moves. To this point, we have executed at least one run where the rover has navigated to five targets in turn, and placed the CHAMP on each of the rocks with very little tracking error. In some instances the feature based tracker has lost the target due to occlusions, and was able to reaquire the target after the occlusion was removed. The fact that the tracker is able to provide a confidence measure allows the rover to fall back to dead reckoning if the confidence drops, and allows the rover's executive to change the course of action entirely if a target is lost. The tracker performs so well, however, that we typically have to introduce failures into the system in order to test the ability of the executive to cope with tracking failures.

ACKNOWLEDGMENTS

The authors of this paper would like to acknowledge the support of the Intelligent Systems and the Mars Technology programs. We would also like to acknowledge the support of other members of the Intelligent Robotics Group at NASA Ames Research Center.

REFERENCES

[1] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[2] P.J. Burt and E.H. Adelson. The laplacian pyramid as a compact image code. *IEEE Trans. on Communications*, pages 532–540, April 1983.

[3] B. K. P. Horn. *Robot Vision*. MIT Press, 1986.

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

[5] S. Julier and J. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*, 1997.

[6] K. Cho, P. Meer, and J. Cabrera. Performance assessment through bootstrap. *IEEE Trans. PAMI*, pages 1185–1198, 1997.

[7] C. F. Olson, L. H. Matthies, M. Schoppers, and M. W. Maimone. Robust stereo ego-motion for long distance navigation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 453–458, 2000.

[8] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C*. Cambridge University Press, 1988.

[9] Eric Park, Linda Kobayashi, and Susan Y. Lee. Extensible hardware architecture for mobile robots. In *IEEE International Conference on Robotics and Automation*, 2005, under submission.

[10] R. Volpe and et al. The claraty architecture for robotic autonomy. In *Proceedings of the 2001 IEEE Aerospace Conference*, 2001.

[11] C. Urmson, R. Simmons, and I. Nesnas. A generic framework for robotic navigation. In *Proceedings of the 2003 IEEE Aerospace Conference*, 2003.

[12] G.M. Lawrence, J.E. Boynton, and et al. Champ: Camera handlens microscope. In *The 2nd MIDP Conference, Mars Instrument Development Program*, 2000.

Matthew Deans *did his PhD work at the Carnegie Mellon University Robotics Institute, before joining the Intelligent Robotics Group at NASA Ames Research Center. He has developed simultaneous localization and mapping (SLAM) algorithms and sensor fusion based localization systems for rovers deployed in desert field sites in California, Nevada, Chile and Antarctica. He has also developed and supported machine vision ground tools used in MER mission science operations.*

Clayton Kunz *is the lead software engineer for the K9 rover at NASA Ames, and spends much of his time working on computer vision problems for robotics. He is also the head of the math and data structures subgroup of CLARAty, a collaborative project developing a software architecture for robotic autonomy. He's been an employee of QSS Group, and has had his hands inside K9, at Ames since 2001, before which he spent time making robot tour guides at a start-up company in Pittsburgh, PA. Clay holds BS and MS degrees from Stanford University, and lives in San Francisco.*

Randy Sargent *is software lead for the K9 rover target approach and instrument placement system at NASA Ames since 2002. In 1994 Randy co-founded Newton Research Labs, a machine vision company, with Anne Wright and Carl Witty. Randy led the Newton Labs team which won the 1996 and 1997 MIROSOT robot soccer tournaments, as well as the 1996 AAAI "Clean up the Tennis Court" robot contest. Randy left Newton Labs*

in 2000 to join Blastoff!, and in 2001 became Director of Open-Source Robotics at the KISS Institute for Practical Robotics. Randy holds BS and MS degrees from the Massachusetts Institute of Technology.

Eric Park *is the instrumentation lead for the K9 rover at NASA Ames Research Center. Currently a systems engineer within the Intelligent Robotics Group, he has interests in mobile robot architectures and computer vision. Eric received his BS in Electrical Engineering and Computer Science from the University of California, Berkeley in 2003 and has been building robots since 1994.*