2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

Robust and Efficient Stereo Feature Tracking for Visual Odometry

Andrew E. Johnson, Steven B. Goldberg, Yang Cheng and Larry H. Matthies, *Members, IEEE*

Abstract— **Visual odometry can augment or replace wheel odometry when navigating in high slip terrain which is quite important for autonomous navigation on Mars. We present a computationally efficient and robust visual odometry algorithm developed for the Mars Science Laboratory mission. This algorithm is a significant improvement over the algorithm developed for the Mars Exploration Rover Mission because it is at least four time more computationally efficient and it tracks significantly more features. The core of the algorithm is an integrated motion estimation and stereo feature tracking loop that allows for feature recovery while guiding feature correlation search to minimize computation. Results on thousands of terrestrial and Martian stereo pairs show that the algorithm can operate with no initial motion estimate while still obtaining subpixel attitude estimation performance.**

I. Introduction

VISUAL Odometry is the process that uses images to estimate the motion of a robotic vehicle. Unlike traditional wheel odometry, motion estimates from visual odometry are not degraded by the slip of the vehicle, so they can be used to reliably estimate motion in rugged and low traction terrain. However, the high computational cost and sensitivity to imaging conditions (e.g., terrain appearance, vehicle motion, camera parameters) of visual odometry must be overcome before visual odometry can be used reliably in real-time.

On the Mars Exploration Rovers (MER) the visual odometry algorithm [6] uses image feature tracking between stereo image pairs to estimate the translation and rotation between image captures. The software ensures vehicle safety by halting a dangerous drive when the rover enters a predefined keep-out zone and performs intermittent slip checks in slippery terrain, which enable reliable mid-drive imaging of science targets and reduces the overall number of days needed to reach science targets. The MER Visual Odometry algorithm (MER-VO) has been used extensively and effectively on both rovers on Mars, but it has some short-comings which led to funding of an update to the algorithm for the Mars Science Laboratory (MSL) mission, a rover mission to Mars scheduled to land in 2010.

Specifically, MER-VO requires a large amount of computation time. Given the extremely limited computational resources on board MER (20MHz RAD 6000 running dozens of tasks under VxWorks), the operations team rarely chose to run MER-VO and the autonomous hazard detection software during the same traverse. This presents a problem when autonomously navigating in rough terrain because hazard detection and avoidance depends on a reliable position estimate to navigate around hazards and wheel odometry cannot be depended on to provide position estimates of sufficient accuracy. MSL is a larger vehicle that can mechanically tolerate more rugged terrain than MER, so the ability to navigate precisely in rough terrain is desirable.

Occasionally, especially during the beginning of the mission, MER-VO sent a grossly incorrect motion estimate to the rover navigation system. This occurred mainly when the scene was low texture and/or a small high contrast element was in the scene (e.g., a rock or rover shadow). In many of these failure cases, the number of features tracked was low, but not low enough to throw out the motion estimate (because good motion estimates often have a low number of features as well). Increasing the number of features tracked when the motion estimate is correct relative to when the it is incorrect will make it easier to distinguish correct from incorrect motion estimates. MER-VO also failed to converge to a solution during many of the Opportunty rover's drives through sandy terrain due to lack of detected or well-tracked features. Increasing the number of tracks also improves the accuracy of the motion estimate.

MER-VO uses the on-board position from wheel odometry as an initial estimate to decrease run time. This estimate can be quite wrong due to slipping of the rover, which can lead to failure of MER-VO to compute a motion estimate. Unfortunately, when these estimates are wrong is precisely when visual odometry is needed most. Visual odometry should not depend on the input motion estimate.

This assessment leads to a list of desired improvements to the MER visual odometry algorithm:

- decrease algorithm run-time
- increase number of features tracked
- eliminate dependence on initial motion estimate

Because MER-VO has worked well enough and a lot of effort went into validating its performance before it was used on Mars, there was also a desire to minimize the changes to the MER algorithm. With these goals in mind an update to MER-VO was designed, implemented and tested. The update resulted in an integrated motion estimation and feature tracking algorithm that is very efficient while

Manuscript received September 14th, 2007. The work described in this publication was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract from the National Aeronautics and Space Administration. The Mars Technology Program funded development of algorithm and the Mars Science Laboratory Project funded testing.

Andrew.E. Johnson, is with the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109 USA (corresponding author: phone: 818-354-0357; fax: 818-393-5007; e-mail: aej@ jpl.nasa.gov).

Yang Cheng and Larry H. Matthies are with the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109 USA (e-mail: {ycheng,lhm}@ jpl.nasa.gov).

Steven B. Goldberg is with Indelible Systems Inc., Northridge, CA 91324 USA (e-mail: indeliblesteve@gmail.com).

tracking a large number of features and still not relying on an initial motion estimate. Due to the dramatic improvement in run-time and number of features made possible by this stereo tracker, it should be of general interest to the visual tracking and autonomous navigation community.

The paper begins by describing related work with an emphasis on the MER visual odometry algorithm. It then describes the updated algorithm with an emphasis on the integrated multi-resolution stereo tracker and motion estimation loop. Results are then presented for terrestrial and Martian image sequences which demonstrate the algorithm performance with respect to run-time, motion estimation accuracy and number of features tracked.

A. Related Work

The concept of visual odometry from stereo images has been around for quite some time. Initial work started with Moravec [9] and was then continued by Matthies [8]. Since that time, refinements have continued [11] culminating in an algorithm implemented in real-time flight software for the Mars Exploration Rover mission [6].

The MER-VO algorithm is based on feature tracking; features are selected and located in subsequent images using spatial correlation search. The feature tracking approach works better when there is an initial estimate of the motion, which is often the case for robotic vehicles, so this approach has been applied to many different navigation problems [1][2][7]. On MER, the attitude estimate is very accurate because it comes from inertial propagation of gyro measurements; the position estimate from wheel odometry can also be accurate in low slip terrains.

Recently a different approach based on feature matching has been used to perform real-time visual odometry [10][4]. In this approach features are selected and then matched based on a descriptor associated with the feature; no spatial correlation search is performed. These approaches require stable and salient descriptors to work well with large image motions, but the advantage is that they do not necessarily require an initial motion estimate. The Hirschmuller approach [4] uses a dense stereo depth map. Since computing resources are limited on Mars rovers, generating a dense depth map is undesirable. The Nister approach [10] assumes very small steps between images. But rover operators try to run VO as infrequently as possible, to minimize power consumption due to image acquisition and computing. This forces the VO algorithm to be reliable for large step sizes.

II. Algorithm Description

A. Definitions

In the description below *left* and *right* refer to the cameras locations, *first* and *second* refer to the order in which the stereo pairs are taken and *top* and *bottom* refer to the levels of an image pyramid where top is the coarsest level and bottom is the finest level of the pyramid. *Template* refers to a square image patch that is correlated with a larger *window* in another image. *Feature* is a pixel location and a *track* is the location of the feature in another image established through correlation. A feature is considered *valid* if it survives various algorithm checks all the way to the final motion estimation stage.

B. MER Visual Odometry Algorithm

An excellent description of the MER algorithm implementation including performance on Mars is given in [6]. The algorithm takes as input a two stereo image pairs, camera models, a set of algorithm parameters and an estimate of the motion between the images from wheel odometry and integration of inertial measurements. It outputs an estimate of the change in position and orientation of the cameras between stereo image captures and the covariance of this estimate. The algorithm steps are briefly described below because they form the basis for the new algorithm. The order of the steps and the use of image pyramids are shown in Figure 1.

Feature Selection: Features are selected in the bottom first left image using the Harris interest operator [3]. The image is broken up into square blocks and the best feature in each block is selected. This forces the features to span the image, which improves motion estimation accuracy. Since feature selection occurs in the input image, these features are expected to be good features to track in the coarser levels of the image pyramid, which is not always the case.

First Stereo Matching: The features selected in the first left image are searched for in the first right image using Pseudo Normalized Cross Correlation (PNC)[5]. Image pyramids are used to speed correlation. Down-sampling of the selected feature location to the top left image determines the location of the correlation template. The bounds of the correlation search window in the top right image are set by constructing a rectangular box that contains the projection of the feature ray, truncated by parameters for maximum and minimum feature depth, from the left camera into the right camera (an epipolar segment). Spatial correlation between the template and the search window is used to find the location of the template in the first top right image. The feature and its track are then up-sampled to the next finer level of the pyramid. A template is established in the left image, and it is correlated with a small search window that just surrounds the track location in the right image. This process repeats to the end of the image pyramid.

If a correlation peak is questionable (low or wide peak) then the feature is eliminated from further steps of the algorithm. If the correlation peak is good, then triangulation is used to generate a 3D point and its covariance. Checks on the distance between feature rays from left and right image are used to eliminate some bad tracks.

Motion Tracking: The remaining features are tracked from the first left to the second left image using the pyramidal PNC method described above. However, the bounds of the correlation search window are set through a different means. First the 3D point defined by the first stereo match is projected into the second left camera using the supplied

motion estimate. Then a single fixed parameter defines the square extent of the correlation search window around this pixel. The motion estimate and its associated uncertainty are not used to set the search window, and no accommodation is made for the fact that features that are far away may move in the images much less than features that are close. As we will show, this simple approach led to unnecessarily large search windows. An example of the motion tracking search windows for MER-VO is shown in Figure 4.

Second Stereo Matching: This step is similar to first stereo matching with the following exceptions. The template location is given by the motion track location in the second left image. Projecting this pixel into the second right image using the depth from the corresponding first stereo match defines the center of the search window in the second right image. The search window bounds are set based on fixed max and min depth parameters that are tighter than the global max and min depth parameters used in the first stereo track where depth is not known at all. As in the motion tracking stage, this method makes no use of a motion estimate to predict the change in depth to the feature, which can result in unnecessarily large correlation search windows.

Triangulation between the second left to and second right matched pixels is used to generate a 3D point and its covariance.

Motion Estimation: The corresponding 3D points and covariances from the first and second stereo pair are used to estimate the motion in three steps. First a rigidity test based on the difference of distances between points in each stereo pair (weighted appropriately by the point covariances [8]) is used to find outlier points. A weighted Least Median Squares technique is the applied to the remaining points to generate an initial motion estimate and eliminate points that do not agree with rigid the motion estimate. Finally, Maximum Likelihood Estimation is used to estimate the final motion estimate using the full point covariances. Motion estimation details are given in [8][11][6].

C. *MSL Visual Odometry Algorithm*

Inspection of the MER-VO algorithm indicates that there could be a number of improvements to make it more efficient, track more features and eliminate dependence on an initial motion estimate. First, the correlation search windows are set somewhat blindly which results in unnecessarily large search windows. Second, a significant number of features are eliminated by various checks throughout the algorithm without any attempt to recover them. Third, features selected at the bottom of the pyramid are expected to be good features to track at the top which can lead to loss of features.

The MSL Visual Odometry algorithm (MSL-VO) starts with the same stages as MER-VO but reorganizes them to make a more efficient algorithm that also tracks more features and does not require on an initial motion estimate. Timing analysis showed that correlation incurred the most computation, so improvements were focused there.

As shown in Figure 2, MSL-VO goes through all of the VO stages (from feature selection through MLE motion estimation) first using just the images at the top of the pyramid. The resulting motion estimate, location of valid features and the depth to valid features are all used to constrain the search for feature tracks when the stages in VO are applied to images at the next level of the pyramid. This process repeats until the bottom of the pyramid is reached.

This iterative application of the VO stages increases the number of features tracked because, as the pyramid is traversed, the correlation search windows shrink based on the improving confidence in the motion estimate. When the bottom of the pyramid is reached, the correlation search windows are so small that the chance of a incorrect feature track due to noise or ambiguity is also very small.

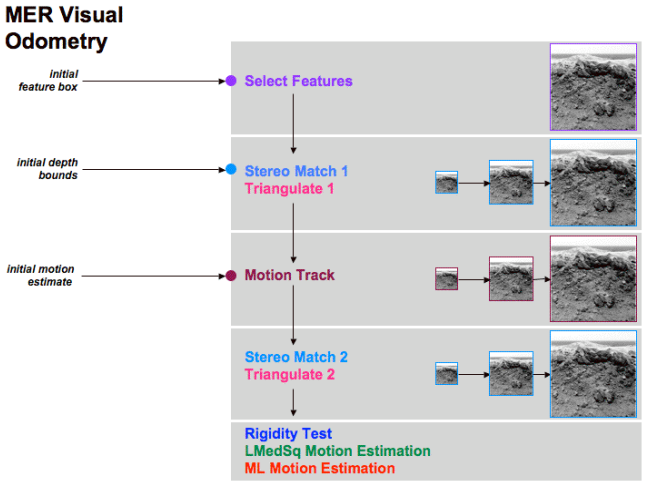

Figure 1. MER VO algorithm. Algorithm only passes through stages once.

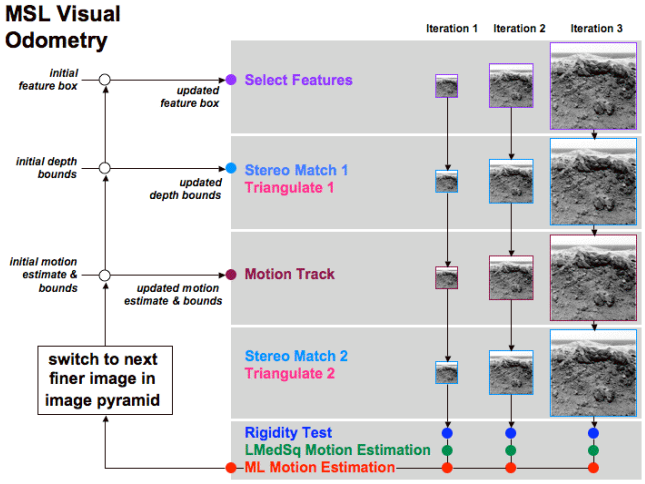

Figure 2. MSL VO algorithm. Algorithm passes through all stages at each level of the image pyramid. i

Computation is also reduced in this approach because the correlation search windows are set just large enough to contain the current motion estimate and its uncertainty. Also, as the pyramid is traversed, feature depths are used to refine the maximum and minimum depth parameters so that the first stereo matching search windows are as small as possible. Finally, feature selection at each level is only performed inside a box defined by the valid features from

the previous level, so searching for features that will not eventually be used in motion estimation is avoided. The algorithm details are given below.

Feature Selection: At the top level of the pyramid features are selected inside of a bounding box set by user specified parameters. This box can be used to avoid pieces of the rover in the field of view or to avoid the sky or foreground where tracking is known a-priori to be problematic. After the top VO iteration, this bounding box is updated to just surround the valid features from the top iteration (plus a user specified parameter to grow the box slightly to prevent excessive shrinkage as the pyramid is traversed). Features are selected in this box in the first left image at the next pyramid level. The idea is that the algorithm learns where the good features to track are located in the image so that it does not have to waste computations searching for features that will not survive to motion estimation. This approach also guarantees that the best features for tracking are selected for the image resolution at the current pyramid level, which is not the case for MER-VO. Figure 7 shows an example of the features that are selected at each pyramid level.

First Stereo Matching: At the top level of the pyramid, user specified parameters for the maximum and minimum depth to features define the correlation search window in the top second image. After the top VO iteration, these parameters are updated to just contain the depths to all the features from the top iteration (plus a user specified parameter to slightly grow the depths to prevent shrinkage of the depth bounds as the pyramid is traversed). These new bounds are then used to set the correlation search window at the next pyramid level. Here again the idea is to learn the depth to features in the scene so that excessive computation is avoided.

Because an entirely new set of features is selected for each pyramid level, it is not straightforward to pass the feature depth information from the previous level of the pyramid to the next (although we are looking into ways to do this with bounding planes instead of bounding depths). Given the desire to maximize the number of features tracked and given that the depths could be incorrect or not exist, it is debatable whether this is desirable. However, the consequence is that this step requires more computation because it searches the entire range of depths at each pyramid level instead of just at the top level of the pyramid, which is what the MER-VO does. However, resetting the depth search does increase the chance that the correct stereo match will be found for each selected feature, which would not be the case if the depths to features were somehow constrained based on previous possibly erroneous depth estimates. Thus the path we have taken adds robustness to the algorithm while still increasing the number of features that are tracked at a slight computational penalty. As will be shown the overall algorithm is still much faster. Figure 7 shows an example of the first stereo matching correlation search windows for each pyramid level.

As in MER-VO, triangulation generates a 3D point and covariance. Points with rays that are too far apart are eliminated. An additional check added for MSL-VO computes the angles between the rays; if the angle is too small then, because the covariance computation is ill-conditioned, the feature is eliminated. In general this eliminates points that are far away.

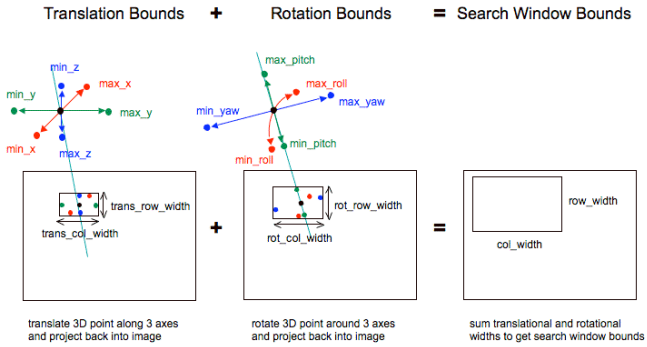

Figure 3. Setting of motion ttracking correlation search window using motion bounds.

Motion Tracking: The purpose of motion tracking is to track features from first left image to second left image. The position of the feature in the second left image depends on the depth to the feature, which is known from the first stereo track, and the motion between stereo pairs. The motion typically has an estimate, but it may have a large uncertainty which is unknown. The challenge is to translate the estimate and uncertainty into bounds on the correlation search windows.

An intuitive way for the user to specify the motion uncertainty is by placing bounds on the maximum and minimum motion uncertainty in all six degrees of freedom (x,y,z,roll,pitch,yaw). For example, suppose the rover is commanded to drive forward for 0.5 m over flat ground but that there is also the possibility of the rover slipping. It is reasonable to constrain the forward motion between full slip and no slip ($0.0m < x < 0.5m$). The cross track motion is probably small ($-0.2m < y < 0.2m$), and the vertical motion is definitely small ($-0.1m < z < 0.1m$). The roll, pitch and yaw are all small ($-5^\circ<roll<5^\circ$, $-5^\circ<pitch<5^\circ$, $-5^\circ<yaw< 5^\circ$). These constraints can be converted into a motion estimate and 12 motion bounds

$(x,y,z,roll,pitch,yaw)_{est} = (0.25, 0.0, 0.0, 0.0, 0.0, 0.0)$

$(x,y,z,roll,pitch,yaw)_{min} = (-0.25, -0.2, -0.1,-5,-5,-5)$

$(x,y,z,roll,pitch,yaw)_{max} = (0.25, 0.2, 0.1, 5, 5, 5)$

Initially, for tracking at the top level of the pyramid, the user specifies the motion bounds based on the rover characteristics and the drive command. The motion estimate, motion bounds and the known depths to the features are then used to set the search windows. The search window for a feature should be set such that all transformations within the motion uncertainty bounds project the feature into the search window. Enumerating all transformations and computing the corresponding projection is computationally infeasible, so a simpler, but conservative, approach is taken. As shown in Figure 3, search window

bounds are first computed for the translational uncertainty and then these are added to the search bounds from the rotational uncertainty.

For each feature the 3D point associated with the feature in the first stereo pair is transformed based on the motion estimate (translation and rotation). The projection of this point in to the second left image is called the nominal pixel. Then six points are constructed that are offsets from the transformed point along the coordinate axes with length equal to the respective translational motion bound (c.f., Figure 3). These six points are projected into the second left image and the bounding box of the projections is determined. This process is repeated for the rotational motion bounds except that the transformed point is rotated by each motion bound instead of translated.

These bounding boxes are combined as follows. Each bounding box has a left, right, top and bottom offset defined by the difference between a bounding box edge and the nominal pixel. The four edges of the final bounding box are defined by the sums of the left, right, top and bottom offsets from the translational and rotational bounding boxes.

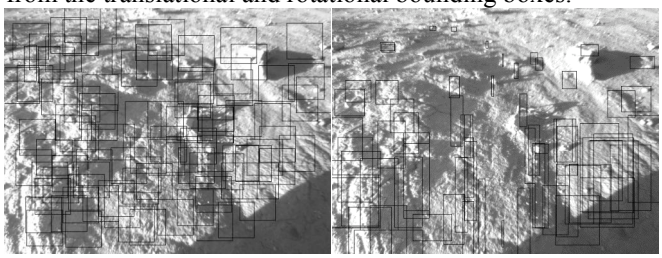

Figure 4. Comparison of MER-VO (left) and MSL-VO (right) motion tracking correlation search windows.

Figure 4 compares the MER search windows to the MSL search windows for a forward motion test case. As can be seen the MER search windows are all the same size while the MSL search windows change size based on the distance of the feature and the expect forward motion of the vehicle. The MSL search windows are smaller for more distant features because only the rotational uncertainty has an effect on these windows. Furthermore, the features in the center foreground have narrower search windows than the ones on the edge of the image because the features are expected to move mostly down while the ones on the edge will move down and out.

Figure 5 shows the result of motion tracking with these search windows. The red tracks in each picture indicate the tracks that have survived to motion estimation. In the MER-VO picture, the tracks in the center far field are tracked while the ones in the foreground were not tracked because the search windows were not set large enough and there is no chance to recover lost features. In contrast, the MSL-VO algorithm is able to track features over the entire image because the search windows were set based on the expected motion of the each feature. The end result is more features with less total correlations.

The user specifies the motion bounds at the top level of the pyramid, but for each subsequent level, the motion bounds are set based on the covariance of the motion estimate from the previous level (described later). As the image pyramid is traversed the motion estimate covariance shrinks. This forces the motion tracking search windows to shrink and become more consistent with the motion estimate. Since the correlation search windows are shrinking around the correct location of the feature, the chance of an incorrect feature track due to noise or ambiguity decreases as the pyramid is traversed. The end result is that more features are successfully tracked. Figure 7 shows an example of the motion tracking correlation search windows for each pyramid level.

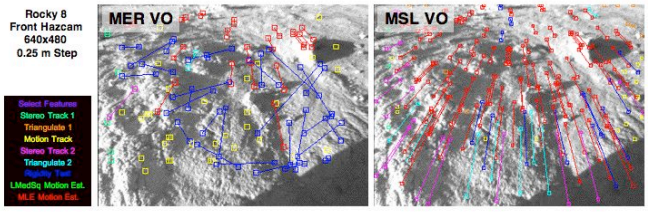

Figure 5. Motion tracks comparison between MER-VO and MSL-VO.

If the user would like to eliminate any dependence on the motion estimate or no motion estimate is available (e.g., if the on-board inertial measurement unit fails) then at top level of pyramid, the search window for each feature can be set to the entire image. The motion estimate from the top level is still used to set the search windows at the next level, so this global search in only used at the top level. This approach requires more computation, but as we will show it is still faster than the MER-VO approach with no degradation in algorithm robustness or accuracy.

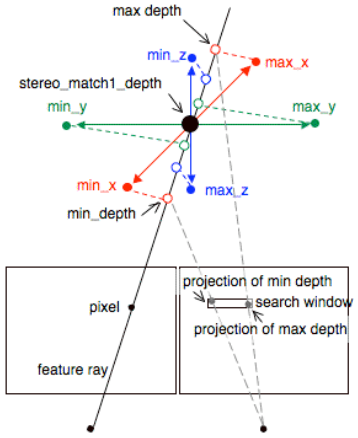

Figure 6. Figure for setting search bounds based on depth bounds.

Second Stereo Matching: Like in the motion tracking stage, the depth to features in first frame is known and motion is unknown, but bounded. Following the motion tracking concept the motion estimate and bounds are used to set the bounds on the maximum and minimum depth for each feature as follows. A 3D point along motion track pixel ray is constructed. The translational motion bounds are added to this point to generate six 3D points; rotational motion bounds are not used because the feature depth does not depend on rotational motion. As shown in Figure 6, these six points are then projected back onto the pixel ray to get the maximum and minimum depth for the feature. These

depth bounds are then used in the standard fashion to limit the stereo matching correlation search window. This process is repeated for each feature. Figure 7 shows an example of the second stereo matching correlation search windows for each pyramid level.

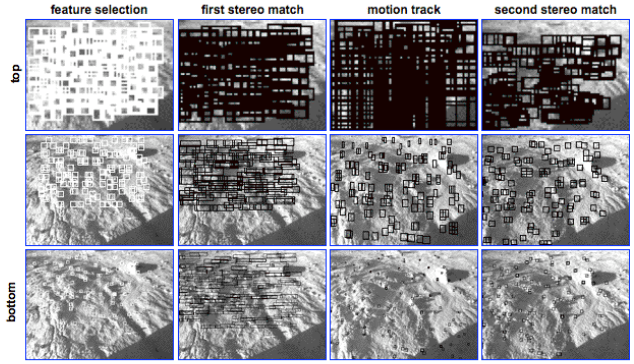

Figure 7. Example of feature selection and correlation search windows for each pyramid level.

Motion Estimation: MSL-VO uses the same rigidity test, Least Median Squares motion estimation and Maximum Likelihood Motion Estimation (MLE) that are used in MER-VO. The difference is that they are applied at each at each pyramid level. Following the notation in [6], the cost function minimized in MLE motion estimation is

$$\sum e_j^T W_j e_j \qquad (1)$$
$$e_j = P_{Cj} - RP_{Pj} - T, \quad W_j = \left[R\sum_{Pj} R^T + \sum_{Cj}\right]^{-1}$$

P_{Cj} is the position of a 3D point in the second (current) stereo pair, P_{Pj} is its position in the first (previous) stereo pair, R is a rotation matrix constructed from the roll,pitch and yaw estimates, T is a vector for the translational motion estimate and Σ_{Pj} and Σ_{Cj} are the covariances of the points in the first and second stereo pairs. The covariance of the motion estimate is

$$C = \left[\sum_i H_j^T W_j H_j\right]^{-1} \qquad (2)$$
$$H_j = \begin{bmatrix} J_j & I \end{bmatrix}, \quad J_j = \begin{bmatrix} R_x P_{Pj} & R_x P_{Pj} & R_x P_{Pj} \end{bmatrix}$$

where $R_{\{x,y,z\}}$ are the partial derivatives of R with respect to the three rotation angles.

In the MSL-VO algorithm, to improve numerical conditioning, the mean of 3D points is subtracted from the points in both stereo pairs before computing the MLE error term; the means are added back in to correct the translation estimate after MLE estimation is finished.

As mentioned previously, after the top level of the pyramid, the motion covariance is used to set the motion estimate bounds as follows. C is 6x6 matrix in (roll,pitch,yaw,x,y,z) order. An eigen-vector decomposition of C is computed

$$Ce_i = \lambda_i e_i \quad i = 1\ldots 6 \qquad (3)$$

with eigenvectors e_i and eigenvalues λ_i. e_i are principal directions of 6 dimensional ellipsoid and $\sqrt{\lambda_i}$ are the magnitudes of the axes. The motion bounds are a bounding box aligned with the (roll, pitch, yaw, x, y, z) motion axes. The motion bounds are set by scaling the principal axes by a user specified parameter (S) and then projecting them onto the motion space axes. The maximum and minimum projections over all six principal axes define the maximum and minimum motion bounds.

$$\begin{aligned} \text{max\_roll} &= S * \max_i(\sqrt{\lambda_i} e_i[0]) \\ \text{min\_roll} &= S * \min_i(\sqrt{\lambda_i} e_i[0]) \\ &\vdots \\ \text{max\_z} &= S * \max_i(\sqrt{\lambda_i} e_i[5]) \\ \text{min\_z} &= S * \min_i(\sqrt{\lambda_i} e_i[5]) \end{aligned} \qquad (4)$$

Figure 8 shows a cross section of the motion covariance ellipsoid and how its projection onto the motion axes do not completely contain the motion covariance. However the approximation used is very quick to compute and a fractional increase to the scale factor S will ensure that the motion covariance is completely contained.

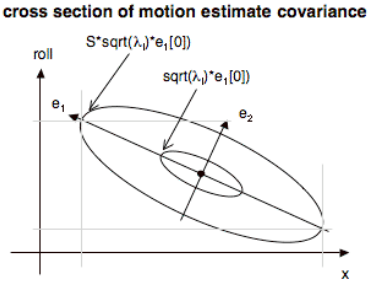

Figure 8. Cross section of motion covariance showing how it is used to set motion uncertainty bounds.

Validity Checking: In MSL-VO, number of features tracked at the end of all the visual odometry iterations is a very strong metric for the validity of the motion estimate. Another metric employed is the condition number of the motion covariance matrix

$$l_m = \max_i(\lambda_i)/\min_i(\lambda_i) \qquad (5)$$

which should be small for well conditioned motion estimates.

To check for collinear or compact configurations of features where motion estimation is unstable, the scatter matrix A for valid features is computed

$$A = \begin{bmatrix} \sum_i r_i^2 - (\sum_i r_i)^2 & \sum_i r_i c_i - \sum_i r_i \sum_i c_i \\ \sum_i r_i c_i - \sum_i r_i \sum_i c_i & \sum_i c_i^2 - (\sum_i c_i)^2 \end{bmatrix} \qquad (6)$$

An eigenvector decomposition of A is performed and the condition number, as defined is in Equation 5, is computed. If either condition number is larger than a threshold, then the motion estimate is considered invalid.

III. Analysis

Since efficiency and robustness are the primary motivation for the update to MER-VO, the main metrics tracked in this analysis are run time and number of features

tracked. In addition, the motion estimation accuracy and optimal rover step size are important factors for operation of the MSL rover.

A. Comparison between MER-VO and MSL-VO

The main objective of this work is to create an algorithm that is faster and tracks more features than MER-VO. As shown in Table 1, three test cases were selected for making this assessment. All run-times are with both algorithms running on the same computer, a 400 MHz R12000 SGI O2 processor. In each case, the MER-VO parameters were optimized to maximize performance while the MSL-VO parameters were not optimized.

The first test case was from the Rocky 8 research rover with wide field of view cameras 30cm above the terrain and a 25cm forward step. This is a challenging test case with lots of image motion between stereo pairs. The MSL-VO algorithm tracked 5.6x more features and took 15x less time than the MER-VO algorithm.

The second test case was from the MER-B Opportunity rover 45˚ FOV navigation cameras, which are 120cm above the terrain, and essentially no motion between stereo pairs. Even with no motion the MSL-VO algorithm is four times faster than MER-VO; the number of features tracked is comparable for the two algorithms because both are able to track features across the entire image.

The final test case is from a stereo pair mounted on the DARPA LAGR vehicle. The cameras have a 45˚ field of view, are 2m above the terrain and the vehicle step size is 50cm. This is the most similar to the MSL navcam configuration, so the results for this test case closely represent the expected improvements for MSL. For this test case, the algorithm is six times faster while still tracking 2.7 times more features. This is obviously a significant improvement.

TABLE 1: COMPARISON OF MSL VO TO MER VO

Test Case	MER VO		MSL VO		COMPARISON	
	run time	# feats. tracked	run time	# feats. tracked	run time decrease	feature increase
Rocky8	6.91s	17	0.46s	95	15x	5.6x
MER-B	0.53s	45	0.13s	54	4x	1.2x
LAGR	4.72s	29	0.76s	79	6x	2.7x

B. Mars Terrain Performance

MSL-VO must work on Mars imagery. To investigate this, the first 2757 of the down-linked MER-A VO navcam stereo pairs were run through MSL-VO. The same parameters were used for all cases. The MER onboard attitude estimates (generated by integrating gyro measurements during vehicle motion) were used for ground truth. The position estimates could not be used because they came from an unknown combination of wheel odometry and visual odometry, and consequently could not be trusted to convey the truth.

Figure 10 shows the error between the MSL-VO attitude estimate and the MER onboard attitude estimate. Errors are only shown for valid estimations where valid is defined as cases with more than 25 valid features used in motion estimation. All of the attitude errors are sub-degree and most of them are less the MER navcam pixel size of 0.17˚, which indicates very good estimation performance.

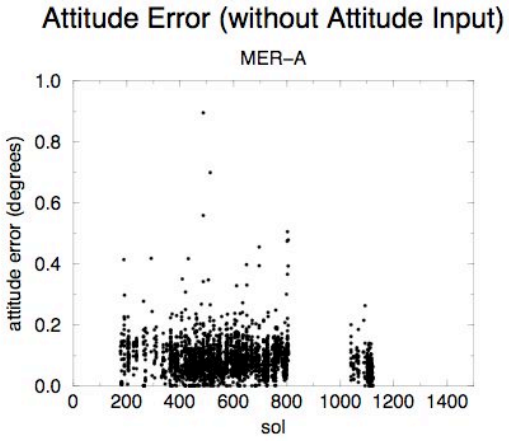

Figure 9. Error between the MSL-VO attitude and MER onboard attitude estimates.

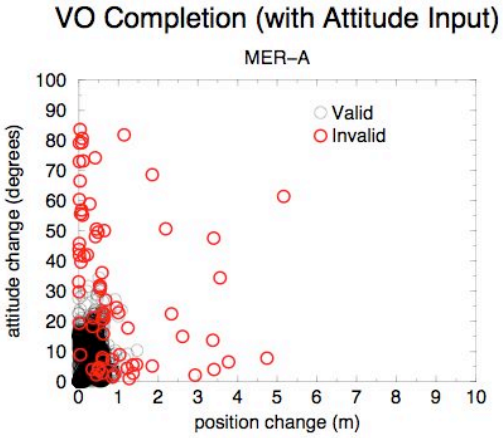

Figure 10. Assessment of valid test cases.

Figure 10 shows that there were 2677 valid motion estimates and 80 invalid motion estimates (97% valid). Most of the invalid test cases were for large motions where VO should not have worked anyway (large change in attitude or position). This performance is as good as or better than the performance reported for MER-VO [6] (while using a much more computationally efficient algorithm).

No initial motion estimate is used to generate these results. The quality of the results indicates that MSL-VO can be used without a motion estimate and still generate robust motion estimates. Re-running the test cases with an attitude estimate to constrain the correlation search shows very similar estimation performance, but with a four times faster computation time.

C. Motion Accuracy for Long Traverses

To test the motion estimation accuracy for long traverses, a data set collected by the DARPA LAGR vehicle on a dry lake bed in the Mojave Desert was used. The camera configuration was similar to the MSL navcam configuration

(2m height, 30cm baseline, 45˚ FOV). The traverse was 1375 meters long over flat ground, and the images contained vehicle tracks, repetitive terrain features and at times the shadow of the sensor mast (c.f., Figure 11). Differential GPS provided ground truth.

The initial camera attitude was not known in the GPS frame, so the MSL-VO trajectory was aligned to the GPS trajectory using a Least Squares alignment of the 3D points in each trajectory. The aligned trajectories are shown in Figure 11. Since this global alignment can remove some errors the following procedure was used to assess motion estimation accuracy for 100m traverses. Starting at the beginning of the trajectory, 100 corresponding points between the GPS trajectory and the MSL-VO trajectory were used to generate the trajectory alignment. The GPS trajectory is then traversed until 100m distance. At this point the GPS position and MSL-VO position are compared. This process is repeated for every MSL-VO position in the trajectory (walking along the trajectory) and the statistics shown in Table 2 were generated. Worst case error of 6.39% of distance traveled is better than the MSL performance requirement.

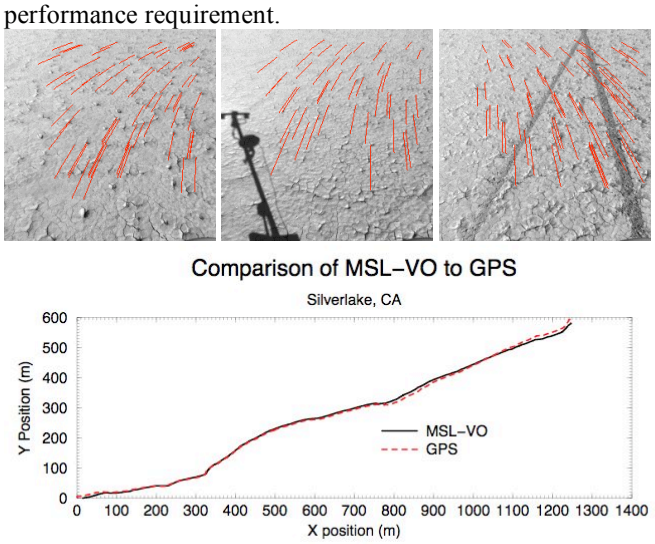

Figure 11. 1km+ traverse images and trajectory comparison to GPS.

TABLE 2: POSITION ERROR STATISTICS FOR LONG TRAVERSE

Traverse Distance	# of Cases	Position Error		
		Mean	Std. Dev	Mean + 3 Std. Dev.
100m	2500	2.91m	1.16m	6.39m

D. *Effect of Rover Step Size*

The final analysis investigates the effect of rover step size on MSL-VO performance. The same DARPA LAGR vehicle with MSL navcam configration was used to collect a 275m image sequence with 50cm steps between image captures. By skipping stereo pairs the effect of 1.0, 1.5 and 2.0 m step sizes were investigated. GPS was not available for this image sequence, so the MSL-VO estimates for the 50cm steps were uses as a truth measurement. The left plot in Figure 12 shows that the number of features drops with increasing step size. The right plot shows position estimation error for 1.0 and 1.5m steps remains small, but the variance of the error becomes much larger for 2.0m step sizes. This analysis indicates that a maximum 1.5m step size should be used for visual odometry on MSL.

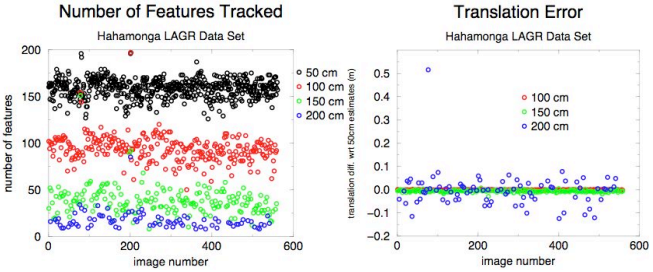

Figure 12. Analysis of step size on performance.

IV. CONCLUSION

The update to the MER visual odometry capability has created an algorithm that is at least 4x faster while tracking significantly more features. The updated algorithm can also be operated when no motion estimate is available. The algorithm has been tested with thousands of MER and MSL-like images over a variety of terrain and this testing strongly indicates that MSL VO will meet all MSL mission requirements.

The algorithm has been coded in C++ and delivered for MSL flight software implementation. The code has been ported to the Rocky 8, FIDO and ATHLETE real-time robotic platforms where it has been shown to perform more efficiently and robustly than the MER-VO algorithm.

ACKNOWLEDGMENT

The authors would like to thank Andrew Howard and Mark Maimone for providing image sequences, DARPA for provided the LAGR vehicle and Mike McHenry and Todd Litwin for helping with the real-time implementation.

REFERENCES

[1] M. Agrawal, K. Konolige, R. Bolles, Localization and Mapping for Autonomous Navigation in Outdoor Terrains: A Stereo Vision Approach", *Proc. IEEE Workskop on Apps. of Computer Vision,* 2007.

[2] P. Corke, D. Strelow and S. Singh, "Omnidirectional Visual Odometry for a Planetary Rover," *Proc. IEEE/RSJ Int'l Conf. Robotics and Systems*, 2004.

[3] C. Harris and M. Stevens, "A Combined Corner and Edge Detector," *Proc. 4th Alvey Vision Conf.*, pp. 147-151, 1988.

[4] H. Hirschmüller, P. Innocent, and J. Garibaldi, "Fast Unconstrained Camera Motion Estimation from Stereo without Tracking and Robust Statistics," Int'l Conf. Control, Automation Robotics & Vision, 2002.

[5] J. P. Lewis, ``Fast Template Matching", *Vision Interface*, 1995.

[6] M. Maimone, Y. Cheng and L. Matthies, "Two Years of Visual Odometry on the Mars Exploration Rovers," *Jour. Field Robotics: Special Issue on Space Robotics* **24**(3), pp. 169-186, March 2007

[7] A. Mallet, S. LaCroix, L. Gallo, "Position Estimation in Outdoor Environments using Pixel Tracking and Stereovision," *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA00)*, 2000.

[8] L. Matthies, *Dynamic Stereo Vision*, PhD Thesis, Dept. of Computer Science, Carnegie Mellon University, CMU-CS-89-195, 1989.

[9] H. Moravec, *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot*, PhD Thesis, Stanford University, 1980.

[10] D. Nister, O. Naroditsky and J. Bergen, "Visual Odometry," *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2004.

[11] C.F. Olson, L. H. Matthies, M. Schoppers and M.W. Maimone, "Robust Stereo Ego-motion for Long Distance Navigation," *Proc. IEEE Conf. Computer Vision Pattern Recognition* (CVPR00), 2000.