EVALUATION OF TRACKING CONFIDENCE INDICATORS AND FEATURE EXTRACTORS ON A VISUAL TRACKING ALGORITHM

By

PAVAN KAYATHI

Bachelor of Science in Computer Science and Engineering

Jawaharlal Nehru Technological University

Hyderabad, Andhra Pradesh, INDIA

2004

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 2008

EVALUATION OF TRACKING CONFIDENCE INDICATORS AND FEATURE EXTRACTORS ON A VISUAL TRACKING ALGORITHM

Thesis Approved:

Dr. Douglas Heisterkamp

Thesis Adviser

Dr. Jay Hanan

Dr. Venkatesh Sarangan

Dr. A. Gordon Emslie

Dean of the Graduate College

.

ACKNOWLEDGMENTS

I would like to thank Dr. Jay Hanan and the department of MAE for providing me continuous support over the last two years for carrying out my work. Dr. Hanan's ideas and suggestions have been invaluable to me in carrying out my research. I would like to thank Dr. Douglas Heisterkamp for providing me guidance and valuable insights into my thesis work. I would also like to Dr. Venkatesh Sarangan for his time and support. I would also like to Sarah Cary for her support. I would also like to thank Pietro Perona at the California Institute of Technology and Tien-Hsin Chao of the Jet Propulsion Laboratory for their support. Casey L. Hughlett of Zion Labs, Inc. was instrumental in providing software development support. Chris Assad of JPL was instrumental in many areas of the project. I would like to thank him for his vision, resources, and assistance with the Mars Yard runs. Ambrus Csaszar of Caltech updated the low-cost rover hardware and aided in collecting data for the experiments. Pierre Moreels also provided valuable insight. I would also like to thank the SPIE for providing me a chance to present my work at the SPIE Defense and Security Symposium. A portion of the research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration and funded through the Director's Research and Development Fund program.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

LIST OF GRAPHS

CHAPTER 1

INTRODUCTION

The National Aeronautics and Space Administration (NASA) twin robot geologists, the Mars Exploration Rovers (MERs), were launched toward Mars in 2003 in search of answers about the history of water on Mars [1]. The Mars Exploration Rover mission is part of NASA's Mars Exploration Program. Until recently, autonomous driving had not been attempted for planetary exploration [2]. Even now, the present MER rovers on Mars are limited by data transfer, power, and processing constraints. The MER rover systems, Spirit and Opportunity, principally rely on dead-reckoning. Position estimation is calculated from onboard sensors such as wheel encoders and Inertial Measurement Units (IMUs). Changes in the environment are unseen and errors in the traverse, for example, from slipping go undetected. A dedicated vision based autonomous guidance system would provide the capability to drive with a real-time response to the terrain and environment.

Planning a rover traverse involves quickly assessing many terrain properties and developing a command sequence that strikes a balance between vehicle safety and traverse efficiency [3]. Rover operators have a number of on board drive types at their disposal, including directed drives, guarded drives in which the rover can image terrain and veto a predetermined motion, and rovers can also use image-based odometry to

accurately measure the rover's position while driving in high-slip environments. In an ideal world, the rovers would use visual odometry and hazard avoidance at all times, but this is impractical due to the slow processing speed of the rover's CPU. Thus, blind drives are preferred when a hazard-free path can be seen in images. On level ground, the rover slips very little and the combination of inertial measurements and wheel odometry enables precise blind drives. A very steep slope or a moderate slope combined with rocks or significant sinkage can cause the rover to tip over. Moderate slopes can cause high slip depending on the underlying material, and loose material on low slopes can cause the wheels to sink and can block direct uphill progress.

Correcting error in position estimation is often critical to save time in reaching a target destination and prevent damage to a rover or lander on hazardous terrain. High accuracy, thus allows selection of higher science return destinations. Image based motion estimation such as stereo methods [4, 5] or pose estimation [6] are under investigation for several autonomous robotics applications. Such algorithms depend on image information based on edges and corners (such as grayscale feature extractors [7]) detected in scene elements. Some grayscale feature extractors have also shown sensitivity to the identification of hazardous terrain.

Visual Odometry, an algorithm currently in use on MER uses successive snapshots taken at half meter intervals for estimating the position of the rover relative to known obstacles. While this system has been used for obstacle avoidance [8] and instrument placement [9], in between these snapshots the rover still has a "blind" drive, and there have been incidents where a MER rover has slipped dangerously close to an obstacle, even using this system.

A neural network has the characteristics of emulating the behavior of a biological nervous system and draws upon the analogies of adaptive biological learning [10]. Because of their adaptive pattern classification paradigm, they posses the significant advantages of complex learning capabilities plus a very high fault tolerance. They are not only highly resistant to aberrations in input data, but they are also good at solving problems too complex for conventional technologies such as rule based and determinative algorithm methods.

For visual tracking, a radial basis function neural network algorithm was used [11], [12]. Coupled with a feature extraction algorithm, the neural network has advantages for pattern recognition, including practical implementation in parallel hardware for real-time operation and low power requirements [14], [15]. Upon selection, a target feature is broken down into a set of values that represent a vector in a feature space. This feature space position and associated radius encompasses a neuron with the receptive field representing a local region of feature space around the prototype where generalization is possible. The recognition task then consists of evaluating if a new input feature vector lies within the influence field of any neuron stored in the network. Fast learning in the network model lends itself to real-time adaptation to changing target features when objects are viewed from varying angles and the imagery changes scale with distance.

A radial basis neural network algorithm in combination with feature extractors for visual tracking has been looked at. An analysis of the various feature extractors along with tracking confidence indicators has been provided. The second section contains the literature review. The third section contains the problem statement followed by the

hypothesis. It is followed by the approach to the problem and then the results and discussions and finally the summary is presented.

CHAPTER 2

LITERATURE REVIEW

2.1 Visual Odometry [16]

The MER Visual Odometry system has onboard software for comparing stereo pairs which are taken by the pointable mast-mounted 45 degree FOV Navigation Stereo Cameras (NAVCAMs). The system computes an update to the 6-DOF rover pose (x, y, z, roll, pitch, and yaw) by tracking the motion of autonomously selected interesting terrain features between two pairs of stereo images, in both 2D pixel coordinates and 3D world coordinates. In the early stages, the uncertainty in the amount of slip resulting from drives on high slopes or loose soils forced the operation team to spend several days driving towards some targets, even those that were just a few meters away. The use of Visual Odometry software has enabled precision drives over distances as long as 8 meters, on slopes greater than 20 degrees. The key idea of the Visual Odometry method is to determine the change in position and attitude of two or more pairs of stereo images using maximum likelihood estimation. The basic steps of this method are described as follows.

1. Feature Detection: Features that can be easily matched between stereo pairs and tracked across a single motion step are selected. An interest operator is applied to one of the image pairs, and pixels with highest interest values are selected.

2. Feature-based Stereo Matching: Each selected feature's 3D position is computed by stereo matching. The 3D positions of these selected features are determined by intersecting rays projected through the camera models. After the rover moves a certain distance, a second pair of stereo images is acquired. The features selected from the previous images can be projected into the second pair using the prior knowledge of the motion provided by the onboard wheel odometry. A correlation based search reestablishes the 2D positions precisely in the second image pair. Stereo matching is then performed on these tracked features on the second pair to determine their new 3D positions.
3. Robust Motion Estimation: The motion estimation is done in two steps. First, a less accurate motion is estimated by Least-squares estimation. The features selected are then estimated using the maximum likelihood motion estimation.

Several benefits were realized by the use of Visual Odometry. The accuracy of driving in new or mixed soil terrains was improved by re-pointing to the drive goal or recomputing the distance remaining to the goal after each step along the way. Sometimes rover safety was maintained by having the rover terminate a planned drive early when it realized that it was making insufficient progress towards its goal.

2.2 Radial Basis Function Neural Network

The radial basis function neural network learns 'by example' from data samples and corresponding categories. A block diagram of this system is shown in 1. Feature extraction methods are tailored for a specific class of targets. These feature vectors,

representing the target of interest, are then fed into the neural network processor. A gradient feature extractor is suited for detecting the edges or texture of an object:

$$F(x,y) = \sum_x \frac{\partial I(x,y)}{\partial y} + \sum_y \frac{\partial I(x,y)}{\partial x},$$

where $I(x, y)$ is the input pixel value at (x, y).

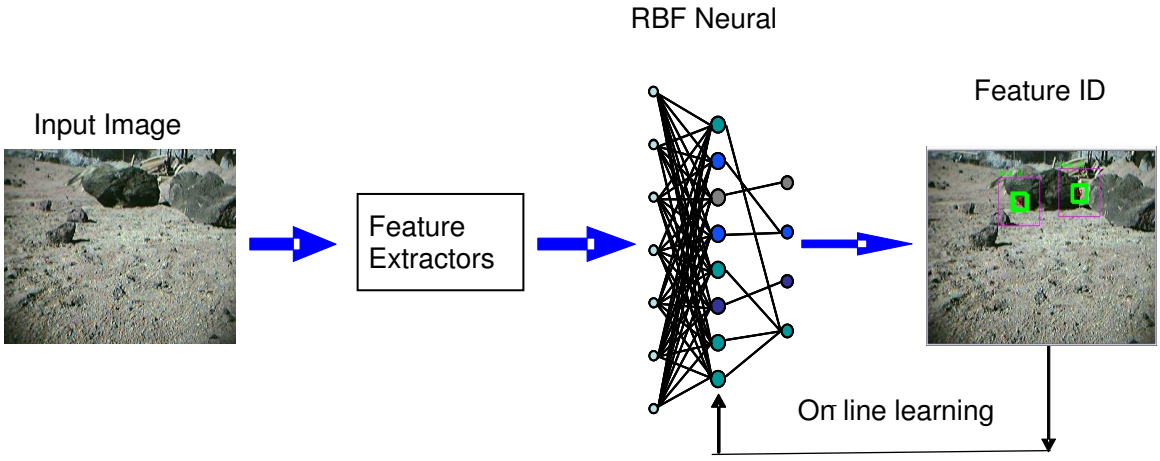


Figure 1: Block diagram of a neural network, showing simplification of a science target using feature extractors [5].

A distance in feature space is calculated from each neuron to the input feature:

$D(N) = |W_n(x, y) - F(x, y)|$, $W_n(x, y)$ is the N^{th} neuron weight vector.

If the input feature vector falls within the influence field of a neuron, then it is classified as belonging to this neuron class. The influence field of a neuron is a multi-dimensional hyper-spherical volume centered at the neuron. The radial distance between the center and the edge of the influence field is defined by the shortest distance between the neuron and the counter examples of a different class or clutter. Each neuron adjusts its influence fields when presented with examples of objects and clutter so that it

occupies a volume around the prototype example up to the position of the counter examples. The influence field is defined as,

$Inf(N) = min(D_{ce}(N))$, $D_{ce}(N)$ - distance between counter-examples and the N^{th} neuron.

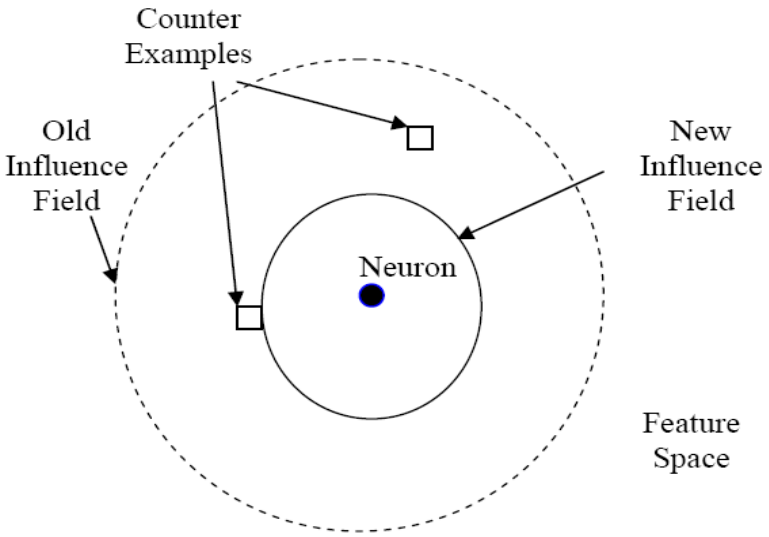


Figure 2: Influence field of a neuron and counter examples. [22]

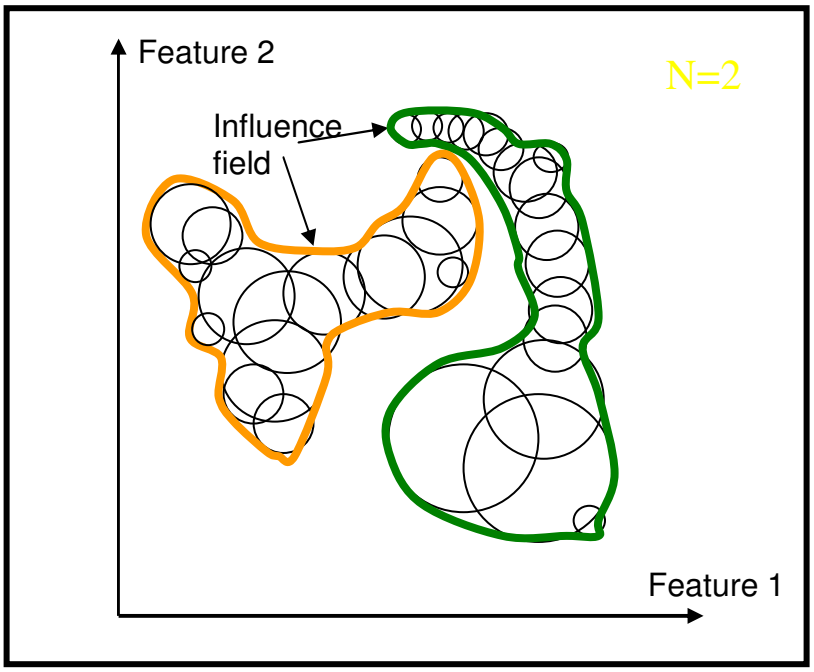


Figure 3: An example of RBFNN in a simplified 2-D feature space.

Upon selection, a target feature is broken down into a set of values that represent a vector in the feature space. For example, the lightness values of all the pixels in each row and column could be summed to obtain a vector in feature space; let this be called as Neuron A. If the feature represented by Neuron A is 10 pixels wide by 8 pixels tall, then it describes 10+8=18 dimensional vector in feature space. During tracking, frames are streamed continuously and tracked features are searched in each new frame. When a new frame is received, the tracker looks at every 10 pixel by 8 pixel box within a set distance (defined by a region of search parameter) from the last known location of the feature for Neuron A, and breaks each of these features down into a vector in feature space. If one or more of these vectors is within a set distance from Neuron A in feature space, then Neuron A will "fire" indicating that the feature has once again been detected.

2.3 Terrain Map Generation [9]

The inputs into the hazard detection and avoidance algorithm are two overlapping images of the surface and a measurement of the distance between the camera and the surface along the camera optical axis of the first image. The details of each stage of the algorithm are given below.

A. Initial feature selection and tracking

The first stage in the algorithm finds locations in the first image that will be good for tracking and then searches for their corresponding location in the second image using image correlation. Feature selection is done using the efficient implementation of the Shi, Tomasi and Kanade feature detector described in [17]. First image gradients $I_r(r, c)$, $I_c(r, c)$ are computed using finite differences over the entire first image. Next the autocorrelation matrix $A(r, c)$ for a small window T around each pixel is computed.

For efficiency, the elements of A are computed using a sliding sum; each time the template is shifted by a pixel, the gradients that leave the template are subtracted from the sum and the gradients that appear in the window are added. Pixels are better for tracking when A has two large Eigen values.

Motion estimation is more likely to be well conditioned if the selected features are evenly spread over the image. To enforce an even distribution, the image is broken into blocks of pixels and the feature that meets the conditions and maximizes P over the block is selected as the best pixel in the block. Once features are selected they are tracked into the second image using a 2D correlation-based feature tracker. No knowledge of the

motion between frames is assumed, so the correlation window is typically square and large enough to handle all expected feature displacements

Correlation is applied in a coarse to fine fashion as follows. First, block averaging is used to construct an image pyramid for both images. The template half-width and window half-width at each level are scaled depending on the level in the pyramid.

Feature tracking starts at the coarsest level of the pyramid with a template and a window size scaled to match the coarse resolution. The pixel of highest correlation is used to seed the correlation at the next finer level [18]. After the coarse level, the template size increases as the pyramid level increases while window size is fixed. At the finest scale, the original image data is correlated, albeit with a small window size, and the feature track is accepted if the correlation value is higher than a threshold. Sub-pixel tracking is obtained by fitting a biquadratic to the correlation peak and selecting the track location as the peak of the biquadratic.

B. Structure from motion

The next stage in the algorithm is a structure from motion estimation that uses feature tracks to solve for the change in position and attitude of the camera between the images and the depth to the selected features in the first image. This stage uses a robust nonlinear least squares optimization [19] that minimizes the distance between feature pixels by projecting the features from the first image into the second image based on the current estimate of the scene structure and camera motion. In this approach the motion between two camera views is described by a rigid transformation *(R, t)* where the rotation *R*, represented as a unit quaternion *q*, encodes the rotation between views and *t* encodes the translation between views. The altimetry measurement is used to set the initial depths

to the features in the scene. This altimetry augmentation from motion algorithm eliminates the scene scale ambiguity present in structure from motion algorithms based solely on camera images. The output of this stage of the algorithm is the 6 DOF motion between images and the depth to the features selected in the first image.

C. Dense structure recovery

The final stage of the algorithm uses the motion between images and the coarse structure provided by the depths to the feature tracks to efficiently generate a dense terrain map. For a pinhole camera, the projection of a pixel in the first image must lie on a line in the second image that is determined by the motion between images (the epipolar line). The depth to the pixel determines the location of the pixel along the line. To generate a dense set of scene depths, a grid of pixels are selected in the first image. Next, the epipolar segment is determined for each pixel in the grid. Next the matching location of pixel along the epipolar segment is determined. Once the grid of feature tracks is established, triangulation is applied to establish the depth to each feature. Next, the homogenous coordinates of each feature are scaled by the correspond depths to produce a cloud of 3D points in the coordinate frame of the first image.

2.4 Rover-Based Visual Target Tracking [20]

When no visual target tracking is used, the target positioning error after 10-m travel to the target is in order of 20 cm at minimum due to the stereo range error and the rover pose estimation error [21]. With visual target tracking, the target positioning error can be within a few cm. The rover moves in short steps and the designated feature is

tracked at these discrete intervals. Due to limited computational resources available on-board flight rovers, it is not possible to use continuous or frame rate target tracking techniques. The 2D image feature tracker tracks the target image at each step as the rover approaches the target in small steps. The various kinds of trackers are as follows:

A. Affine Matching

The affine tracker creates a template window image of a given size. As the rover moves a short distance and takes a new image from its mast-mounted camera, the tracker determines the new target image position in the new image using a Newton-Raphson style iterative method to minimize the difference between the template image and the new target image. The tracker supports three kinds of transforms for matching the template image to the subsequent image: pure translation, scale, and affine transforms. The pure translation matching uses two parameters, t_x and t_y, that shifts the feature in x and y coordinates to find the matching location of the feature in the new image. The scale transform uses a scale parameter in addition to the two translation parameters. The scale parameter resizes the feature template image window for matching. The affine transform uses 6 parameters to match: 4 deformation parameters in addition to 2 translation parameters. The four deformation parameters are the elements of a 2×2 affine deformation matrix that allows scaling, rotation, stretch, and shearing of the 2D planar template image window, which assumes a planar feature. Further, matching is done by pyramidal feature matching. First, feature matching starts with low-resolution coarse matching at the highest pyramid level. Then the tracker uses lower pyramid levels to refine the target location, and finishes with full-resolution fine matching at pyramid level 0. The pyramidal feature matcher uses the same template window size for all pyramid

levels, where increasing the pyramid level by one reduces the image size by half, doubling the effective window size. Two issues were observed by using the affine tracker. The first issue is that the affine matching had a very limited matching range. The second issue is that the affine tracker does iterative search.

B. Normalized Cross-Correlation

After the shortcomings of the affine tracker were discovered a new visual tracker was developed. The iterative tracker was replaced by a normalized cross-correlation matcher (NCC) matcher. The NCC matcher does a brute-force search, increasing the search range virtually to the entire image area.

C. Tracking with Straightforward Rover Motions

Initial results indicated that iterative scale/affine matching was still not as reliable as brute-force NCC, although scale/affine matching helped improve accuracy when it tracked. It was thus desirable to consider an alternate approach to scale/affine matching to take into account the target image size change as the rover gets closer to the target. The mast camera pointing computes the estimated target position relative to the rover after each rover move using point stereo triangulation and a rover pose estimator. Thus, the estimated target distance can be easily computed. Since the target image size in the template window is inversely proportional to the target distance from the camera based on perspective projection imaging geometry, the template image magnification can be made accordingly before applying NCC. So, in this configuration the template image magnification was preceded by NCC, while the iterative scale/affine matcher was removed.

D. Tracking with Sideways Circular Rover Motions

The rover motion during the visual target tracking will not be always straight. It will sometimes be necessary to avoid obstacles by autonomous navigation with hazard avoidance. Therefore, it is necessary to examine the tracking performance with non-straightforward rover motions.

E. Tracking with Turn-in-Place Rover Motions

Since the goal of the autonomous navigator is specified by the goal position only, the heading of the rover is usually not aligned with the target when the rover reaches the goal. Therefore, a turn-in-place rover motion is needed to face the rover towards the target. In order to determine the appropriate step size for the turn-in-place motion, tracking was performed with every 5 degree of turn-in-place motion. As the rover turns, the mast camera turns in the opposite direction to point to the target.

CHAPTER 3

PROBLEM STATEMENT

Methods have been implemented to monitor tracking error and ensure a safe, accurate arrival at the intended science target. The methods are situation independent relying only on the confidence error of the target recognition algorithm. A single calibrated camera has been used for position estimation.

Targets vary in terms of texture, contrast, sharpness of edge, relative speed, and size. Various feature extractors exhibit tradeoffs in terms of sensitivity and processing requirements as related to the characteristics of candidate target classes. An analysis of feature extractors based on the horizontal and vertical profile has been performed and the best feature extractor has been selected for optimal representation in the neural network. The best feature extractor has been selected based on the number of frames traveled for the number of neurons trained.

Feedback from the network can offer an indication of tracking confidence which will be useful in determining if the estimated position is correct. An attempt has been done to look at the various confidence factors to determine if the position estimated is correct.

A key comparison that has been made is the distance traveled computed from vision to the distance traveled computed from the wheel encoders. With this comparison an observation of slipping has been made.

The following hypothesis has been proposed for the evaluation of the confidence factor indicators.

1. The confidence of the network response is related to the number of hits (or neuron firings) of the feature extractor given by the network for a particular target in the frame.

2. When two targets are selected, relating the camera roll with the difference in the vertical positions of the targets can be used to indicate regions where the target is lost. Rover pitch and roll was indicated by an accelerometer.

3. Monitoring the change in the vertical position of the two tracked targets provides additional feedback on tracking confidence. A significant error reveals error in tracking and can be used to pause forward motion.

CHAPTER 4

EXPERIMENTAL METHODOLOGY

The data presented has been collected using an autonomous rover [23]. The four wheeled rover was based on an off-the-shelf platform that had been previously modified at JPL to allow for computerized data collection and remote driving. The present version is equipped with an onboard laptop with wireless link, single forward-facing camera (CMOS 320 x 240 pixels), a horizontal two-axis accelerometer, wheel encoders, and other sensors.

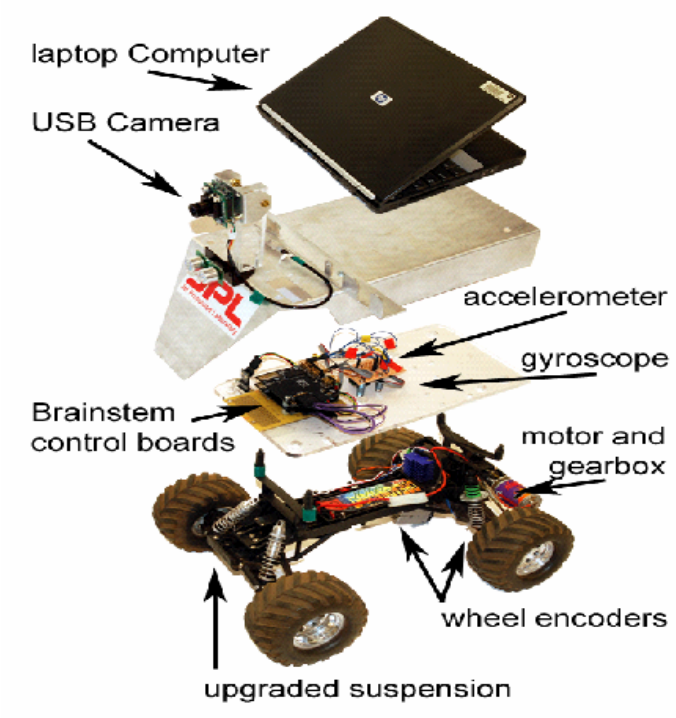


Figure 4: A blow-up view of the vehicle hardware in its current configuration [23].

Figure 5: Assembled configuration with USB webcam and laptop computer [23].

For the JPL Mars yard, which is an environment designed for testing rovers in Mars-like conditions, the natural rocks showed sufficient contrast to be tracked throughout a traverse. When a rock (target) was selected, driving toward that rock was done by simply tracking the rock or a single point on that rock. Optionally, multiple points were selected on a rock or from a group of rocks and driving can be toward a weighted center or other geometrically relevant location. Another advantage of tracking multiple targets is the potential to identify the accuracy from information intrinsic to the scene.

Here, two classifications of data have been collected using the rover. Video recorded while driving manually, and video recorded under autonomous control. Manual driving is useful for initial tests of the visual tracking methods. In addition, manually

collected video allows testing of methods for calculating tracking confidence for the autonomous runs. The manually collected video is often more difficult to process since tracking is not done in-the-loop. Decisions to turn or guide the rover are made by the driver which has external information to the rover video. Tests of the algorithm on this data may not be sufficient to prove the capability for autonomous driving, but it often identifies areas where further improvement is possible. Furthermore, the auxiliary data collected from the wheel encoders, accelerometers, and other sensors provides valuable feedback for comparison to dead reckoning position estimation. Autonomous driving differs from manual in that the steering is controlled by the result of tracking on the frame. Thus the target is less likely to move out of the frame. Autonomous driving for collection of this data set is described further elsewhere [23]. Test conditions included slipping while driving a slope and more simple approaches on rough sandy terrain.

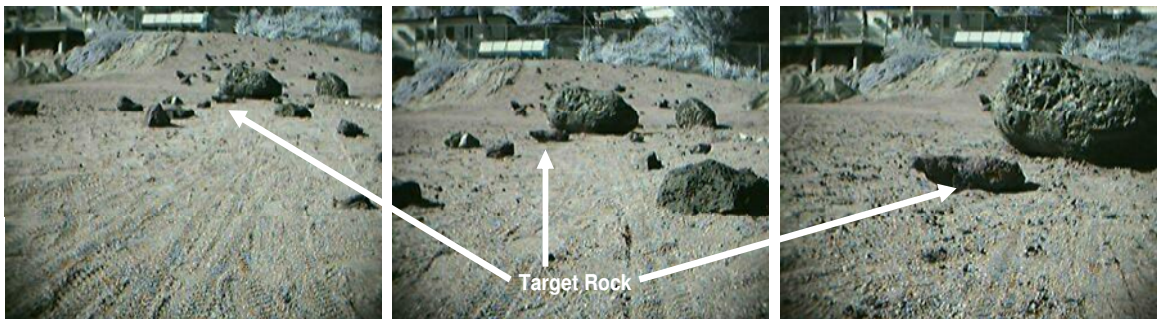
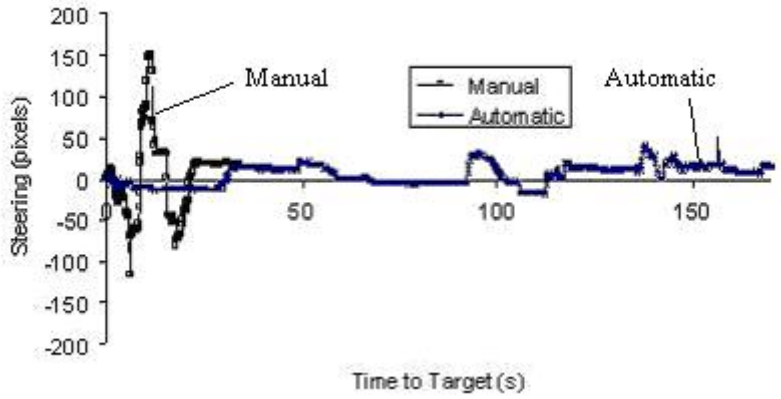


Figure 6: Example video from autonomous driving. The target rock remains in the field of view throughout the sequence.



Graph 1: A typical run from automatic steering is compared to a typical pattern in similar terrain calculated from post processing on frames recorded during manual steering. Automatic steering required lower magnitude corrections. Manual runs were generally shorter than the automatic runs as they were primarily used for testing system configurations. The target position is at 0 seconds.

During live tracking of one or more features, the operator may enable a proportional steering controller which can be used for a vision-guided approach to a waypoint. It attempts to aim the vehicle towards the centroid of the tracked feature set using a first-order approximation for the vehicle's angular deviation from desired course. The steering output angle, α, is given by the proportional relation [23].

$$\alpha_t = k_1 k_2 \sum_{i=0}^{N} \frac{x_{i,t-1}}{N},$$

where $x_{i,t-1}$ is the horizontal displacement of the i-th tracked feature from the center of the most current video frame; k_1 is the proportional gain of the controller; k_2 is the angle subtended by a single pixel in the video camera's field of view. The "true" value of k_2 deviates from the constant near the edges of the video frame, due to higher-order camera lens distortion that can be approximated [24].

One method of generating localization data from visual information is the tracking of two distinct features whose initial location in space relative to the vehicle is known [23]. The MER rovers can acquire such information from a 3-dimensional terrain map generated by image correlation from their stereo mast cameras [25]. Although the vehicle described here has no such capability, it's operated under the premise that high-speed visual distance estimation could be used on hardware with similar still-image stereo cameras to those on MER. Thus, a method is presented for monocular distance estimation given that the initial location of at least two suitable features is known relative to the vehicle, as shown in Figure 7.

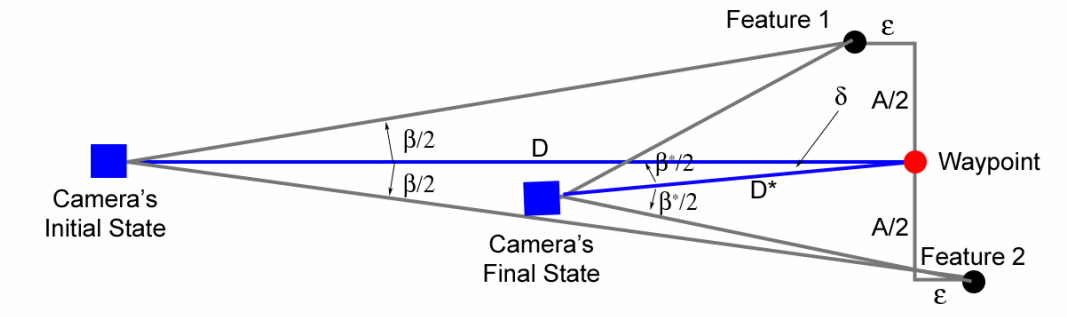


Figure 7: Graphical representation of the geometry involved in estimating D* based on Feature 1 and Feature 2

For calculations, the same first-order approximation for the angle subtended by a single pixel is used. Given the apparent pixel distance, $(x_2–x_1)$ between the two tracked features, the initial angle, β, which separates them, can be found by using [23]:

$$\beta = k_2 (x_2 - x_1)$$

The waypoint of the vehicle is defined by the centroid of the two features. Knowing β and the initial distance, D, along a straight path to the waypoint, the linear distance, A, which separates them can be approximated by

$$A = D[\tan(\beta)]$$

Finally, distance D^* along a straight-line path to the waypoint can be approximated as:

$$D^* = \frac{A}{\tan(\beta^*)}$$

For small β, $\tan(\beta) \sim \beta$ and D^* becomes:

$$\frac{C}{x_2 - x_1}$$

Where, $C = A/K_2$ and scales with target size.

D^* will provide an adequate approximation for the linear distance remaining to the waypoint, with the caveat that $\delta << \beta$ and $\varepsilon << A$ both must hold [23]. Given that the vehicle platform has a forward-facing camera with a field of view less than 30 degrees, and is actively steering towards the waypoint during operation, it is likely that one feature will be lost from the field of view and tracking will fail before this approximation breaks down.

The tracking procedure begins by loading the images needed for tracking. Then an existing tracker is loaded for automatic target recognition, or a new one created by specifying the coordinates in the image to commence training. The tracker has three components: the target, the associated neural network, and the tracking policy. For the target component, the coordinates for the region of interest and the region of search were specified along with the distance for automatic counter examples. For the neural network the configurable values have been automated such that no values must be set. The tracking policy contains fields for automatic learning. For example, the thresholds for hits above and below which the network will automatically learn. The result for the number of hits has been discussed in the results and discussions section. Also, the user

specifies the reaction to a lost target such as increasing the search window, termination, or moving back. After initial set-up, a target was selected for tracking. Targets can be a single region associated with a coordinate, or multiple regions depending on the interest of search. For off-line studies, once the target has been selected, the frames advance. For real-time work, the rover was commanded to go to the target after selection. In case of a traverse to a location which is over the horizon or the case of a lost target, an automatic feature extractor can be used to search the scene for acceptable targets. Each feature that is monitored by the tracker corresponds initially to a single neuron in our neural network. However, as the rover moves towards the selected target(s), the scales and orientations of the tracked features may change significantly. To cope with this, the neural network is capable of learning the new characteristics of its tracked features by adding new neurons. Once the rover is sufficiently close to the original target rock (~ one meter), the approach run is considered a success.

Once tracking is complete an output file is saved containing the frame number, the *x* position, the *y* position, number of hits, and influence field parameters for all the trackers. For example, two point tracking has coordinates such that the *x* difference (dX), *y* difference (dY), and average number of hits. The difference, dX is used to calculate distance from the target.

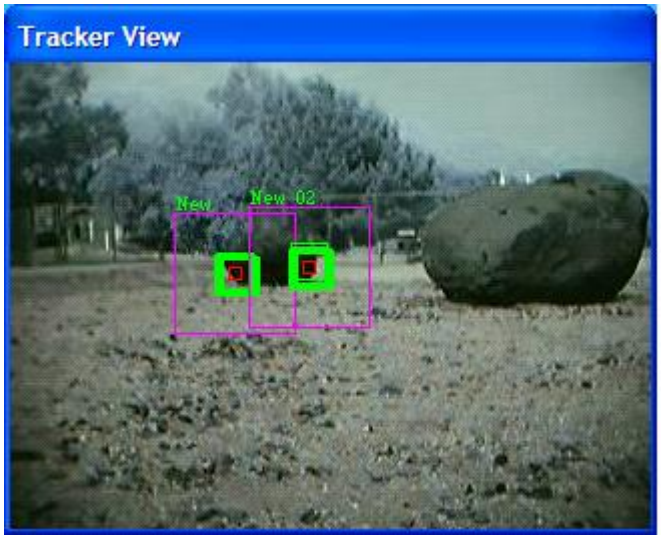


Figure 8: A frame from the sequence tracking the target rock during approach in JPL Mars Yard. The difference between the two rock edges noted by the center of square gives *dX*.

Feature extractors will be considered for the role of supplying additional information to the neural network tracker. The key idea behind feature extractors is simple. We want to store only small collections of data about an image. Homogenous zones contain little information, and more important, any zone from a homogenous region looks exactly the same as any other, giving poor localization. Many extractors look at places where there is a high gradient. Different kinds of feature extractors based on the horizontal and vertical profile have been analyzed. The vertical profile is a relevant feature to characterize patterns oriented vertically where as the horizontal profile is a relevant feature to characterize patterns oriented horizontally.

The different kinds of feature extractors that have been analyzed are named HV, HVMean, and HVMeanGradient. These feature extractors are based on the horizontal (H) and vertical (V) profile. In the HV feature extractor the horizontal and vertical profiles are concatenated and stored in the same neural domain. In the HVMean feature extractor

the horizontal and vertical profiles are range-stretched from 0-100 and their profiles are concatenated and stored in the same neural domain. In the HVMeanGradient feature extractor the horizontal and vertical profiles are first range-stretched from 0-100, then the inter row gradient takes place of the horizontal profile while the inter column gradient takes place of the vertical profile. These horizontal and vertical profiles are then concatenated and stored in the same neural domain. An example of these feature extractors is given in the appendix. Each of the video sequence has been run using these feature extractors. The results and the analysis of the feature extractors on the video sequences have been discussed in the results.

Determining accuracy of the tracking measurements requires truth data for the coordinates of the rover. For flat hard surfaces, the wheel encoders provide reasonable truth. Traverses in the lab were used for this purpose to calibrate the vision system and characterize the tracking confidence. Video external to the rover was also recorded to aid in measuring distances traveled.

A key comparison is the distance traveled computed from vision to the distance traveled computed from the wheel encoders. With this comparison the observation of slipping was most pronounced. Runs on hard surfaces were used to calibrate the encoders and the visual system. Comparisons of these distance measurement values from both the encoders and the video are given in the results.

Feedback from the network offers an indication of tracking confidence useful in determining if the estimated position is correct. Principally, the confidence of the network response is related to the number of hits (or neuron firings) given by the network for a particular target in the frame. This value is reported in the results for comparison to

position error. This parameter has been investigated for error estimation. Lastly, measurements of tracking error can be taken from estimations of the rotation of an object with respect to the roll of the rover. Rover pitch and roll was indicated by an accelerometer.

CHAPTER 5

RESULTS AND DISCUSSIONS

The tests were conducted on four different days on the Mars Yard Runs giving a total of ten runs for performing the offline tests on the feature extractors. The results for each of these Mars Yard Runs on the feature extractors are as follows. For the first run traversed on the July 26th, there were a total of 725 frames for the run. The HV feature extractor was able to successfully traverse till the end of the run. In this run the target was lost once at frame number 111 and it required a total of 6 neurons for this run. The HVMean feature extractor was able to traverse till the end with one training at frame number 111 when the target was lost and it required a total of 3 neurons for this run. The HVMeanGradient lost the target three times and it required a total of 12 neurons to complete the run. For this run HVMean proved a better feature extractor (See table 1).

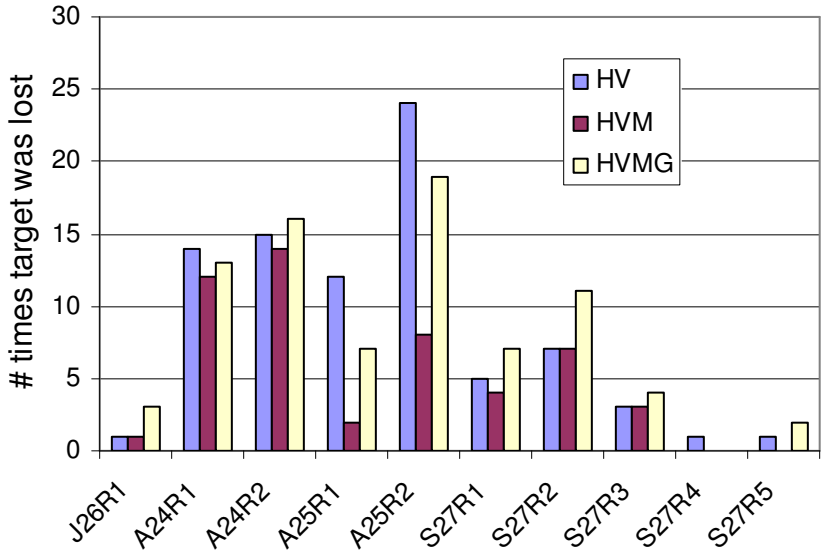
There were two runs traversed on the Aug 24th, a total of 264 frames were covered on this run. For the HV feature extractor the target was lost on a total of 14 frames, the first being on 112. It required a total of 28 neurons to complete the run. The HVMean lost the target on 12 frames, the first being on 112 and it needed 12 neurons to complete the run. For the HVMeanGradient the target was lost on 13 frames the first being on 112 and it needed 16 neurons to complete the run. For this run HVMean proved a better feature extractor. The detail on the second run is given in See table 1.

There were two more runs traversed on the Aug 25th. The first run had a total of 440 frames covered. For the HV feature extractor the target was lost on a total of 12 frames, the first being on 134. It required a total of 80 neurons to complete the run. The HVMean lost the target on only 2 frames, the first being on 166 and it needed 24 neurons to complete the run. For the HVMeanGradient the target was lost on 7 frames the first being on 153 and it needed 53 neurons to complete the run. For this run HVMean proved a much better feature extractor compared to the other two (See table 1).

For the second run on the same day, a similar kind of observation was made. The run had a total of 498 frames. The HV feature extractor lost the target on 24 frames and it needed a total of 90 neurons to complete the run. It lost the target initially on frame 111. The HVMean lost the target on 8 frames and it needed a total of 42 neurons to complete the run. It lost the target initially on frame 152. The HVMeanGradient completed the run by requiring 75 neurons and by losing the target on 19 frames. It lost the target initially on frame 126. The HVMean performed much better compared to the other two.

There were five runs traversed on the September 27th. On the first run a total of 900 frames were covered. For the HV feature extractor the target was lost on a total of 5 frames. It required a total of 52 neurons to complete the run. The HVMean lost the target on 4 frames and it needed 12 neurons to complete the run. For the HVMeanGradient the target was lost on 7 frames and it needed 20 neurons to complete the run. The target was first lost on frame 241 for all the three feature extractors. The details of the rest of the runs are given in table 1.

From all the above runs it was observed that in some cases all the three feature extractors performed similar with HVMean having a slightly better performance than the other two. In other runs it was clearly observed that the HVMean performed much better than the other feature extractors. So it can be concluded that the HVMean feature extractor has a better performance compared to the other feature extractors. The summary of the above runs including the total number of frames covered, the number of times the target was lost, the number of neurons that were trained for reaching the target are summarized in the table given below. The chart below demonstrates the summary given in the table.



Graph: Graph demonstrating the feature extractors on various runs. The Y axis represents the number of times the target was lost. The X axis indicates the day and the name of the run. It can be seen that the HVMean performed better than the other feature extractors.

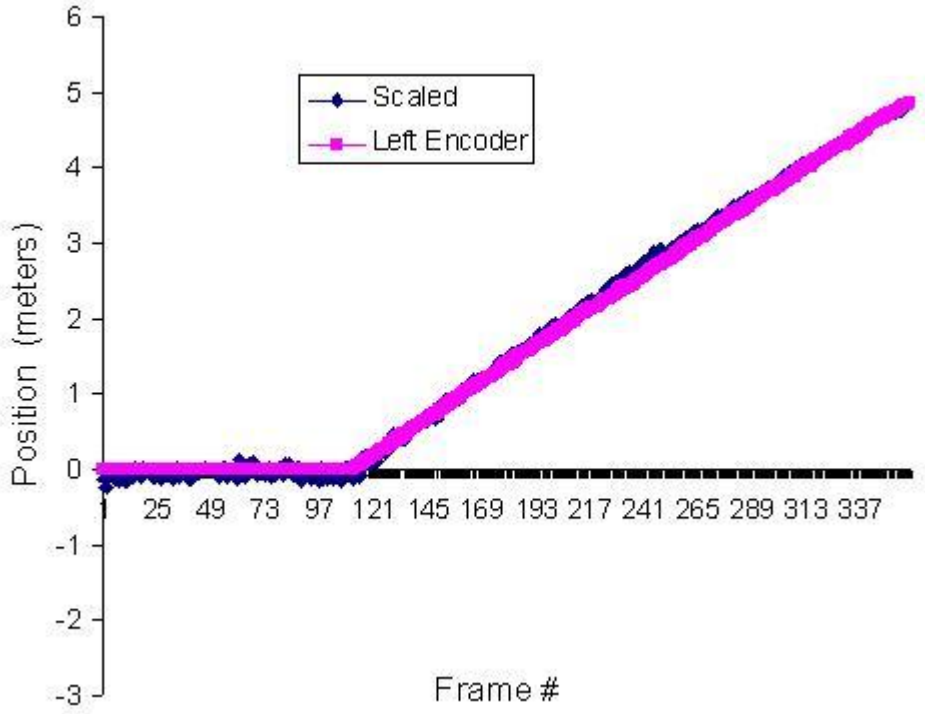
Name of the run	Total number of frames	Feature extractor	# of times target lost	# of neurons trained
		HV	1	6
July 26 Run1	725	HVMean	1	3
		HVMeanGradient	3	12
		HV	14	28
Aug 24 Run1	264	HVMean	12	12
		HVMeanGradient	13	16
		HV	15	33
Aug 24 Run2	368	HVMean	14	22
		HVMeanGradient	16	24
		HV	12	80
Aug 25 Run1	440	HVMean	2	24
		HVMeanGradient	7	53
		HV	24	90
Aug 25 Run2	498	HVMean	8	42
		HVMeanGradient	19	75
		HV	5	52
Sept 27 Run1	900	HVMean	4	12
		HVMeanGradient	7	20
		HV	7	19
Sept 27 Run2	700	HVMean	7	13
		HVMeanGradient	11	16
		HV	3	8
Sept 27 Run3	1000	HVMean	3	5
		HVMeanGradient	4	6
		HV	1	5
Sept 27 Run4	1850	HVMean	0	2
		HVMeanGradient	0	3
		HV	1	90
Sept 27 Run5	1933	HVMean	0	19
		HVMeanGradient	2	74

Table 1: Summary of the feature extractors on Mars Yard Runs.

A key comparison that has been made is the distance traveled computed from vision to the distance traveled computed from the wheel encoders. With this comparison an observation of slipping can be made. The following figure shows the rover approaching a target on the sandy slope surface.

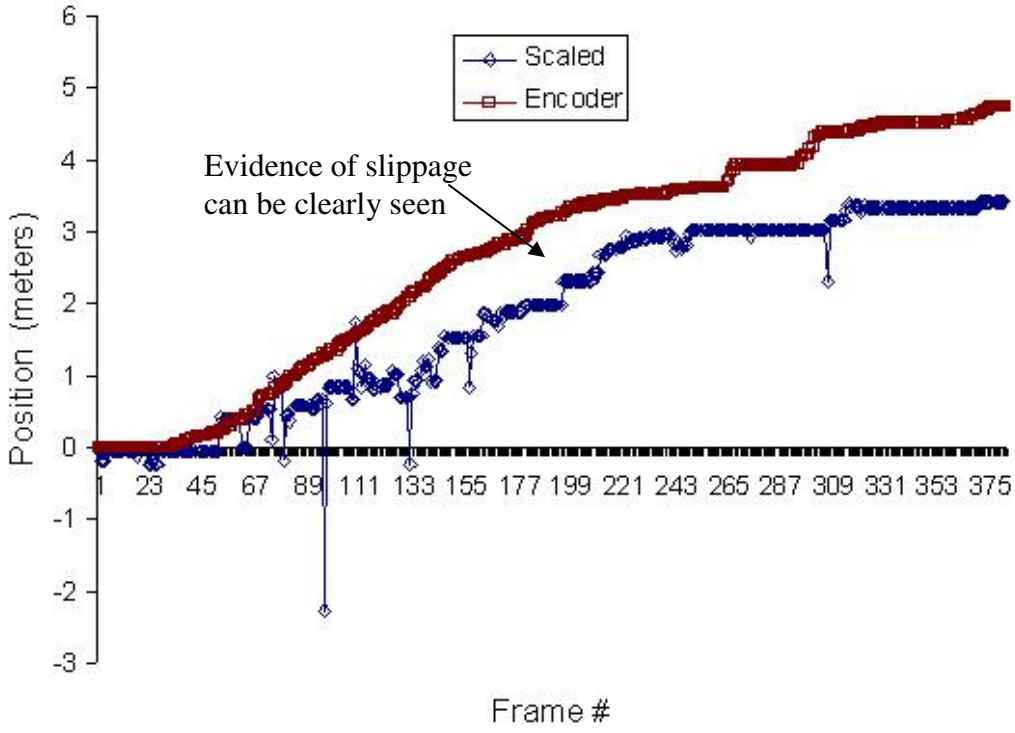
Figure 9: Video frames of rover approaching a waypoint on a sandy surface.

The following graph shows the tracking results from traversing on the hard surface. It can be observed that the encoders compare well with the tracking and there is no slipping during the run.

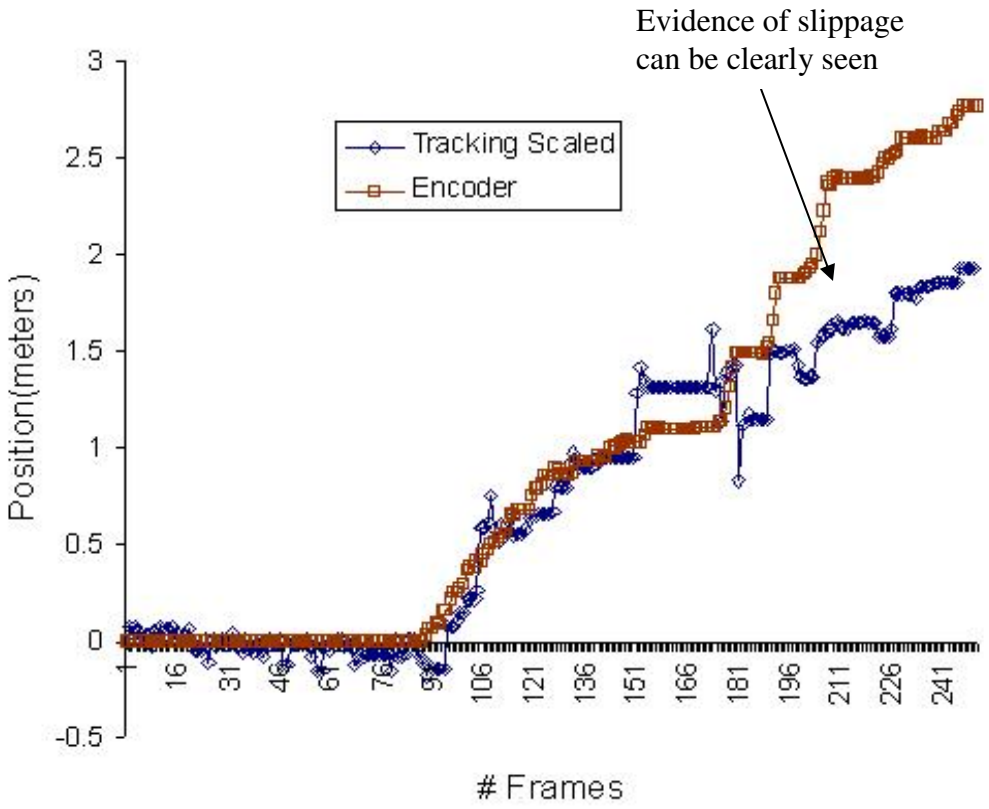


Graph 2: The Graph of tracking results from traversing a hard surface. The encoders compare well with tracking.

The following graphs illustrate the slipping evidence on the Mars Yard Runs which had a sandy sloppy surface.



Graph 3: Comparison of distance traveled by tracking and wheel encoders on the Aug 24 Mars Yard Run 3. The encoders show evidence of slipping contributing to position estimation error.



Graph 4: Comparison of distance traveled by tracking and wheel encoders on the Aug 24 Mars Yard Run1. The encoders show evidence of slipping contributing to position estimation error.

It can be clearly seen from the above graphs that the there is a difference in the distance traveled computed using the wheel encoders and the distance computed from tracking. So the rover can successfully traverse a target even in cases where there is a slope.

Accuracy of an individual tracked point is determined by the strength of the response from the network. A strong response results from multiple neurons recognizing the target. A weak response results from only a few recognitions of the target. Typically strong responses involve more than 10 recognitions and weak responses less than 3. For the range between 3 and 10, new neurons can be automatically generated to increase the strength of the network response. For example, due to the presence of another vehicle

traversing the scene during one of the approaches, a target was temporarily modified and the network response dropped (less hits). This leads to a temporary error in the position estimation of target. The network subsequently recovered the target and the rover was able to continue its traverse to the science target.

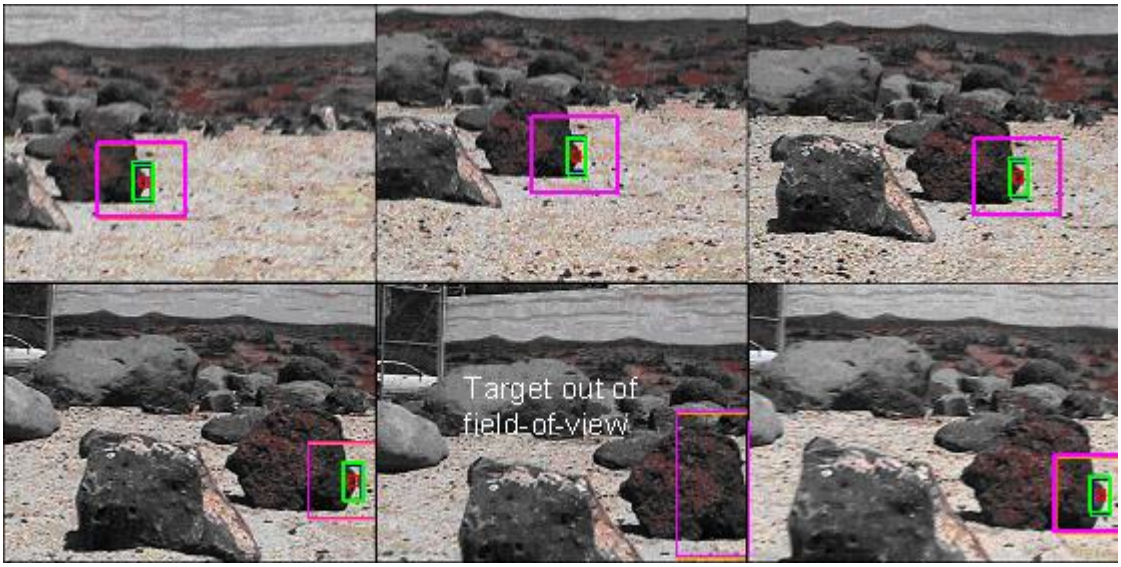
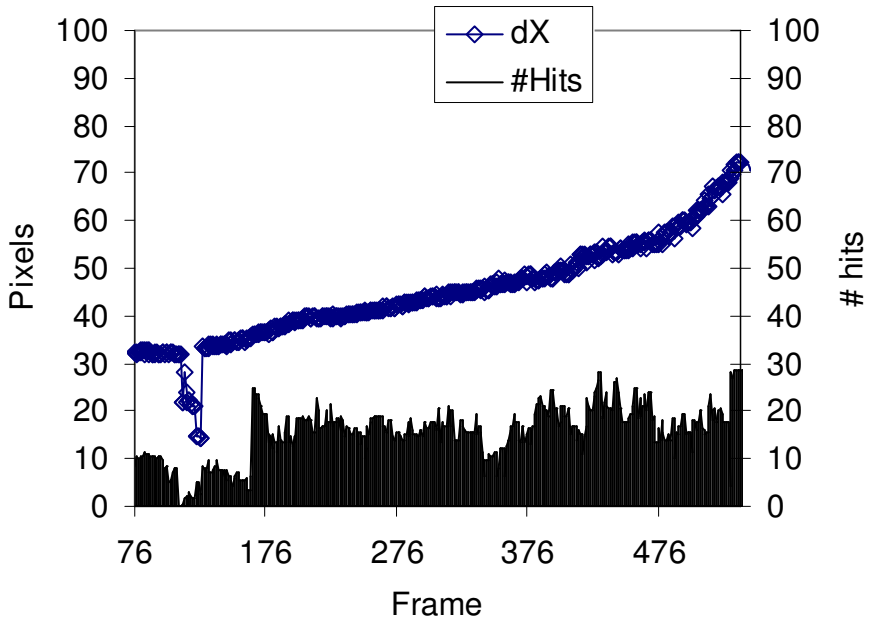
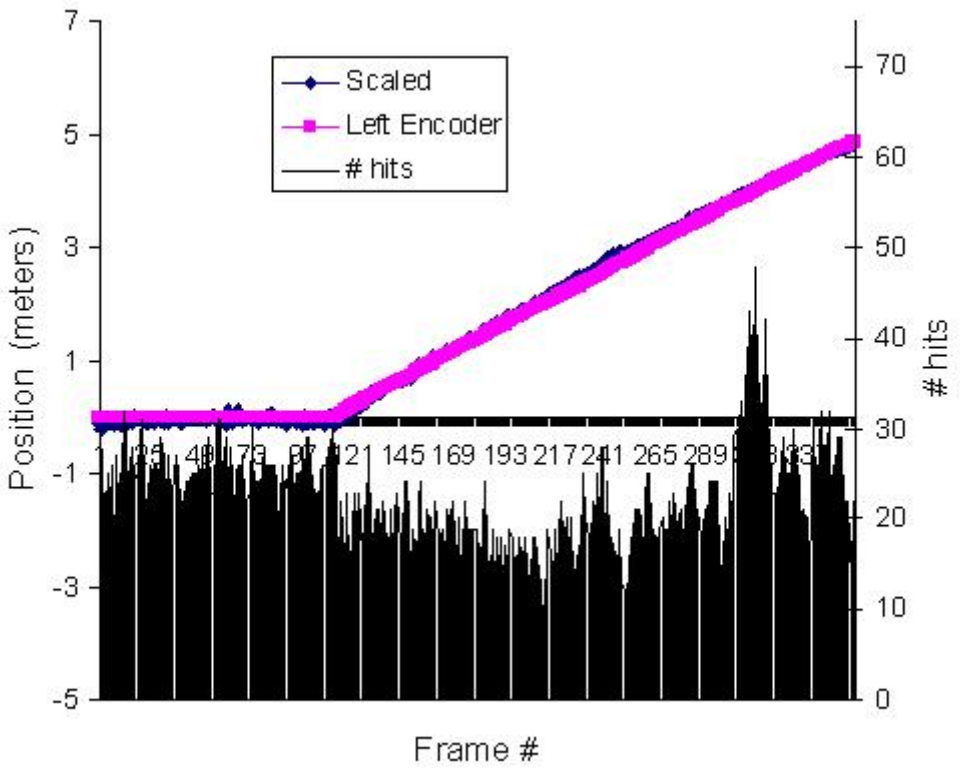


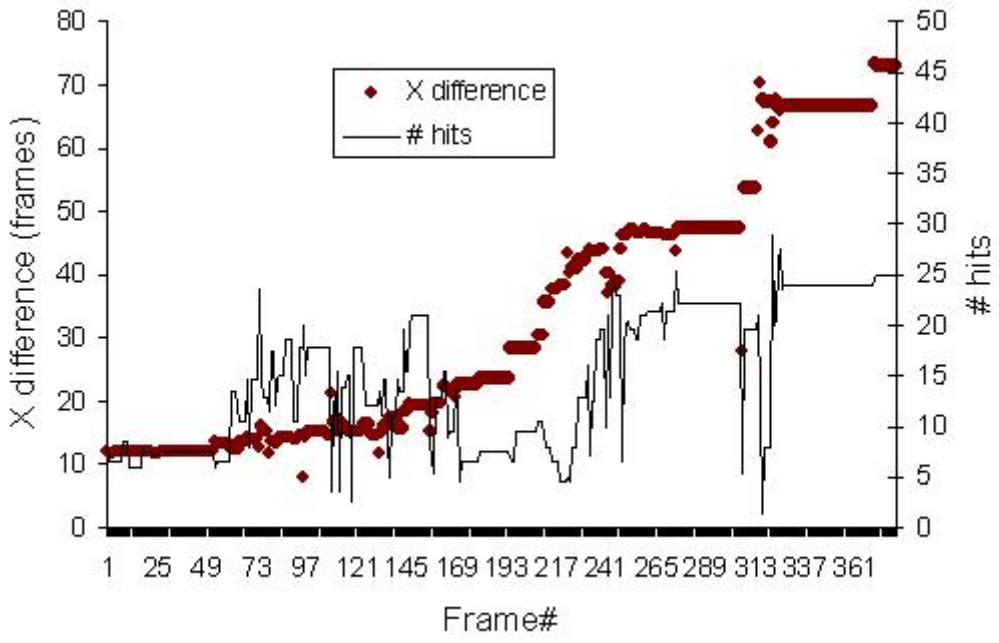
Figure 10: Example of tracking a single point on a rock in the JPL Mars Yard. This data was collected using manual driving. Note how the algorithm looses and regains the target when it temporarily goes out of the field-of-view



Graph 5: Plot of an example traverse showing a drop in network response at the target from a second vehicle in the scene. dX is the horizontal difference between two points tracked on the rock. The #Hits variable shows the network response.



Graph 6: Plot of an example traverse on a hard surface showing the number of hits as the rover reached its target destination.



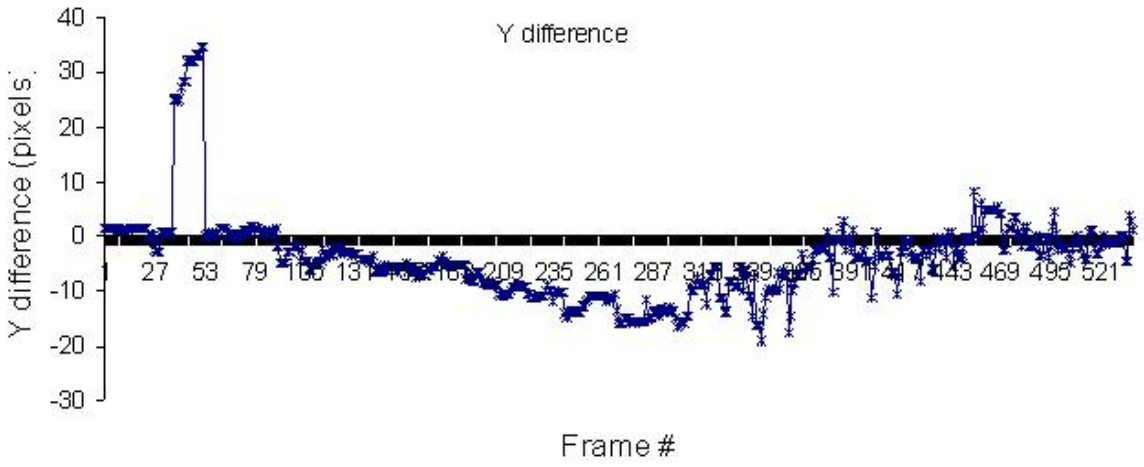
Graph 7: Plot of an example traverse on a Aug 24th Mars Yard Run 3 showing the number of hits as the rover reached its target destination. In the graph one can see that the number of hits on some frames is very less. These points indicate the frames where the target was lost.

There have been some cases where it has been observed that the neuron strength was high on a frame but the target selected was changed to a different target. The following example shows this case in which the target has been changed during tracking.

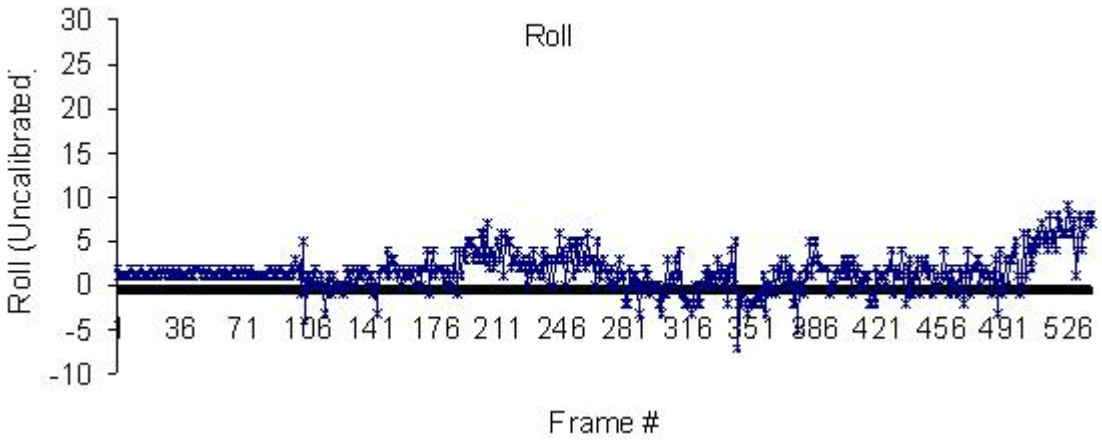
Figure 11: Example of tracking a single point on successive frames on a rock in the JPL Mars Yard. The target changes in the second image but the number of hits still remains high.

It can be concluded that in most of the cases the confidence of the network response is related to the number of hits given by the network for a particular target but there also exists some cases where the target has been changed but the number of hits has been high. So the confidence of the network response is not related to the number of hits (or neuron firings) given by the network for a particular target in the frame.

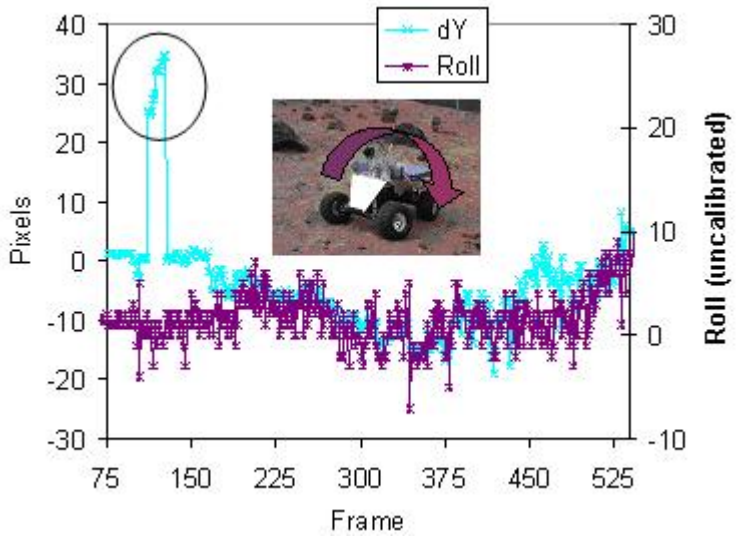
When two targets are selected, for example, on each end of a target rock, their difference in vertical position can be used to relate it with the camera (rover) roll to indicate regions where the target was lost. As shown in the graph 10, the dY value is proportional to the roll of the camera as measured by the accelerometer.



Graph 8: An example of the Y difference for a particular Mars Yard Run.



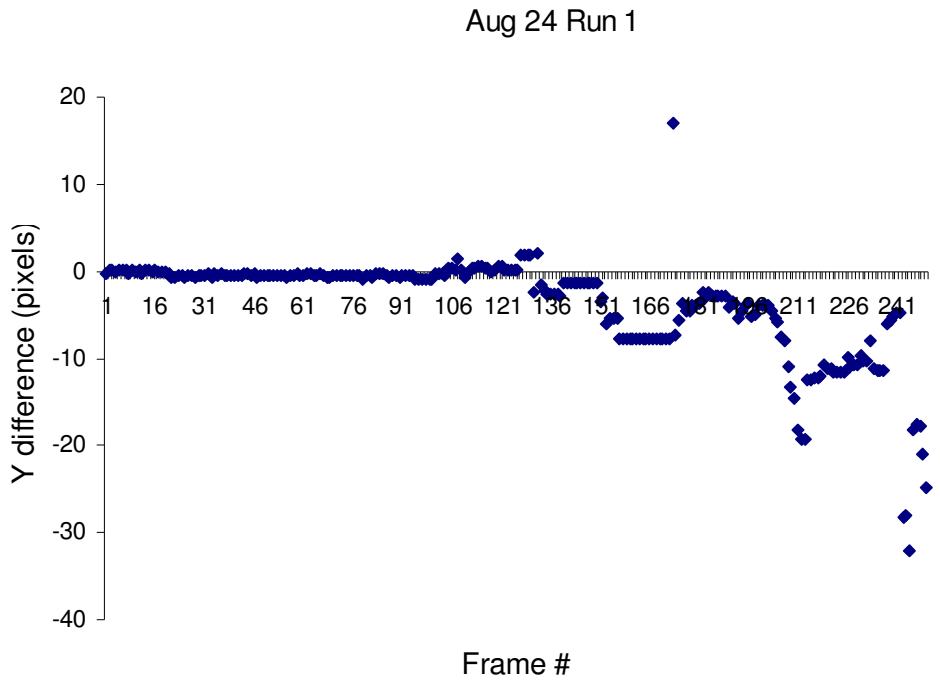
Graph 9: The graph showing the Roll of the rover for the same Mars Yard Run.



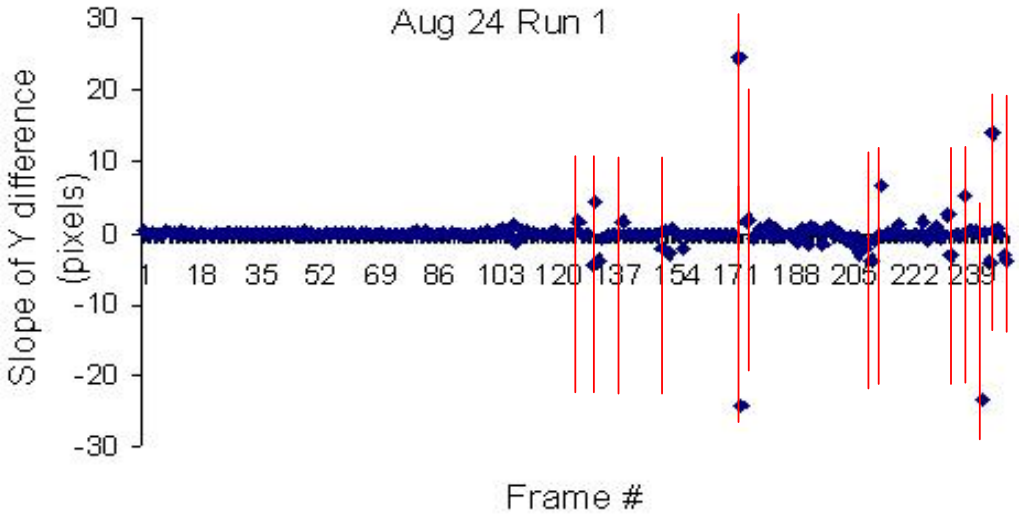
Graph 10: An example of the difference in vertical position of the tracked points correlating with the roll of the rover recorded by the accelerometer. The significant deviation highlighted by the circle indicates low confidence in the tracked locations. Photo in the inset shows the assembled rover in the Mars yard. The roll direction is indicated.

Due to the lack of enough data on the camera roll the hypothesis stating that when two targets are selected, the camera roll can be related to the difference in the vertical positions can not be verified indicating regions where the target is lost can not be verified.

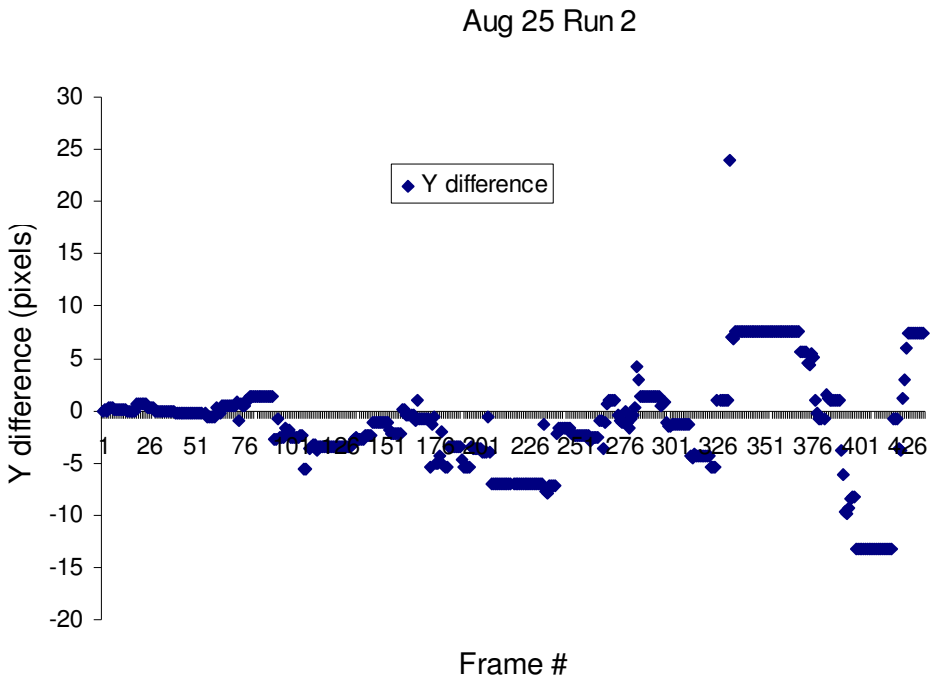
For the rover application, monitoring the change in the vertical position of two tracked targets, dY, provides additional feedback on tracking confidence. A significant error in dY reveals error in tracking and can be used to pause forward motion and a signal to stop learning to move back for reacquisition of the target. By plotting the slope of Y difference one can see the frames where the target was lost.



Graph 11: The graph showing the Y difference for a Aug24 Mars Yard Run 1.

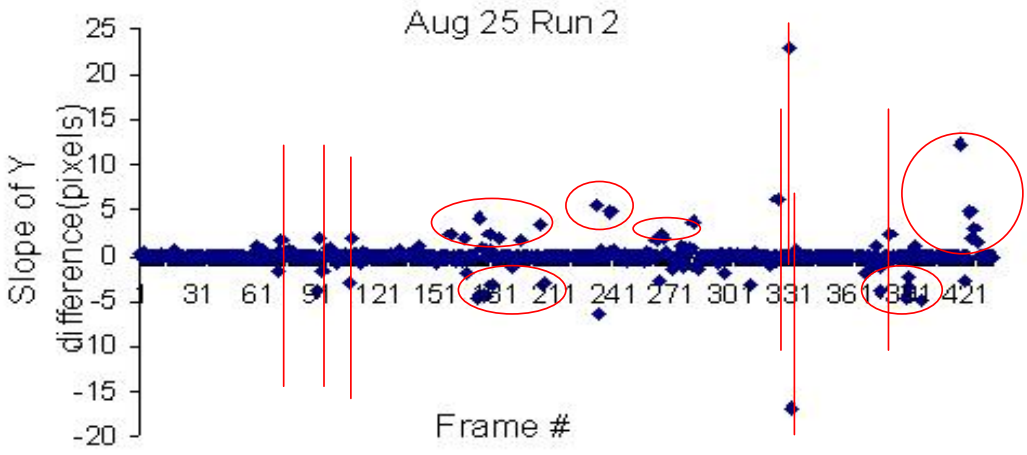


Graph 12: The graph showing the slope in the Y difference for the same Mars Yard Runs. The red lines indicate frames where the target was lost.



Graph 13: The graph showing the Y difference for the Aug 25 Mars Yard Run 2

The graph below shows the graph obtained by plotting the slope in Y difference against the frames.



Graph 14: The graph showing the slope in the Y difference for the same Mars Yard Run given above. The red circles and lines indicate the frames where the target was lost.

From the graphs, it can be observed that whenever there was a change in the target it was identified by a change in the Y difference. Hence it can be concluded that, monitoring the change in the vertical position of the two tracked targets can provide additional feedback on tracking confidence.

CHAPTER 6

SUMMARY

An attempt has been made to look at a radial basis neural network algorithm in combination with feature extractors for visual tracking. An analysis of the various feature extractors has been provided. It has been concluded that the HVMean feature extractor has a better performance compared to the other feature extractors. A comparison of the distance traveled computed from vision to the distance traveled computed from wheel encoders has been provided. With this comparison an observation of slipping has been made. The hypothesis stating that the confidence of the network response is related to the number of hits has been proved to not always be incorrect. The second hypothesis stating that the camera roll can be related to the difference in vertical position indicating regions of lost target could not be verified due to the lack of sufficient data. The final hypothesis which is the change in the vertical position of two tracked targets to provide an additional feedback on tracking confidence was proved correct.

CHAPTER 7

FUTURE WORK

A new rover is in development that offers more stability and control. The algorithms should be optimized for higher frame rates. The position estimation given by the vision system can be improved by replacing the current camera with a higher resolution camera. A further study can be done on the distance estimation accuracy by getting the accurate ground truth values. For this, an option would be including the GPS system. Automatic target and feature selection is envisioned for replacement of features that have gone out of the camera's field of view.

REFERENCES

1) http://marsrovers.nasa.gov/overview/

2) JPL web-site, "NASA Mars Team Teaches Old Rovers New Tricks to Kick Off Year" (2006) <http://www.nasa.gov/centers/jpl/news/mars-20061228.html>

3) P. C. Leger, R. Deen, R. Bonitz, "Remote image analysis for Mars Exploration Rover mobility and manipulation operations", IEEE International Conference on Systems, Man and Cybernetics, Volume 1, pp. 917-922, October 2005.

4) E. Johnson, J. F. Montgomery and L. H. Matthies, "Vision Guided Landing of an Autonomous Helicopter in Hazardous Terrain", *IEEE International Conference on Robotics & Automation*, Barcelona, Spain, pp. 3966-3971, April 2005.

5) C. F. Olson, L. H. Matthies, M. Schoppers, M. W. Maimone, "Rover navigation using stereo ego-motion", *Robotics and Autonomous Systems*, Volume 43, Issue 4, pp. 215-229, June 2003.

6) Y. Ma, S. Soatto, J. Kosecka, S. Sastry, "An Invitation to 3D Vision", *Springer-Verlag*, New York, pp. 109-158, 2004.

7) P. Moreels, P. Perona, "Evaluation of Features Detectors and Descriptors based on 3D objects", *International Journal of Computer Vision*, Volume 73, Issue 3, pp. 263-284, July 2007.

7) I. A. Nesnas, M. Bajracharya, R. Madison, E. Bandari, C. G. Kunz, M. Deans, M. G. Bualat, "Visual Target Tracking for Rover-based Planetary Exploration", *IEEE Aerospace Conference*, Big Sky, Montana, pp. 747-761, March 2004.

8) M. Bajracharya, A. Diaz-Calderon, M. Robinson, M. Powell, "Target Tracking Approach and Camera Handoff for Automated Instrument Placement", *IEE Aerospace Conference*, Big Sky, Montana, pp. 52-59, March 2005.

9) T. T. Lu, C. L. Hughlett, H. Zhou, T. H. Chao, J. C. Hanan, "Neural network post-processing of grayscale optical correlator", *SPIE, Optical Information System III,* Volume 5908, pp. 291-300, September 2005.

10) J. C. Hanan, T. H Chao, C. Assad, C. Hughlett, H. Zhou, T. Lu, "Closed Loop Automatic Target Recognition and Monitoring System", *SPIE, Optical Pattern Recognition XVI*, Volume 5816, pp. 244-251, March 2005.

11) J. C. Hanan, T. H. Chao, P. Moreels, "Neural Network tracking and extension of positive tracking periods", *SPIE, Optical Pattern Recognition XV,* Volume 5347 pp. 233-237, April 2004

12) J. C. Hanan, H. Zhou, T.-H. Chao, "Precision of a radial basis function neural network tracking method," *SPIE, Optical Pattern Recognition XIV,* Volume 5106, pp. 146-153, August 2003.

13) F. Yang, M. Paindavoine, "Implementation of an RBF neural network on embedded systems: Real-time face tracking and identity verification", *Special Issue*

on Hardware Implementations, IEEE Transactions on Neural Networks, Volume 14, pp. 1162-1175, September 2003.

14) Acosta FMA, "Radial basis function and related models – an overview", *Signal Processing*, Volume 45, Issue 1, pp. 37-58, July 1995.

15) Yang Cheng, Mark Maimone, "Visual Odometry on the Mars Exploration Rovers", *IEEE International Conference on Systems, Man and Cybernetics*, Hawaii, USA, pp. 903-910, October 2005.

16) P. Perona and A. Benedetti, "Real-Time 2-D Feature Detection on a Reconfigurable Computer", *IEEE Conference on Computer Vision and Pattern Recognition*, Santa Barbara, CA, pp. 586-593, June 1998

17) P. Garcia-Padro, G. Sukhatme, and J. Montgomery, "Toward Vision-Based Safe Landing for an Autonomous Helicopter", *Robotics and Autonomous Systems*, Volume 38, Issue 1, pp. 19-29, January 2002.

18) S. Roumeliotis, A. Johnson and J. Montgomery. "Augmenting Inertial Navigation with Image-based Motion Estimation", *International Conference on Robotics and Automation (ICRA '02)*, Washington DC., pp. 4326-33, 2002.

19) Kim. W. S, Steele R., Ansar. A. I, K. Ali, I. Nesnas, "Rover-Based Visual Target Tracking Validation and Mission Infusion", *Space 2005,* Long Beach, CA, August 2005.

20) Kim W. S., Ansar A., Steele R., "Rover Mast Calibration, Exact Camera Pointing, and Camera Handoff for Visual Target Tracking*", International Conference on Automation and Robotics (ICAR),* Seattle, Washington, pp. 384-391, July 2005.

21) P. Moreels processed the feature extractors for this science target example (2004), see also [12]

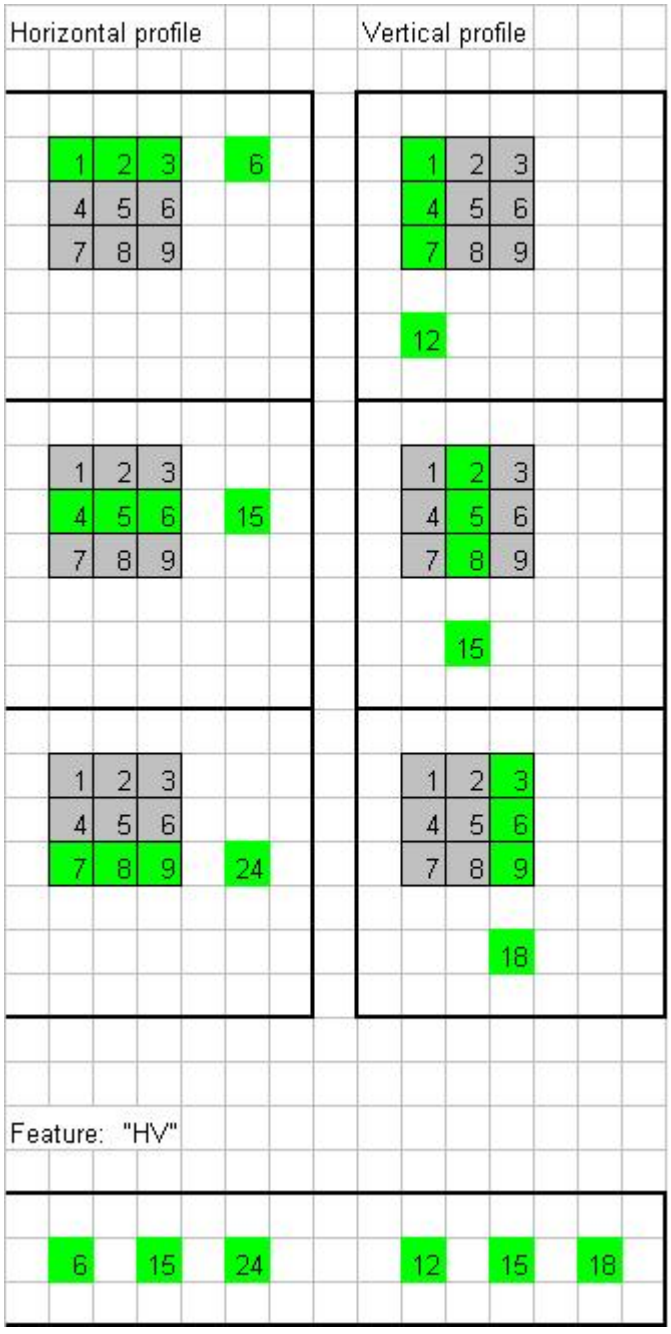
23) C. Assad, J. C. Hanan, A. Csaszar, P. Moreels, C. L. Hughlett, T.-H. Chao, P. Perona, "A Low Cost Test-Bed for Real-Time Landmark Tracking." *SPIE Sensors and Systems for Space Applications* (2007).

24) Bouguet, Jean-Yves, "Camera Calibration Toolbox for Matlab", retrieved March 2007 from http://www.vision.caltech.edu/bouguetj/calib_doc/

25) J.N. Maki, *et al.*, "Mars exploration rover engineering cameras", *Journal of Geophysical Research*, **108**, E12, (2003) 12-1–12-24.

APPENDIX

Example of HV



Example of HVMean

Feature: "HV"								
	Horizontal profile				Vertical profile			
	6	15	24		12	15	18	
	width + height = 6							
	6 / (width + height) = 1.0				12 / (width + height) = 2.0			
	15 / (width + height) = 2.5				15 / (width + height) = 2.5			
	24 / (width + height) = 4.0				18 / (width + height) = 3.0			
	1	2.5	4		2	2.5	3	
	min = 1.0							
	max = 4.0							
	range = 4.0 - 1.0 = 3.0							
	(1.0 - min) / range = 0.0				(2.0 - min) / range = 0.3			
	(2.5 - min) / range = 0.5				(2.5 - min) / range = 0.5			
	(4.0 - min) / range = 1.0				(3.0 - min) / range = 0.7			
	100 * 0.0 = 0.0				100 * 0.3 = 33.3			
	100 * 0.5 = 50.0				100 * 0.5 = 50.0			
	100 * 1.0 = 100.0				100 * 0.7 = 66.7			
Feature: "HVMean"								
	0	50	100		33.3	50	66.7	

Example of HVMeanGradient

Feature: "HVMean"													
	Horizontal profile								Vertical profile				
	0		50		100				33.3		50		66.7
	0 + 50 - 100 - 33.3 = -83.3												
	50 + 100 - 33.3 - 50 = 66.7												
	100 + 33.3 - 50 - 66.7 = 16.7												
	-83.3		66.7		16.7								
Feature: "HVMeanGradient"													
	-83.3		66.7		16.7								

VITA

Pavan Kayathi

Candidate for the Degree of

Master of Science

Thesis: EVALUATION OF TRACKING CONFIDENCE INDICATORS AND FEATURE EXTRACTORS ON A VISUAL TRACKING ALGORITHM.

Major Field: Computer Science

Biographical:

Personal Data: Born in Hyderabad, India on February 22nd, 1983

Education:
Completed the requirements for the Master of Science with a major in Computer Science at Oklahoma State University, Stillwater, Oklahoma in May, 2008.

Experience:
Graduate Research Assistantship, Dept. of Mechanical and Aerospace Engineering, Oklahoma State University, 2006 - 07.

Name: Pavan Kayathi Date of Degree: May, 2008

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: EVALUATION OF CONFIDENCE TRACKING INDICATORS AND FEATURE EXTRACTORS ON A VISUAL TRACKING ALGORITHM

Pages in Study: 54 Candidate for the Degree of Master of Science

Major Field: Computer Science

Abstract:

For visual tracking, a radial basis function neural network algorithm will be used. Coupled with a feature extraction algorithm, the neural network has advantages for pattern recognition, including practical implementation in parallel hardware for real-time operation and low power requirements. Targets vary in terms of texture, contrast, sharpness of edge, relative speed, and size. Various feature extractors exhibit tradeoffs in terms of sensitivity and processing requirements as related to the characteristics of candidate target classes. An analysis of feature extractors based on the horizontal and vertical profile has been provided. A comparison of the distance traveled computed from vision to wheel encoders is presented to observe slipping. Feedback from the network can offer an indication of tracking confidence which will be useful in determining if the estimated position is correct. An attempt has been made to look at the various confidence factors to determine if the position estimated is correct.

ADVISER'S APPROVAL: Dr. Douglas Heisterkamp