A VISION AND BEHAVIOUR BASED APPROACH FOR SHORT-RANGE AUTONOMOUS NAVIGATION OF PLANETARY ROVERS

Michel Makhlouta, Yang Gao[(1)], Karin Shala[(2)]

Surrey Space Center, University of Surrey
Guildford GU2 7XH
United Kingdom
Email:
[(1)]*yang.gao@surrey.ac.uk*
[(2)]*k.shala@surrey.ac.uk*

ABSTRACT

Planetary rovers, an essential robotic tool in planetary expeditions, have been used for decades to study the Moon and the planet Mars. The problem with operating on Mars and the dark craters on the Moon is the long cycle communication delay, which makes it inefficient to remotely operate the rover from Earth. Taking the MER (Mars Exploration Rovers) mission as an example, the rovers are remotely controlled by the ground station to perform long-range navigation. The distance between waypoints, which allows the rover to travel autonomously in a short range, is reported to be limited to a short distance due to the computational complexity of the hazard avoidance algorithm. As a result, the rover can sometimes travel only a few meters per Martian day (sol). Increasing onboard autonomy of the rover can help to reduce human intervention and improve mission efficiency.

The aim of this project is to introduce and investigate a robust and fast onboard navigation system that allows the rover to travel autonomously and safely through a longer distance given a grand goal. We propose to use vision and behaviour based navigation algorithms based on 2D images collected by a single camera. The navigation algorithm relies on the extracted information from image processing, such as features, motion vectors, focus of expansion (FOE), time-to-contact (TTC), and makes behavioural decisions to track targets and/or avoid obstacles. To use a simple sensor for the system has advantages of low power consumption and low computation and design complexity.

INTRODUCTION

Rovers are currently the most useful tool for planetary exploration. During the first rover mission to Mars, called Mars Pathfinder, the rover was teleoperated and the low distance to time ratio achieved due to communications delay clearly showed the need for autonomous navigation systems. Despite the implementation of such systems in the MER mission, the navigation algorithms for hazard avoidance were far from achieving real-time results due to their complexity regarding the used hardware and power resources available on board.

In this paper a new approach is introduced to achieve a short path navigation algorithm which takes as an input a single vision sensor. The algorithm is based on the combination of several known techniques in vision processing and robotics and it is designed by taking into consideration the limitations of low computation power and inexpensive hardware resources. This work is an extension to the proposed approach in [10].

The navigation algorithm, as outlined in Fig.1, first processes the image sequences grabbed from the single vision sensor to extract features form the real world such as corners. Then, the features are grouped into entities based on the distance between them in order to recognize the objects using an unsupervised learning method that does not require any prior knowledge of the environment. Afterwards, optical flow, a motion analysis technique to estimate the two dimensional motion vectors between consecutive images, is applied to those clusters to find the relative velocity between them and the rover. The optical flow method is also used to track the features from one image to another during movement. This method classifies the groups based on their distance from the camera. Since objects close to the rover have a higher velocity vector field than more distant objects, the navigation system is able to interpret depth perception. The developed navigation algorithm follows a reactive approach based on behaviours that relies on the extracted information from the image processing part to make a decision whether to track the manually specified targets or to avoid the collision with obstacles on its path.

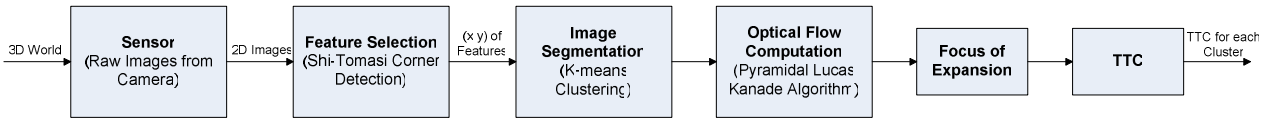

Fig. 1. Blockdiagram of the implemented navigation algorithm

OBJECT REPRESENTATION

Optical flow has proven to be the most efficient method to obtain necessary information for the navigation system using a monocular vision sensor. However, there are a few optimizations to be made in order to further reduce the computational time and resources required. The methods applied to achieve such improvements increase the efficiency of the algorithm, taking into consideration that any tradeoffs regarding accuracy will remain in an acceptable range that ensures the functionality of the system. The first optimization is to cut down the optical flow computation from being generated for the whole images to only the important features. The other optimization increases the efficiency of the decision making part of the algorithm by using the entities to react instead of individual features. There are three important features in two dimensional images that relate to the real world: edges, corners and blobs. Considering further processing on the selected features, there are some constraints that led to consider corner features instead of others. Both of these techniques will not only optimize the system, but they are also used to recognize objects. Once the corners are detected, the ones that have a close distance between them are more likely to belong to one object. By applying a clustering algorithm based on distance, one or more resulting entity is going to represent the areas of interest in the image.

An experiment was carried out to compare different corner detectors based on three tests [1]: A corner stability test checks the stable corners detected throughout the image sequences, then the corner detectors' capabilities of re-detecting the same corners in a frame sequence is assessed and finally the displacement of a corner in the n^{th} frame from its initial position is analysed. Additionally, those three tests can be performed again after introducing an uncorrelated Gaussian noise. As a result, Shi and Tomasi corner detection algorithm [2] was chosen to be implemented in the navigation algorithm proposed. The approach of this corner detector is based on the concept that a corner point should be easy to track between frames. It assumes that the motion can be modelled by an affine transformation, which means a linear transformation followed by a translation, and the method also assumes that the motion is small. Both of these assumptions are true for planetary rover systems: The rover movement is not complex and it is consistent with affine transformation. Additionally, the motion represented by the optical flow vectors will be computed between frames from a vision sensor capable of delivering a frame rate that permits to safely assume the occurrence of small motion displacements between two consecutive frames.

The algorithm first scans the image at each pixel. It considers a window of specified size and it then calculates the covariance matrix M of the pixel's derivates inside that window:

$$M = \begin{bmatrix} \sum D_x^2 & \sum D_x D_y \\ \sum D_x D_y & \sum D_y^2 \end{bmatrix}, \tag{1}$$

where D_x and D_y are the first derivatives and the summations are made over the pixels within the window. Then the eigenvalues of the covariance are computed by solving the equation

$$\det(M - \lambda I) = 0 \ , \tag{2}$$

where λ is the eigenvalue matrix and I is an identity matrix. The minimum eigenvalue at each pixel is then stored to be compared to a threshold.

To achieve good features to be tracked, two criteria must be fulfilled: the covariance of the system must be higher than the image noise level and well conditioned, which means that both eigenvalues must be large and not different by several orders of magnitude. Two small eigenvalues refer to a constant intensity within a window in the image, a large and a small eigenvalue represent a unidirectional texture pattern and two large eigenvalues correspond to corners or any other pattern that can be tracked reliably. For this reason, the minimum eigenvalue previously stored for each pixel is

compared to the noise threshold: if it is higher, then both eigenvalues are acceptable and the covariance matrix is usually well conditioned. Hence, a window is accepted if the minimum of the two eigenvalues of the covariance are higher than the threshold.

Before accepting corners detected based on the two previous criteria as good features, there are a few more calculations to be made on the remaining points. All corners detected so far are compared to a percentage of the maximum eigenvalue of the ones previously stored, usually somewhere between 5% and 10%. If the corners' eigenvalue is lower, they are discarded. As a final step, the corners will be those of the remaining pixels which retain a maximum local eigenvalue among other pixels in their window

After detecting the important features in the images, the system forms a number of groups (entities) from these corner points, which represent the objects. The K-means clustering algorithm is an unsupervised learning technique to cluster data points, which in this case are coordinates in two dimensional image plane. This clustering algorithm will group dispersed features already detected into groups, which leads to forming the shapes of the real world objects in one or more group. The common variable between the data points in one group is the distance.

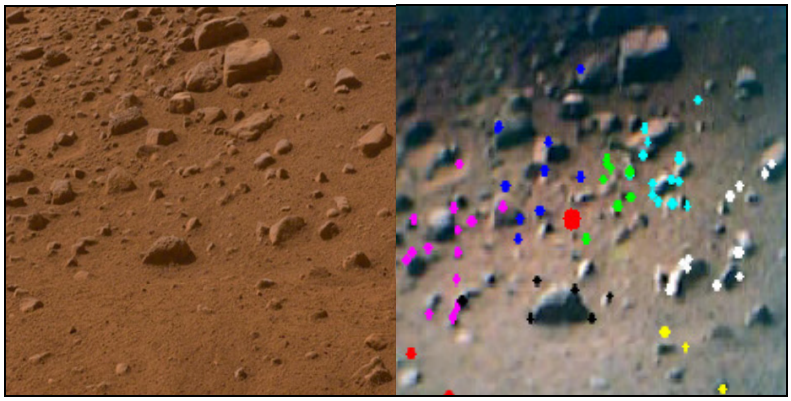

Fig. 2. Corner detection and clustering

The procedure follows a simple and easy way to classify the points coordinates data set through a certain number of clusters fixed a priori. The main idea is to generate, usually randomly, one point for each cluster, which are referred to as centroids. The centroids are placed in a scattered way in the image, because the algorithm will perform better if those points are not close to each other. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. Whenever all the points are assigned to a cluster, the first step is completed and a preliminary grouping is done. Then the centroids' positions are re-calculated so their coordinates become barycentre of the clusters resulting from the previous step. At this point, the two previous steps of calculating distances between the points and new centroids and re-calculating the clusters' new barycentre will be repeated. During this loop, the centroids change their location step by step until no more changes are done.

DEPTH PERCEPTION

At this point, the system is able to detect corner features in the real world and to classify them to represent objects of interest. However, the feature selection and classification process is not sufficient, because an image sequence on a mobile rover will have differences, and features will be displaced. To keep track of the features, and in order not to lose important information such as the amount of displacement, the optical flow technique is implemented to complete the image processing part of the navigation system.

Optical flow (OF), also referred to as apparent motion, is an idea inspired from studying how certain living creatures assimilate the motion of objects or themselves in the real world, seen by their eyes which do not have a vision field in common [3]. In computer vision, the optical flow represents an estimation of the motion within a sequence of images from a video input, either a movie or a camera. The change in motion is represented by the optical flow vector field, showing the displacement of each pixel or selected feature between an image and the preceding one. Each vector defines a direction and velocity magnitude of the objects in an image with respect to the viewer. The 2D vector field is a projection of the 3D velocities of surface points onto the imaging surface, from spatiotemporal patterns of image intensity. Optical flow has several applications in the computer vision domain, such as pattern recognition, vision based

navigation, visual effects and others. Optical flow is not computationally complex and expensive because the processing is done for two dimensional images, which avoids three dimensional image processing of the scenes and objects. This makes optical flow one of the most competitive techniques, suitable for embedded hardware applications such as planetary rovers. However, there is a tradeoff between accuracy and efficiency, which is the reason behind the existence of several methods and techniques to calculate the optical flow.

The accuracy and the efficiency of navigation algorithms are the most essential things to be considered when designing an algorithm for a planetary rover navigation system, given the importance of execution speed and the required robustness to noise. Therefore, it is crucial to choose an optical flow computation method that fits best for this application. Several experiments were made to test different methods, described in papers [4] and [5]. Such tests included noise robustness, computation speed and accuracy. Considering these three factors, the navigation algorithm for the planetary rover is going to be based on the Lucas and Kanade optical flow method, because its average error is small and its execution time lower compared to other methods. In addition, the Lucas and Kanade method is the most robust method when it comes to noisy environments.

The standard Lucas and Kanade method is known to have some weaknesses [6]. The accuracy of the algorithm and its robustness are both directly related to the window size used. A small window size maintains the details in the image and is suitable for the accuracy part of the tracker. On the other hand, a large window size where the optical flow vector is smaller then the window, is in favour of robustness, which is the sensitivity of the tracker with respect to changing in brightness, size of optical flow vector, and others. Therefore, there is a trade-off between robustness and accuracy when it comes to choosing the window size. As a solution, a pyramidal implementation [7] of the standard method is used, which allows the motion vectors to be larger without any accuracy tradeoff.

DECISION MAKING

Although optical flow relatively represents the depth of the feature in the image, there is a scenario where the rover might be heading toward a feature near the focus of expansion. In this case, the optical flow will be relatively small with respect to other features in the other part of the image, even when the depth between the feature and the object became small. To solve this problem, the time-to-contact is estimated. Since optical flow has already been calculated at this point, its resulting vector field is used to estimate position of the focus of expansion in the 2 dimensional plane provided by the camera. Assuming that an optical flow vector $v_i = [v_{xi} \quad v_{yi}]$ is obtained at location (x_i, y_i) in the image and n is the number of vectors in that image, the focus of expansion point p coordinates are estimated by solving the following matrix equation using the least squares with singularity value decomposition (SVD) method [8]:

$$\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ \vdots & \vdots \\ a_n & b_n \end{bmatrix} \begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}, \tag{3}$$

where

$$a_i = \frac{-y_i}{\sqrt{u_i^2 + v_i^2}},\ b_i = \frac{-x_i}{\sqrt{u_i^2 + v_i^2}},\ c_i = \frac{u_i y_i - v_i x_i}{\sqrt{u_i^2 + v_i^2}}\ . \tag{4}$$

The time-to-contact [9] is estimated using a method that does not require prior knowledge of the rover's velocity from additional sensors and assumes that there is only translational movement while the rotational movement is performed. The assumption is verified in the rover's case. The result of the time-to-contact is relative and does not represent a time value in seconds. Therefore, a threshold is used to identify the time-to-contact values that should be considered as a risk and consider the related object at a near position from the rover. It is estimated using the following equation:

$$TTC = \frac{D}{|\vec{V}|}\ , \tag{5}$$

where

$$D = \sqrt{(p_x - u_x)^2 + (p_y - u_y)^2} \ , \tag{6}$$

representing the distance between the starting point of the optical flow vector and the focus of expansion and *V* represents the optical flow vector

$$|\vec{V}| = \sqrt{(v_x - u_x)^2 + (v_y - u_y)^2} \ . \tag{7}$$

The ratio that represents the time-to-contact value solves the problem of the relatively smaller value when the rover is moving toward a near object where the feature is close to the focus of expansion.

The last step in the obstacle avoidance behaviour is to define the acting function of the paradigm. The obstacle avoidance strategy is simply checking for a feature that has a time-to-contact value less than the specified threshold. Once a close feature is detected, the algorithm checks the position of the feature with respect to the focus of expansion point and it sends a signal to the rover to change its path in order to avoid the object. The avoidance strategy is applied to the clusters that group the features, by taking the average of time-to-contact in each cluster and comparing it to the threshold: if it is lower, the algorithm will check the position of the cluster in the image and act to avoid that position. If most of the features in the cluster are on the left side of the image, the rover will make a right turn to avoid it and vice versa. As for target tracking, the algorithm commands the rover to keep the chosen feature to track inside a small window around the focus of expansion so it does not lose it. The feature is tracked continuously during movement of the rover using the pyramidal Lucas and Kanade feature tracker and in order to know when the distance between the rover and the feature is small, the algorithm will also estimate the time-to-contact for that point. Once the target is reached successfully, the algorithm sends a stop signal to the rover and waits for other goals to be indicated. A flow diagram of the decision making is shown in Fig.3.

The reactive architecture that coordinates the two behaviour modules gives the obstacle avoidance the priority over the target tracking in order to avoid collisions with obstacles while tracking a selected feature.

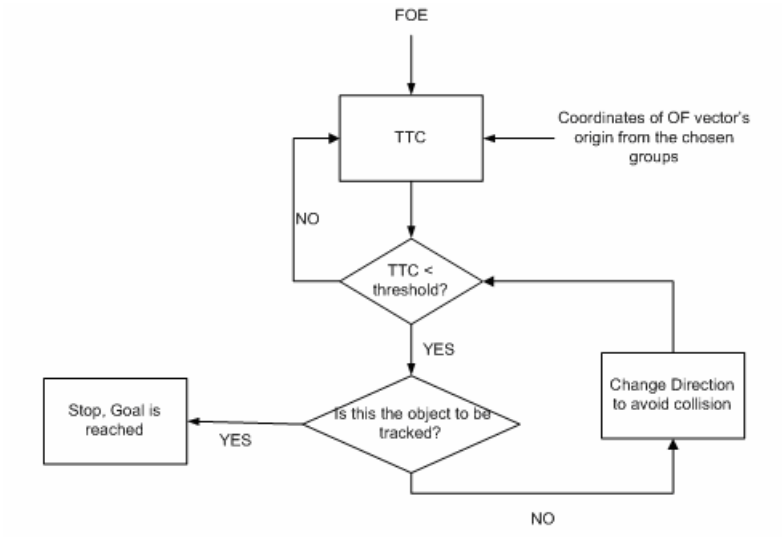

Fig. 3. Obstacle Avoidance and Target Tracking Blockdiagram

EXPERIMENTAL RESULTS

The navigation algorithm has been implemented in C/C++ using the Open Source Computer Vision (OpenCV) library.

Experiment 1 in Laboratory Environment

The algorithm has been tested on an experimental setup consisting of a tracked rover, an onboard CMOS webcam (resolution: 352x280 pixels) and an OBC (1.86 GHz Intel Pentium M processor, 2 GB RAM). This experimental setup verifies the proposed autonomous navigation approach by demonstrating the obstacle avoidance behaviour. The maximum number of corners and clusters to be computed was set to 150 and 8, respectively, which appeared to be enough for obstacle avoidance in a laboratory environment. The overall algorithm used an average of roughly 25% of the CPU speed and 20 megabytes of the memory.

The measured frame rate of the webcam has an average number of approximately 7 frames per second, which is considered to be low, but provides a good experimental setup for worst-case testing of the navigation algorithm. The average time to execute the Shi and Tomasi corner detection is around 47 milliseconds for 150 corners, the K-means clustering algorithm takes only 0.26 milliseconds for dividing the 150 corners into 8 groups. The optical flow computation between two consecutive frames using the pyramidal Lucas Kanade algorithm executes in an average time of 17 milliseconds, being strongly dependant on the distance of the rover to the obstacles. If the rover is far away from the objects, the displacement of the features is small between two consecutive frames, therefore needing a small amount of time for the optical flow computation. If the rover is near an object and the feature displacements are large, the algorithm needs more time to complete the calculations in order to find the corner's match compared to the previous image. Fig. 4 illustrates the optical flow execution time recorded for 100 frames, with the rover moving towards an obstacle. Fig. 5 indicates the corners, clusters and optical flow vectors computed using the proposed method.

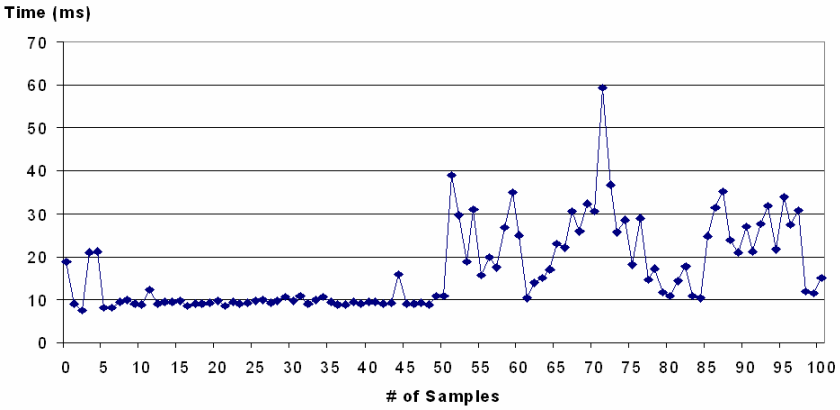

Fig. 4. Optical flow execution time

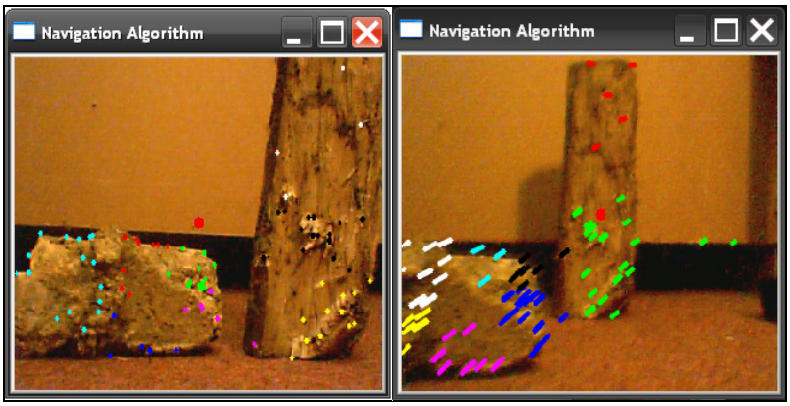

Fig. 5. Corner detection /clustering (left) and optical flow computation (right)

Experiment 2 using MER Images

The algorithm is currently tested on Mars images from the MER rovers Spirit and Opportunity. A major problem arising here is the lack of video material and the limited availability of consecutive images without large displacements. However, some images that roughly meet the requirement of small displacement and the respective computation results can be seen in Fig. 6 and 7. Both image sets were taken by Spirit's Front Hazcam and are low resolution (suitable images could not be found in NASA's Maestro software, and have therefore been extracted from a movie that compiled images of Spirit's first 243 sols). Set 1 in Fig. 6 shows a rock field and the corresponding corner detection/clustering (on the left), as well as the output of the optical flow vector calculation (on the right). It can be seen that the algorithm detects the rotation of the rover wheels, but has difficulty with the objects in the rock field. Set 2 shows images with the rock named "Humphrey" in the background. It can be seen that the algorithm correctly detects this rock.

The results from the two experiments suggest some potential drawbacks of the method. Particularly, the clustering based on distance is not proved to be the best solution, it may be of advantage to use an algorithm that creates the corner groups based on more than one property.

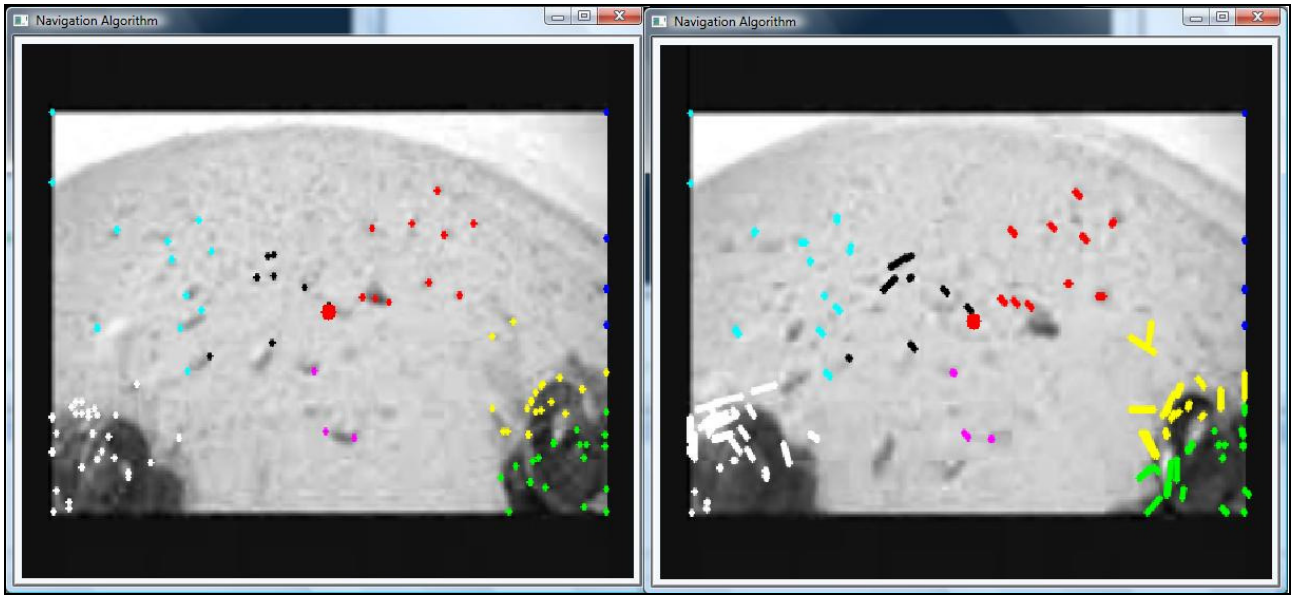

Fig. 6. Spirit Front Hazcam images with optical flow calculation set 1

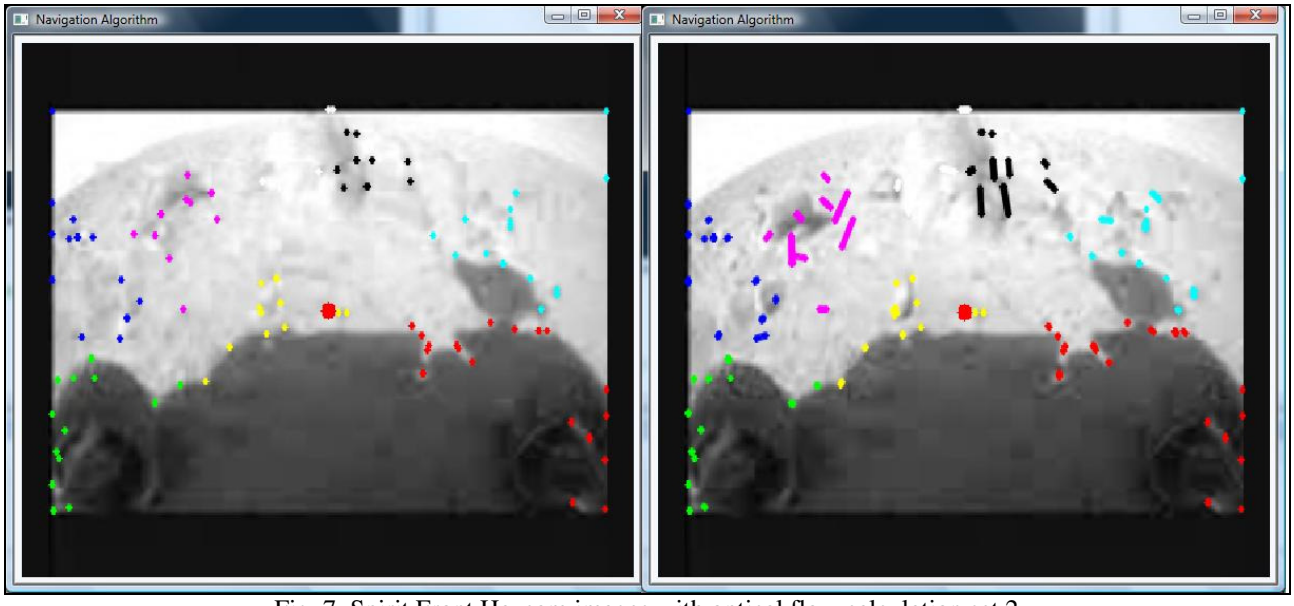

Fig. 7. Spirit Front Hazcam images with optical flow calculation set 2

CONCLUSION

The project presented in this paper suggests a vision and behaviour based approach for autonomous navigation of planetary rovers, given the additional requirement of a low-cost solution with minimal computation and power consumption and inexpensive equipment. The system is based on the combination of several well-known vision processing techniques, such as Shi and Tomasi corner detection, K-means clustering and pyramidal Lucas Kanade optical flow computation. The algorithm is capable of performing both obstacle avoidance and feature tracking and has been implemented in C/C++ and tested on a tracked rover. Furthermore, the algorithm is currently tested on MER images, however, without yet being able to reliably verify the system performance for lack of MER videos.

The autonomous navigation in this algorithm is considered for the short-range navigation compared to a long-range solution. The rover follows a purely reactive paradigm, thus being able to find its path without collisions, but without implying any map building or rover localisation, which is indispensable for a robust planetary application. The next steps to be carried out are thus assessing the robustness of the proposed algorithm by applying it to MER-type images and videos from planetary environments and then embedding it in a "bigger" system including map building and localization.

REFERENCES

[1] Tissainayagam, P. Suter, D. *Assessing the Performance of Corner Detector for Point Feature Tracking Applications*, IVC, No.8, pages 663-679, 2004.

[2] Shi, J. and Tomasi, C. *Good Features to Track*, In Proc. IEEE International Conf. Computer Vision and Pattern Recognition (CVPR). IEEE Press, 1994.

[3] Camus, T. *Real-Time Optical Flow*, PhD Thesis, Brown University Technical Report CS-94-36, 1994.

[4] Barron, J. L Fleet, D. J. Beauchemin, S. S. and Burkitt, T. A. *Performance of optical flow techniques*, CVPR, 1992

[5] Bober, M. and Kittler, J. *Robust Motion Analysis*, In Proc. IEEE Computer Soc. Conf. on Computer Vision and Pattern Recognition, pages 947-952, 1994.

[6] Lucas, B. D. and Kanade, T. *An iterative image registration technique with an application to stereo vision*, Proceedings of the 7th International Joint Conference on Artificial Intelligence, Vancouver, pages 674-679, 1981.

[7] Bouguet, J.Y. *Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the Algorithm*, Microprocessor Research Labs, Intel Corporation, 2000.

[8] Takeda, N. Watanabe, M. Onoguchi, K. *Moving Obstacle Detection using Residual Error of FOE Estimation*, International Workshop on Intelligent Robots and Systems, 1991.

[9] Negahdaripour, S. and Horn, B. K. P. *A direct method for locating the focus of expansion*, A.I. Memo 939, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, January 1987.

[10] Gao, Y., Optical flow based techniques for ExoMars rover autonomous navigation, *Proc. Int. Symp. on Artificial Intelligence, Robotics and Automation in Space (iSARAS)*, LA, USA, Feb 2008