ROBOTICS FOR SMEs – 3D VISION IN REAL-TIME FOR NAVIGATION AND OBJECT RECOGNITION

Fredrik Ekstrand, Jörgen Lidholm and Lars Asplund
School of Innovation, Design and Technology
Mälardalen University
Sweden
Email: fredrik.ekstrand@mdh.se

Abstract By utilizing the power of hardware we can make a versatile, real-time system capable of both navigation and object gripping, based on basic feature detectors. By using an FPGA together with two cameras we can remove the need for descriptors for navigation by performing what we call spurious matching and the use of 3D landmarks. The approach bypasses the problem of outliers and reduces the time consuming task of data association, which slows many matching algorithms. We also present an approach to object gripping by taking advantage of the full capability of the feature detector.

1. Introduction

The goal of the project Robotics for Small and Medium-sized Enterprises (SMEs) is to develop a system for automation with focus on flexibility, interactive instruction and reconfigurability. Such an intelligent, semi-autonomous and mobile system requires sensors for analysis of the surroundings. The perfect fit is vision, probably the most versatile sensor available today.

The possible applications with vision are endless, but they all require analysis of the image. For this analysis to be possible we need to classify the image content with feature detectors. The fundamental task of these detectors is the reduction of image complexity. In order to reduce the image information, the feature detector finds lines, edges or corners, or a combination thereof. Various approaches have been made on adapting these principle feature detectors into more complex, high-level detectors with different sets of descriptors. The idea is to increase the invariant properties of the detector and thereby increase the repeatability.

The frame rate of these feature detectors rarely approach real-time when implemented in software on standard workstations. We are now in a situation where it is possible to embed highly sophisticated algorithms in hardware (FPGA; Field Programmable Gate Arrays - or reconfigurable hardware) that can manage the high resolution and the high bandwidth of Mega-pixel sensors in real-time, in this context defined as a frame rate of around 30Hz. The advantages of an FPGA is manifold; the parallel properties of an FPGA makes for a high-throughput, small footprint system, and the comparatively low power consumption makes it ideal for mobile applications.

When features have been extracted, some sort of matching needs to be done, e.g., by tracking a feature in subsequent images. The general idea is that if this is to be performed in real-time it requires a high repeatability detector and a computationally light matching algorithm with minimal dynamic properties.

In this work we intend to show that the middle approach is possible. We suggest that it is possible to produce adequate matching for navigation and object recognition by a lower performance detector and by performing the matching in a 3D, as opposed to a 2D, environment.

2. Related Work

Robot navigation is a well-explored subject with vision based navigation being where the current focus lies. The approach of using an FPGA for the system is also becoming widely adopted as it enables real-time image processing [1] [6], which is a crucial part in mobile applications [4]. For certain applications, FPGAs are better suited than desktop computers due to their parallel structure. In [15] an FPGA implementation outperforms a PC by one order of magnitude for the SIFT detector [8]. The power of the FPGA is further evident when comparing our implementation with [11] where they are unable to run Harris corner detector in real-time on a computer with an Opteron processor running at 2.6 GHz.

Both navigation and object detection by vision requires matching of images, or rather features in separate images. Tracking features in real-time in subsequent images from a camera is not a trivial task, partially due to the fact that it requires a very stable feature detector with high repeatability [3]. The current feature detectors with the highest repeatability, such as SIFT [8] and SURF [2], create descriptors for each feature in order to simplify the matching task. Unfortunately, the high dimensionality of such a de-

scriptor means that it is computationally intense [9].

All matching algorithms are faced with the correspondence problem, i.e., how to match corresponding features in two images without assigning any incorrect matches. The approaches differ, from *cross-correlation* to the *sum of squared differences*, but there will always be outliers, features not correctly matched, or not matched at all. Many have tried to minimize the occurrence of outliers, and in [13] a comparison, and yet a new approach, is presented.

3. Experimental platform

We have produced a system intended to work as a general-purpose research platform for FPGA-based vision.

The system uses the MT9P031 5-megapixel CMOS camera sensor from Micron. The imaging sensor is capable of running at 96 MHz, with the frame rate being dependent on the clock frequency and the frame size.

The FPGA board has a size of 70×55mm and is equipped with a Xilinx Virtex II XC2V8000 FPGA together with 256 Mbit flash, 512 Mbit SDRAM and a CPLD. The flash memory is for storing FPGA configurations and is accessed at power on by the CPLD which loads the FPGA. The FPGA board is mounted on a carrier board that incorporates power supply, and a USB controller.

4. Feature detectors

A feature can be a corner, edge or any salient region which can be extracted from an image.

One of the first feature detectors is Moravec's corner detector [10] from 1977. Since then, many other feature detectors have been developed with different qualities [12]. Most detectors are designed with repeatability in mind, although some are designed for other properties, such as speed [11]. Repeatability, as defined in [12], is an important property, however, for our application, speed is equally important.

4.1. Stephen and Harris detector

The Stephen and Harris combined corner and edge detector [7] was presented in 1988 and expands on the work of Moravec. In [12] the authors concluded that, among the tested feature detectors, (Foerstner, Cottier, Heitger, Horaud, Harris and Improved Harris), the improved version of the Harris corner detector performed best regarding repeatability and information content. The original implementation of the same detector was, however, not far behind.

Moravec's corner detector calculates the variation in intensity in an image and looks for low self-similarity in a point. A corner is defined as a point with low similarity to the surrounding region in all directions, i.e., a point where the minimum change in intensity, in any direction, is large (above a certain threshold) [7].

Stephen and Harris improved on the problem of anisotropy and noise in Moravec's detector by introducing an analytic expansion about the shift origin together with a smoothing Gaussian filter. They also introduced a response function, that includes a structure matrix calculated from image derivatives, which indicates the quality of the detected feature and allows for the filtering out of less distinctive features with the use of a threshold.

4.2. FPGA implementation of Harris corner detector

We have a VHDL implementation of the Stephens and Harris combined corner and edge detector. It was originally implemented as an undergraduate thesis for a previous vision system. We have adapted it to a new, larger FPGA, allowing us to increase the parallelism and thus improve the speed.

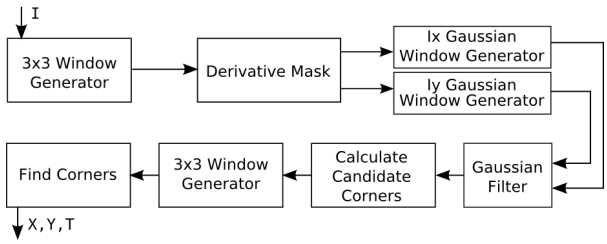

Figure 1. A block diagram of our VHDL implementation of the Harris corner detector.

The corner detector uses 3×3 and 5×5 pixel windows. This is the only buffering required, all other processing is performed as the pixel data arrives. In the block diagram in figure 1, our implementation of Harris corner detector can be seen. The process consists of 7 major internally piped blocks. The first block creates a 3×3 sliding window. When two pixel rows plus one pixel have been extracted from the camera the first window is passed on to the "Derivative Mask" block.

The derivative block calculates the intensity x- and y-gradients. These values, the first derivatives, are then passed onto the window generator for the multiplication/-Gaussian stage, which creates a 5×5 sliding window.

In the multiplication stage, the structure matrix is calculated and then run through a Gaussian filter. The filter is not a true Gaussian function as the values are selected to enable shifting, but no performance degradation has been observed for the approximation, which can be supported by [5] that shows that Gaussian weighting need not be the optimal weighting function.

The filtered value is then used in the response function and its result is filtered so that only the local maxima within the 3×3 sliding window generates a corner response, as long as it exceeds the current threshold.

5. Interest point location

We define an interest point as a stereo-matched feature that can be located in a coordinate system as a landmark,

which a robot can use for navigation. In this section we describe how we can calculate the location of a landmark from two stereo-matched features. The same procedure, in reverse order, can be followed to calculate the pixel coordinate at which a landmark should appear, given the robots current location and attitude.

We use the right-handed coordinate system with positive X to the right, positive Y in front and positive Z above.

The full definition of the robot absolute vector defines the position in three dimensions and the attitude in three dimensions (5.1). The robot center is located at the floor in the center of the robot in the x, y plane.

$$\mathbb{R} = (X_r, Y_r, Z_r, \alpha_r, \beta_r, \gamma_r) \tag{5.1}$$

Since the robot is moving in a controlled indoor environment without slopes, we can consider Z_r constant and zero. The same applies for α_r and β_r. The stereo camera rig has a fixed location on the robot and the constant relative vector of each camera is defined in (5.2), where n marks the camera, left or right. The vector is relative to the robot center. To simplify the stereo matching the $\hat{\beta}$ factor should be zero and the $\hat{\alpha}$ should be the same for both cameras, resulting in that line j in the left camera corresponds to line j in the right camera.

$$\underline{c}_n = \left(x_n, y_n, z_n, \hat{\alpha}_n, \hat{\beta}_n, \hat{\gamma}_n\right) \tag{5.2}$$

The absolute vector of each camera can be calculated by adding the relative camera vector to the absolute robot vector:

$$\begin{aligned}\mathbb{C}_n &= (X_r + x_n, Y_r + y_n, Z_r + z_n, \\ &\quad \hat{\alpha}_n, \hat{\beta}_n, \gamma_r + \hat{\gamma}_n) && (5.3)\\ &= (X_n, Y_n, Z_n, \alpha_n, \beta_n, \gamma_n) && (5.4)\end{aligned}$$

Every pixel in an image corresponds to a two dimensional direction which can be calculated from the focal length of the lens f and the pixel separation on the camera chip P_{width} and P_{height}. The two angles θ and ϕ and an unknown length r form a polar vector (5.7).

(X_p, Y_p) denotes the pixel coordinate, with the camera center at $(0, 0)$, and Q is the transformation from pixel coordinates to a polar vector.

$$\theta = \arctan\left(\frac{X_p * p_{width}}{f}\right) \tag{5.5}$$

$$\phi = \arctan\left(\frac{Y_p * p_{height}}{f}\right) \tag{5.6}$$

$$Q(X_p, Y_p) = (r, \theta, \phi) \tag{5.7}$$

By using (5.5) and (5.6) we can find the angular distance between every pixel.

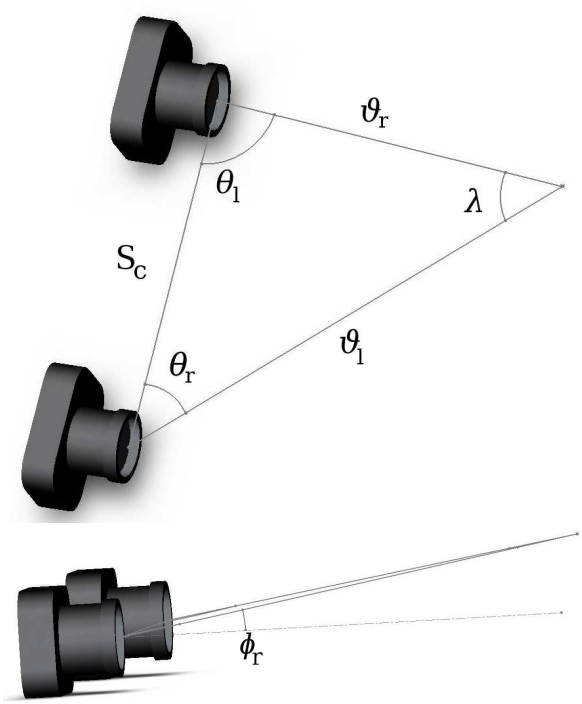

Figure 2. The angles from each camera to a feature point. θ for the left and right camera, the camera separation S_c and ϕ, which should be the same for both cameras.

Lets consider the case where we know which feature in the left camera corresponds to which feature in the right camera. We can calculate the cartesian coordinate of the interest point according to (5.8-5.14) and figure 2. The camera separation is known and denoted S_c.

$$\lambda = \pi - \theta_l - (\pi - \theta_r) \tag{5.8}$$

$$\frac{\vartheta_n}{\sin(\theta_n)} = \frac{S_c}{\sin(\lambda)} \tag{5.9}$$

$$\vartheta_n = \frac{S_c * \sin(\theta_n)}{\sin(\lambda)} \tag{5.10}$$

C and P marks the cartesian and polar coordinate system respectively or a transformation between the two.
The cartesian location of camera n.

$$C(\mathbb{C}_n) = (X_n, Y_n, Z_n) \tag{5.11}$$

The attitude of camera n as a polar unit vector.

$$P(\mathbb{C}_n) = (1, \alpha, \gamma) \tag{5.12}$$

The direction and distance to the interest point k.

$$P(\mathbb{I}_k) = (\vartheta_n, \phi_n, \theta_n) \tag{5.13}$$

The space location of the interest point.

$$C(\mathbb{I}_k) = C(\mathbb{C}_n) + C(\vartheta_n * (rot_{\phi_n, \theta_n} P(\mathbb{C}_n))) \tag{5.14}$$

The conversion from polar vector to cartesian coordinate are simple sin and cos transformations.

5.1. Image sequence feature tracking

To track features in an image sequence is not a trivial problem when using feature extractors such as the Harris corner detector [12].

Tests have shown that a simple tracker, like nearest neighbor, is not reliable enough [3]. Using feature descriptors can simplify the tracking problem substantially, however, they require more computation in the feature extraction phase, and still do not solve the problem completely. Implementations of the SIFT algorithm [8], often do not manage more than a few frames per second.

In real-time applications, direct matching seems like the best approach, as it is desirable to match features on a real-time basis. However, if the frame rate is sufficiently high, intermediate matching can be more than adequate. Statistical approaches for handling outliers by confidence values are well-explored, but they are normally time-consuming and can be problematic when matching in a 2D environment [13]. By moving the matching part onto the 3D coordinates from a stereo-vision camera system, it is possible to eliminate the uncertainty of 2D pixel coordinates.

5.2. Spurious matching and landmark evaluation

To match a feature in the left image with a feature in the right image is known as the correspondence problem. A common approach is to use a correlation window around the features and calculate a matching score, using a statistical method. The matching score can be calculated with methods like *cross-correlation*, *sum of squared differences* and χ^2, for example [13]. In order to successfully use statistical methods, it is necessary to calculate the matching score for many different pairs to find the match with the highest score. Additionally, it is also necessary to find the outliers, or false matches.

Our approach is adapted for a real-time vision system where the data is processed as a stream. No image is stored as a whole, line buffers, however, are used.

A feature appearing at pixel row n in the left camera must appear, if existing, on row $n \pm m$, where $m = 1$ under the condition that the camera distortion is corrected and that the cameras are perfectly aligned. The horizontal limitations can be found by knowing the attitude of the cameras. The search window denoted $W_m(F_i)$ represents the maximum area in which a feature in the right image must be located to correspond to feature F_i in the left image.

By matching every feature F_i in the left image with every feature within $W_m(F_i)$ in the right image we get a set of possible landmarks $LMK(F_i)$. Within this set of 3D coordinates there can be only one that corresponds to the actual landmark, which one is unknown. We call this spurious matching. Instead of trying to find the correct stereo correspondences, we try to find which landmarks in the environment are the correct ones. While moving the robot, measuring the location of the robot using wheel based odometry, and continuously calculating the possible landmark location for every feature F_i, the reappearing landmarks are then put in a landmark database with an increasing confidence related to uniqueness and stability of the landmark location.

To rely on odometry can be risky because it is a relative measurement system with no point of calibration. As soon as enough landmarks have been located with good confidence, the odometry system can be used solely as a support system and is no longer required for the navigation, which can use visual odometry.

By predicting the robots location and attitude before each iteration, using for example a Kalman filter, we can find the pixel coordinate for each possibly visible landmark and exclude those features from the images. This reduces the amount of features that need to be matched.

5.3. Computational requirements

Calculating the space location of a feature pair, as seen in (5.5-5.14), requires 25 operations. Harris extracts approximately 300 corners from a 320×480 pixel frame without being too cluttered. In average this means less than one corner per line, the maximum number of corners possible on a single line is $\frac{320}{3} = 106$, though very unlikely (see section 4.2).

A pessimistic number of matches per feature could be around 20, which would render in 6000 landmark calculations per frame. 25 operations on 6000 landmarks would result in 150'000 operations per frame, which is rather low.

6. Object Recognition

The term object recognition means to identify an object as something of which one has prior knowledge. This requires some sort of model of the object to be stored with associated feature points. Matching of live feature points against stored ones can be a difficult task, especially as the Harris detector is not scale-invariant [14] and its localization performance average.

Object recognition, according to the above definition, might be desirable, but for our target application, pick-and-place, it is not crucial. For companies that require handling of thousands of articles in a dynamic manufacturing environment, the task of creating a model for each article can be daunting and maybe even an impossible feat. Instead of recognizing the object, it should be sufficient to recognize the way to handle the object, in this case, a gripping scheme.

The setup is two cameras mounted on a robot arm behind the gripper, with known displacements. In order to grip an object, one needs to know, at least, the shape of the object. The most straight-forward approach is to identify two opposing surfaces suitable for the gripper in question. Surface detection can be difficult with a detector created for sharp discontinuities, normally associated with contours, as

opposed to the uniform nature of surfaces. Even if most surfaces produce corner responses, the corner density is usually too low for surface modeling.

For the system to be versatile and cost-effective, it would be an ideal solution to use the Harris detector for both the navigation and object handling parts. In order to overcome the limitations of the corner properties we use the often overlooked part of the Harris detector - the edge detector. The response function of the Harris detector indicates the cornerness value [7] of the pixels - the higher the value, the stronger the corner. On the opposite side of the scale is the edge strength - the more negative the number, the stronger the edge. We have not found any comparison between Harris and other edge detectors, but its performance is adequate for our needs.

The approach to object gripping is a combination of edge and corner detection. Start off with detecting both corners and edges. By setting the threshold at a high enough level we receive a fairly limited number of corners and edges, and this is important for two reasons. Firstly, the higher order edges are most often the contours, i.e., the surface and object boundaries. Secondly, parallel lines likely belong to the same object, and a limited number of parallel lines simplifies the matching.

We do approximations of the edges that are near linear. Only edges above a certain length (even multiple small edges adding up to this length), are approximated with a linear function, expanded across the image. Out of these approximated lines, parallel ones are grouped together. Next, we match all the Harris corner points located between the first two lines in a group between the two cameras using the epipolar constraint together with neighborhood distribution. The left-most line pairs of the right camera is the starting position of the matching as this area is most likely present in both cameras. The amount of corners is limited due to the high threshold, hence the number of possible matches are fairly limited.

After the corners have been matched, their 3D-coordinates are compared in order to see if they are coplanar. If so, the edges encompassing the corners are surface boundaries. We then proceed to find boundaries perpendicular to the existing ones. The criteria here is that they need to cover a certain degree of the distance between the lines or be of sufficient continuous length. If the corners are deemed not coplanar, a new check is performed using the recently discovered perpendicular boundaries as temporal surface delimiters. If none of the areas are coplanar we move on to another set of boundaries in the left camera until the criteria is fulfilled or all areas exhausted. When the regions are established, expansion of the newly detected boundaries are made so that all coplanar corners are encompassed, and no others. The surface is stored and its boundaries defined by the 3D-coordinates of the area extremes, i.e., the corners. This is then repeated for all parallel pairs in a group, and for all groups.

Any neighboring, coplanar regions are merged into one. Surfaces without any corners are treated as a non-surface, i.e., a hole when surrounded by other surfaces. The cameras then move into a new position in order to get more information of the object and to identify a surface parallel to already defined ones. With multiple surfaces available for gripping the most suitable ones are selected according to a volume based, mass distribution scheme. The definite representation of object boundaries and the identification of the best suited gripping area are not possible until the object has been gripped and moved.

7. Results

Our FPGA based stereo vision system is capable of real-time feature extraction, using the implemented Stephen and Harris combined corner and edge detector. To stereo match these features, for landmark location, is not a trivial problem. We present an approach which we call spurious matching allowing us to validate which matches correspond to real landmarks by moving the robot and extracting the features at different viewpoints.

For performance results of the corner detector see table 1, and table 2 for frame rates of Harris corner detector on our system.

Table 1. Performance total of Harris corner detector at different frame rates

pixels/frame	fps	Instr./pix	Cameras	MIPS
148'800	27	286	2	2'298
148'800	34	286	2	2'894

Table 2. Performance of our implementation of Harris corner detector.

Frame size	**Cam freq.**	**FPGA freq.**	**FPS**
320×480	96MHz	100MHz	65 fps*
320×480	50MHz	100MHz	34 fps
320×480	40MHz	100MHz	27 fps

* Theoretical value which we have not been able to verify.

8. Future work

The proposed spurious matching algorithm has not been fully verified yet, there are several performance factors which need to be evaluated like, camera discrepancy, odometry precision and landmark localization accuracy. The object gripping application is not fully implemented yet.

9. Acknowledgements

This project is supported by Robotdalen. The authors would also like to acknowledge *Xilinx* for their kind donation of our FPGA's and design software tools, *Hectronic* for the design and manufacturing of our FPGA boards.

10. Authors

Fredrik Ekstrand, Ph.D student at Mälardalen University in the area of FPGA-based Vision Systems since 2007. Earned his Bachelor in Electronics at Uppsala University in 2001. Head of development of electronics at Senseboard Technologies AB during 2002-2003. Worked at RealFast HW with FPGA development and patent applications, before starting his own company in 2005.

Jörgen Lidholm completed his masters degree in computer engineering in the spring of 2006. His masters thesis was in the area of robotics. During 2006 Jörgen worked as a research engineer at Mälardalen University building a control system for a mobile industrial robot platform. Since 2007 Jörgen is studying for a Ph.D degree at Mälardalen University in the area of robotics navigation using FPGA based real-time vision.

Lars Asplund, professor at Mälardalen University in Computer Science with emphasis on System on Chip. Earned his PhD at Uppsala University in Physics 1977 under supervision of laureate Kai Siegbahn. Founder and owner of Asplund Data AB, one of the majority owners and CTO of Senseboard Technologies AB. Author of nine fundamental books in electronics.

Research experiencees in Physics, Electronics, Learning Systems, Safety Critical Systems, Real-Time Systems, and Distributed Systems, all at Uppsala University, and now lately implementation issues for vision algorithms in System on Chip (ChipVision).

Current industrial involvement related to robotics is

- Some cooperation between Senseboard Technologies and Kawada Industries and their human robot.
- Development of a more efficient technique, based on a new sensor system, for Ramsta robotics AB.
- Development of autonomous walking for the underwater robot Roman in the company INVO AB.
- In cooperation with Robotdalen an autonomous carrier for industrial robots will utilize the latest navigation system based on ChipVision.
- There will be a start-up company in the near future that will use both the ChipVision technology and the omni directional driving system used in the Aros-robots.

Lars Asplund was the main driving force in establishing Real-Time research in Sweden and he chaired SNART for several years, and he was also the head of department for department of Computer Systems at Uppsala University for five years. In the year of 2004 Lars Asplund started the first and so far the only Engineering programme (4.5 years) in robotics in Sweden at Mälardalen University.

References

[1] P. Arribas and F.-H. Macia. FPGA board for real time vision development systems. *Devices, Circuits and Systems, 2002. Proceedings of the Fourth IEEE International Caracas Conference on*, pages T021–1–T021–6, 2002.

[2] H. Bay, T. Tuytelaars, and L. J. V. Gool. Surf: Speeded up robust features. In A. Leonardis, H. Bischof, and A. Pinz, editors, *ECCV (1)*, volume 3951 of *Lecture Notes in Computer Science*, pages 404–417. Springer, 2006.

[3] A. Bissacco, S. Ghiasi, M. Sarrafzadeh, J. Meltzer, and S. Soatto. Fast visual feature selection and tracking in a hybrid reconfigurable architecture. In *Proceedings of the Workshop on Applications of Computer Vision (ACV)*, June 2006.

[4] D. Cardon, W. Fife, J. Archibald, and D. Lee. Fast 3d reconstruction for small autonomous robots. *Industrial Electronics Society, 2005. IECON 2005. 31st Annual Conference of IEEE*, pages 6 pp.–, 6-10 Nov. 2005.

[5] T. Cooke and R. Whatmough. Using learning algorithms to improve corner detection. In *Proceedings of the Digital Imaging Computing: Techniques and Applications (DICTA 2005)*, page 54, December 2005.

[6] T. H. Drayer, J. G. Tront, R. W. Conners, and P. A. Araman. A development system for creating real-time machine vision hardware using field programmable gate arrays. In *HICSS '99: Proceedings of the Thirty-Second Annual Hawaii International Conference on System Sciences-Volume 3*, page 3046, Washington, DC, USA, 1999. IEEE Computer Society.

[7] C. Harris and M. Stephens. A combined corner and edge detection. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, 1988.

[8] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[9] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 27(10):1615–1630, 2005.

[10] H. Moravec. Towards automatic visual obstacle avoidance. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, page 584, August 1977.

[11] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pages 430–443, May 2006.

[12] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of interest point detectors. *International Journal of Computer Vision*, Vol. 37(2):151–172, June 2000.

[13] P. Smith, D. Sinclair, R. Cipolla, and K. Wood. Effective corner matching. In J. N. Carter and M. S. Nixon, editors, *BMVC*. British Machine Vision Association, 1998.

[14] P. J. Stephen Se, Timothy Barfoot. Visual motion estimation and terrain modeling for planetary rovers. In *i-SAIRAS 2005 - International Symposium on Artificial Intelligence, Robotics and Automation in Space*, September 2005.

[15] P. J. T.-J. M. Stephen Se, Ho-Kong Ng. Vision based modeling and localization for planetary exploration rovers. In *Proceedings of the 55th International Astronautical Congress 2004*, pages 1–11, 2004.