Final Class Project, ENAE788E – Embedded Real-time Systems

Rover Vision Localization System

Final Report

By: Martin F. Stolen
Project Team: Gregorio Monge

Contents

1 Introduction 3
2 Background 3
2.1 Purpose 3
2.2 Hardware/Software Used 3
3 Visual Positioning 4
4 Software Developed 5
4.1 General Layout 5
4.2 Acquiring Images 7
4.3 Locating Corners 7
4.4 Calculating Extrinsics 8
4.5 Interfacing with Camera Tracking System 8
5 Testing 8
5.1 Introduction 8
5.2 Setup 8
5.3 Test Parameters 12
5.4 Results 13
5.4.1 Effect of distance 13
5.4.2 Effect of Angle 15
5.4.3 Lens effects 16
5.4.4 Effect of Number of Points Used 17
5.4.5 Effect of Checker Size Inaccuracy 18
5.4.6 Rover Integration 18
5.5 Discussion 20
6 Conclusion 21
References 21

1 Introduction

This report describes a final class project on the localization system for an autonomous rover. The project is aiming to allow a rover to accurately position itself with respect to some static feature, using a vision system on a 2 DOF stand. The scenario envisioned, the platform and its components will first be described. Then the vision system localization is detailed. The possible solutions investigated for a camera object tracking system is also described, as well as the software developed to enable visual localization. Finally the extensive testing performed is described, the results presented and the results discussed. This project was performed in cooperation with Gregorio Monge, which developed and assessed the implemented pointing system for the rover camera.

2 Background

2.1 Purpose

Accurate positioning is important in many applications where autonomous robots are involved. Especially for motion planning involving mobile robots can this capability be crucial. Using this feature a robot can use its sensors to map its environment and then easily use this map and its positioning within it to move safely between obstacles. It can also be used to record the accurate position of a sample of interest, or guide the rover to a accurate position with respect to some static feature, for example a sample drop or a charging station.

For situations where GPS positioning will not be available, as is the case in underwater and planetary exploration, an alternate method of locating a robot is desirable. This project will attempt to address the issue for the limited case of using visual positioning with respect to a static object of known position. The vision system localization described is based on a single camera configuration. Stereo cameras are available on the rover, but the added complexity of stereo triangulation and the desire to keep the system as simple as possible led the author to this configuration.

2.2 Hardware/Software Used

The platform for the project was the six-wheeled rover currently under testing in the Space Engineering and Hardware Development Laboratory. The rover has twelve servos (Hitech HS-5445 MG), six of which are used for forward motion, four for turning the two front and back wheels and two for pointing the camera. The servos are controlled by a Pontec SV203 controller connected via a serial connection to a PC 104+ board (1 GHz, 512 MB Ram). This board is part of a stack of several PC-104 boards that perform different functions on the rover. The two cameras mounted on the rover are Unibrain Fire-I, which is connected through a FireWire connection. The cameras have a 640 x 480 resolution (24bit RGB).

The previous operation of the rover is based on a path planning algorithm that produces a path based on a goal point relative to the rover, and uses counters mounted in the wheels to predict the rover's position. A vision system for mapping the environment and avoiding brightly colored obstacles has also been implemented. However, no attempt had previously been made to autonomously position the rover accurately at a given absolute position.

Also available when performing the detailed tests of the vision system was a high precision optical table, and a computer aided measuring device, a FARO Platinum arm [5]. This system has the capability to reasonably quickly and accurately (0.0005”) measure the position and orientation within about a 1 meter radius of the device.

3 Visual Positioning

The single-camera vision system on the rover was used to attempt to localize the rover with respect to some global coordinate frame. This can be achieved by knowing the position and orientation of some static feature in terms of the local camera frame. To find this, some information about the camera and its lens is required. This information, the intrinsic parameters of the camera, can be found by calibrating the camera. For the camera to be used on the rover, the results of such a calibration can be seen in Table 1. These parameters were found using software developed by the author based on functions available in the Open Computer Vision (OpenCV) library [1]. Some issues regarding this data is further discussed in section 5.5.

Intrinsic Parameters				
Parameter	*Values*			
Focal length, pixels (horizontal, vertical)	870.11		870.03	
Principal point, pixels (x, y)	333.97		219.07	
Distortion coefficients	-0.0782	0.1941	0.0018	-0.0023

Table 1 Intrinsic parameters of rovers left camera

The position and orientation of an object is to be determined if a given set of points on the object can be recognized by the vision system. For this project OpenCV functions was used for this application as well. The geometry of the location of these points with respect to each other must be known beforehand. Equation 1 [2] shows the relationship between a point *i* in the image frame (x_{image}, y_{image}) and the same point in the camera frame (${}^cx_i$, ${}^cy_i$, ${}^cz_i$). Note that *f* is the actual focal length (not based on pixel size), s_x and s_y the effective pixel size in the respective directions and (o_x, o_y) is the location of the principal point.

$$x_{image} = -\frac{f}{s_x}\frac{{}^cx_i}{{}^cz_i} + o_x$$

$$y_{image} = -\frac{f}{s_y}\frac{{}^cy_i}{{}^cz_i} + o_y$$

Equation 1 Relationship between point in camera and image frame [2]

This can be applied to all the points located on the checkerboard. The resulting extrinsic parameters of the situation, which is a rotation matrix *R* and a translation vector *p,* can now be found. This is done by using the relationship between the coordinates of multiple points in the object frame (${}^ox_i$, ${}^oy_i$, ${}^oz_i$) and the camera frame as seen in Equation 2 [2].

$$\begin{bmatrix} {}^{c}x_i \\ {}^{c}y_i \\ {}^{c}z_i \end{bmatrix} = R \begin{bmatrix} {}^{o}x_i \\ {}^{o}y_i \\ {}^{o}z_i \end{bmatrix} + \underline{p}$$

Equation 2 Relationship between point in object and camera frame

The checkerboard is assumed to have a known position and orientation (transformation) relative to some world frame, *W*. And the images taken can be used to determine the transformation of the camera, *C*, with respect to the checkerboard (the extrinsic parameters of the camera), *B*. If the transformation of the camera relative to the rover, *R*, is also known, the rover can be accurately positioned in the world frame. This will have to be determined through the camera base frame, *F*, which is rigidly attached to rover. Equation 3 shows this transformation of the rover in the world coordinate frame.

$${}^{W}_{R}T = {}^{W}_{B}T\,{}^{B}_{C}T\,{}^{C}_{R}T$$

Equation 3 Transformation of Rover in Global Frame

4 Software Developed

4.1 General Layout

All the vision localization software will run from a desktop computer (kim-105.ssl.umd.edu), connected to the rover cameras though an Ethernet connection. Although the rover computer itself could have performed this task, the OpenCV library is rather extensive, and would tie up resources possibly needed by other research being performed on the rover. It also simplified the development and testing of the software.

The vision localization system was set up to be running as a single thread. This could be done as the camera tracking system is dependent on all the visual positioning calculations being completed before attempting to update the camera angle. The software was initially thought to be based on one main class which would contain instances of the other classes needed to perform the localization. Due to changes in the work distribution and to keep the software simple for the sake of debugging, however, only the process of attaining the points and calculate the extrinsic parameters were implemented in classes, which can be seen in Figure 1.

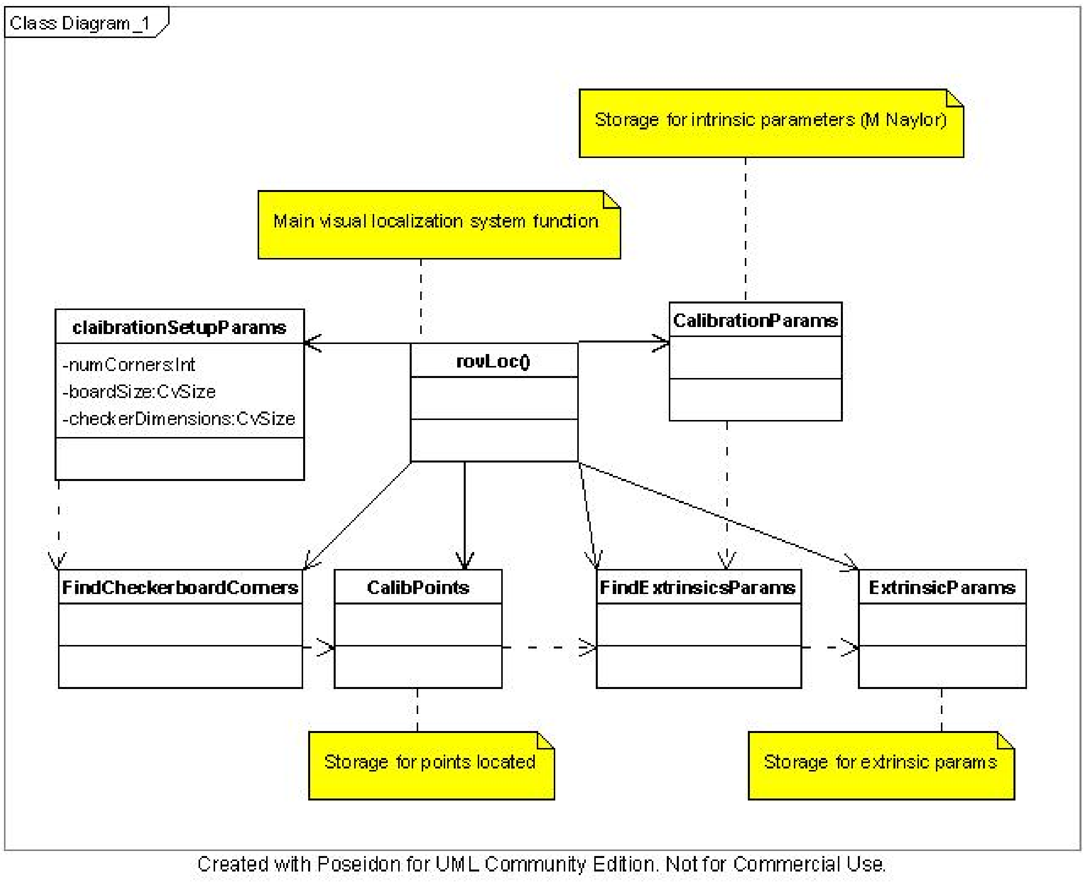
Class Diagram_1
Storage for intrinsic parameters (M Naylor)
Main visual localization system function
claibrationSetupParams
-numCorners:Int
-boardSize:CvSize
-checkerDimensions:CvSize
rovLoc()
CalibrationParams
FindCheckerboardCorners
CalibPoints
FindExtrinsicsParams
ExtrinsicParams
Storage for points located
Storage for extrinsic params
Created with Poseidon for UML Community Edition. Not for Commercial Use.

Figure 1 Class diagram for "core" of vision localization system

This "core" of the vision system is fed an 8 bit grayscale *tlImage* and the geometric relation of the physical checkerboard. The image is then searched for a checkerboard pattern in the *findCheckerboardCorners* class, which stores the results in a separate storage class (dotted arrow). If the localization is successful the extrinsic parameters can be calculated, which is the output of the "core". The code is set up in classes to enable it to be flexible, should for example a new method of locating points for localization be desired.

All other tasks performed by the vision system localizer was separated into functions for simplicity, and these are together with the "core" placed in the *localizer* namespace to prevent misuse and multiple declarations. A complete list of functions can be seen below:

In common.cpp:
display_corners
getRoverView

In rovloc.cpp:
init
getLoc
saveToFile
getImage
doDisplay
clearMemory
setGlobalExtrinsics

saveImage
runStandard
setupRovLoc

The global variables that were to be accessed by the camera tracking system was put under mutex lock protection to prevent simultaneous read and write. These included the rotation matrix and position vector of the extrinsic parameters as well as a flag, *locateFailFlag_g*, dictating whether the checkerboard has been successfully located in the current image.

4.2 Acquiring Images

The images were acquired from on one of the firewire cameras mounted on the rover, as mentioned in section 2.2. The software that existed on the rover for this purpose was based on the TLIB library, which utilizes the *tlImage* image format for storing pixel data and overhead. The library used for the computer vision on this project was the OpenCV library mentioned earlier, which uses the *IplImage* format. Thus there was a need to convert between the two image formats. This was implemented in the *getRoverView* function, which takes an 8 bit grayscale *tlImage* and copies each pixel to the output *IplImage.* This image is declared as a global variable to enable

4.3 Locating Corners

Several methods of detecting the features of an object exist. One way of addressing it is to try and recognize brightly colored regions on the object, but this has some limitations with regard to lighting conditions. OpenCV has a function for attempting to recognize points in a black and white checkerboard, which is then further refined by looking at the intersection of the vectors created by the image gradients in the black-white regions of the board. Figure 2 shows the corners located on a checkerboard, taken with the rover camera and processed by code developed by the author. The annotation of the checkerboard frame used in the extrinsic parameters has been superimposed on the image.

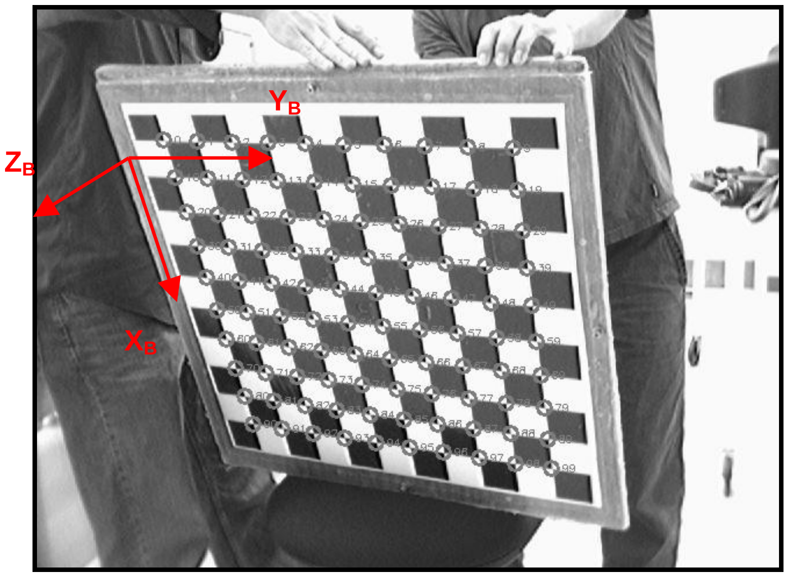

Figure 2 Corners located on checkerboard

4.4 *Calculating Extrinsics*

The calculation of the extrinsics was performed using functions from the OpencV library. The functions require the location of all the points located in the image frame, the intrinsic parameters found and the physical location of the points. The latter describes where all the points are situated with respect to the checkerboard frame. Non-planar surfaces can be used, and the code would work with point recognition methods that do not necessarily use a checkerboard. The output of the algorithm is a position vector and rotation vector, which was then converted to a translation (rotation matrix and position vector), which is simpler to work with for the absolute positioning of the rover in the world frame.

4.5 *Interfacing with Camera Tracking System*

The overall project was in the end to consist of two main components, the visual localizer and a camera tracking system. A main program (*main.cpp*) was set up to declare the two threads, and provide user interaction. It was decided to run the visual localization system thread as often as permissible, as this was thought to be the main limitation on the systems ability to rapidly respond to changes in the position of the checkerboard or rover. The timing and execution of the implemented camera tracking system is detailed in Gregorio Monge's report.

Capabilities were also developed by the author to enable visual feedback and to simplify debugging of the code developed. One function, *display_corners*, takes the current image and the corners located on it and for each point displays a circle and the number associated with it. This image can then be displayed in a window on the screen. The final version of the software includes user selection of two modes of operation, which is passed as input variables to the visual localization thread:

- Nominal operation
    - Only verbal messages on screen
- Debug mode
    - Image of located corners displayed in run-time
    - Global parameters are not updated until the user responds

5 Testing

5.1 *Introduction*

The main testing performed for this project was performed to get a better picture of the capabilities of a visual localization system using a checkerboard. This section describes the testing performed with this in mind and the setup used as well as the initial testing performed of the system integrated with a camera tracking system on the rover.

5.2 *Setup*

The initial testing was performed using manual measurements, on a relatively large checkerboard, which allowed for realistic distances to be tested. However, as the board used had a plastic front plate, this setup produced too much reflections from the overhead lighting, and satisfactory results could only be achieved with a lowered light intensity in the room, which reduced the quality of the images acquired. The tests did quickly show however, that the vision system was more accurate than could be accurately measured manually in this setup.

Figure 3 Initial manual testing

A more structured test setup was desirable to minimize any errors in the measurements. For this task it was decided to make use of an optical bench, which provided high accuracy and could be used to set some constraints on rotations. It also makes it easy to integrate and hold test equipment in a certain position with brackets. The optical table can be seen in Figure 4, which also shows the FARO arm used, mentioned in section 2.2.

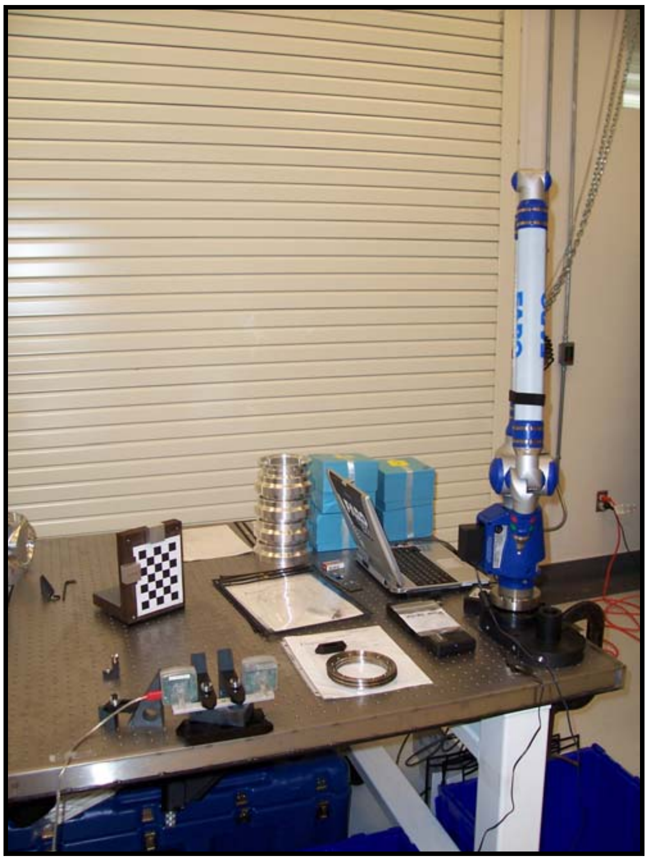

Figure 4 General test setup

The checkerboard itself was printed on a regular laser printer, and was initially mounted on a plastic mounting previously used for calibration in the SSL, as can be seen in Figure 5. this setup proved to have inaccuracies, however, and did not provide a flat surface for the FARO arm measurements. A new mount was used to give better geometric accuracy to the checkerboard, based on a machining mount, which can be seen in Figure 6.

The two metal sections mounted on the stand were used to define two of three orthogonal planes that define the origin of the checkerboard. The third plane is the checkerboard itself, which also serves as the basis for the normal vector measured separately. The camera stand used can be seen in Figure 7. This was designed to make sure the left camera would stay completely still during testing, by the use of special brackets, which is bolted to the optical table. The checkerboard used was 6 high and 4 corners wide, with 32.9 mm checkers.

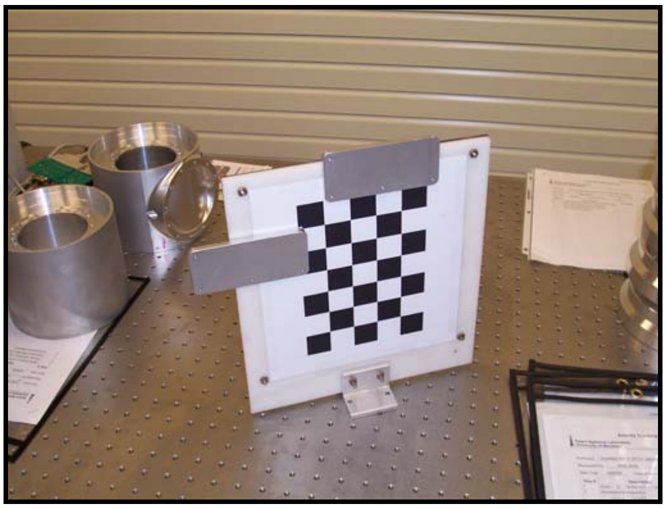

Figure 5 Initial checkerboard setup

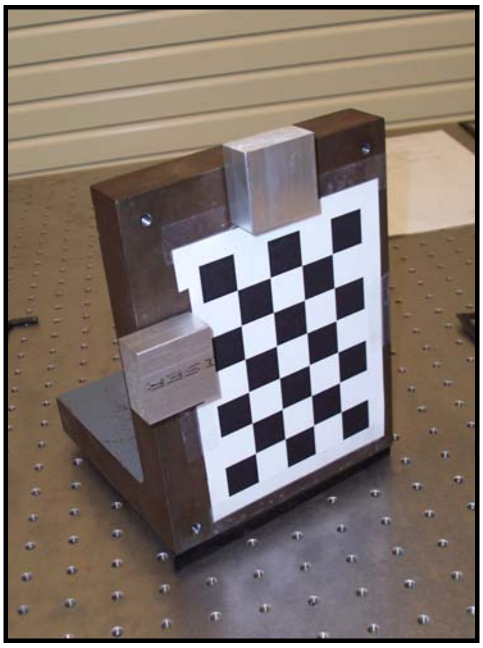

Figure 6 Final checkerboard test setup

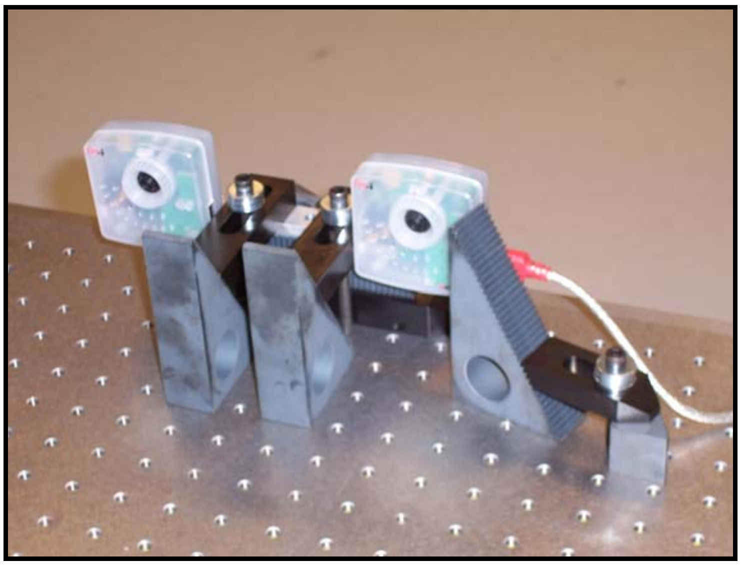

Figure 7 Camera test setup

To be able to use the FARO arm effectively, however, the coordinate system of the camera would have to be accurately represented, to enable all points to be represented in this frame. This way the data received when taking measurements on the checkerboard directly matched the data coming form the visual positioning system. However, as the camera used is encased in a plastic casing, only coordinates based on this casing was possible. The optical axis was defined by taking measurements around the circular lens casing (when fully secured), and then the origin of the measured frame, was positioned and oriented based on symmetrical features on the camera housing. This was the frame all the raw data from the FARO arm was represented in.

To find the true camera coordinate frame however, the specifications of the camera was used to determine that an offset of approximately 3.15 mm existed, as is displayed in Figure 8. Of this the distance from the measured coordinate system to the back of camera circuit board was 5 mm, which is 1.7 mm thick and the effective distance from the back of the CCD sensor to the effective image area is given as 1.41 mm [3]. However, the focal length of the camera, calculated to 4.96 mm (870 pixels of 5.7 micrometer), offsets some of this.

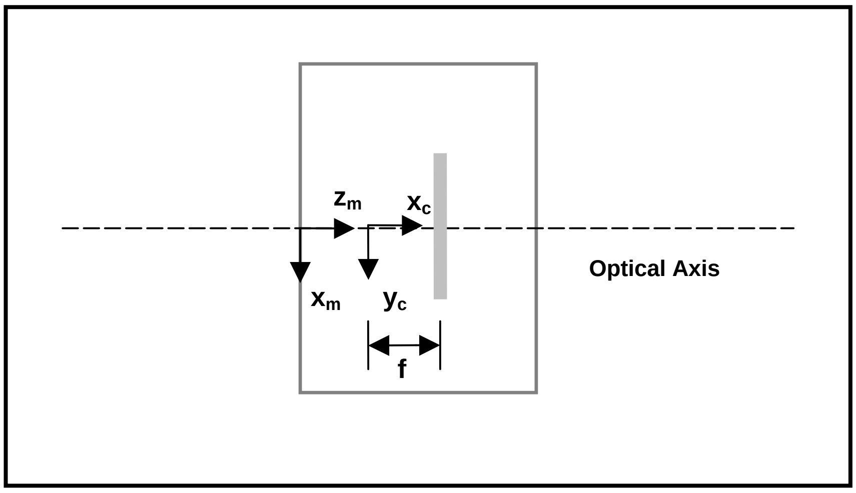

Figure 8 Measured and camera frame

5.3 Test Parameters

As mentioned before the data stored in the tests performed included:

- From the visual localization estimates
  - Rotation matrix and position vector, which together make up the complete translation ${}^{C}_{B}T$
- From the FARO arm measurements
  - A measured position vector
  - A measured checkerboard normal vector

The FARO arm data is fully defined with the assumption that the checkerboard is not rotating about the normal vector. This is a reasonable assumption due to the high accuracy of the optical bench used and the constant checkerboard mount. However, it might have introduced a small constant error with respect to the actual 3 axis orientation of the checkerboard. And it was found to be hard to measure any fixed rotation of the checkerboard when the checkerboard was mounted. A satisfactory method for comparing the estimated and measured data was needed, and so several test parameters were defined. For the position of the checkerboard in the camera frame, the following parameters were used:

- Distance
  - Magnitude of position vector of checkerboard frame in camera frame coordinates, ${}^{C}p_B$
  - Error defined by the difference between the measured and visually estimated values
  - Would define distance from rover to a fixed checkerboard in normal operation
- Position

- Angle between the measured and visually estimated position vector
- Would define rotation of rover (taking into account camera rotation) with respect to a fixed checkerboard in normal operation

Similarly for the rotation of the checkerboard with respect to camera frame the angle between the measured and visually estimated normal vector of the checkerboard was used. Here it was assumed that the change in rotation about this vector was not significant due to setup described above. This would define the angle position of the rover with respect to a fixed checkerboard in normal operation

5.4 Results

5.4.1 Effect of distance

To assess the accuracy of the vision system at distance, several data points within the possible range of the current test setup was taken. All had the same angle, 41.6°, and were along an axis parallel to the optical axis (z direction of the camera frame). The lens was kept constant, and the distance was varied from the feasible 340 to 1080 mm. Figure 9 shows the corners located in these to extreme positions. The results can be seen in Figure 10, Figure 11, Figure 12.

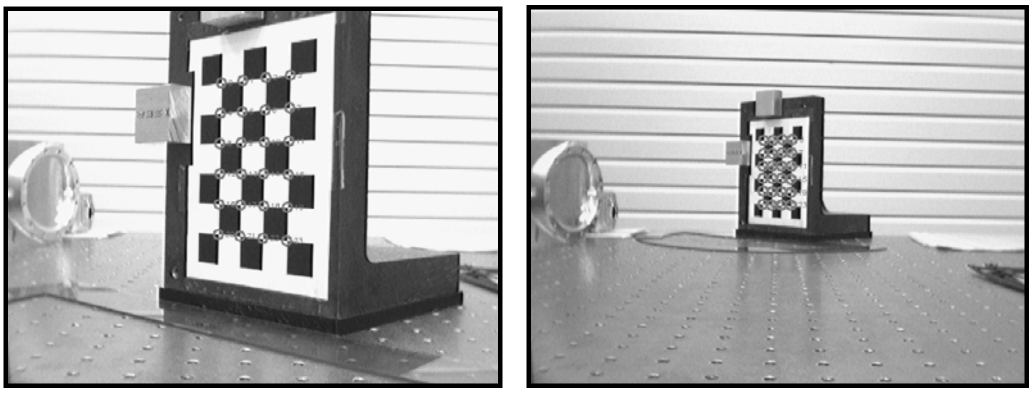

Figure 9 Checkerboard corners recognized at range of distances

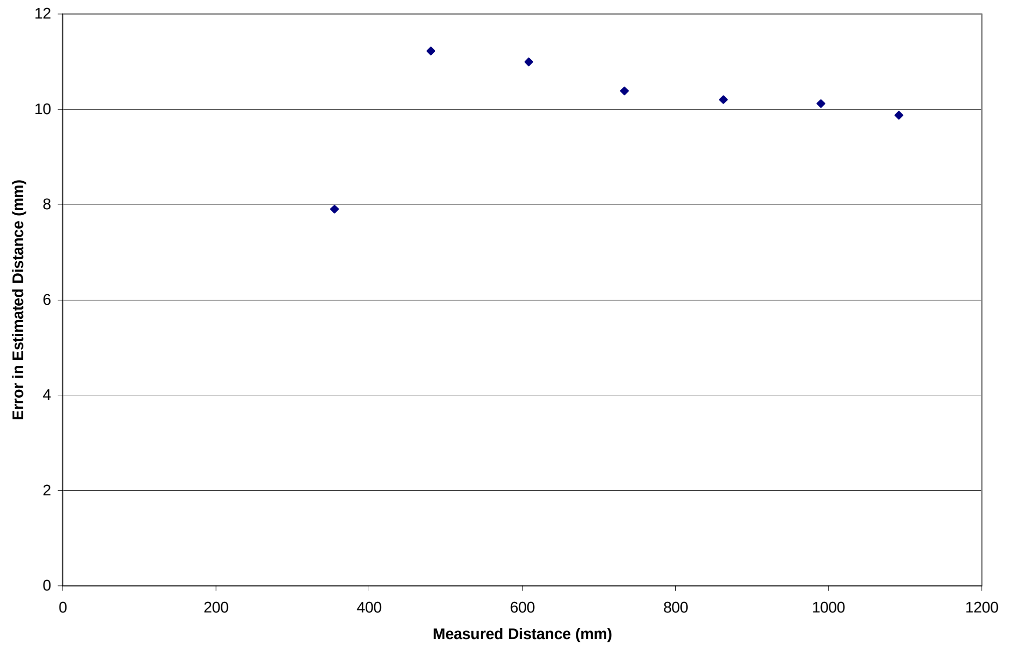

Figure 10 Effect of distance on estimated distance error

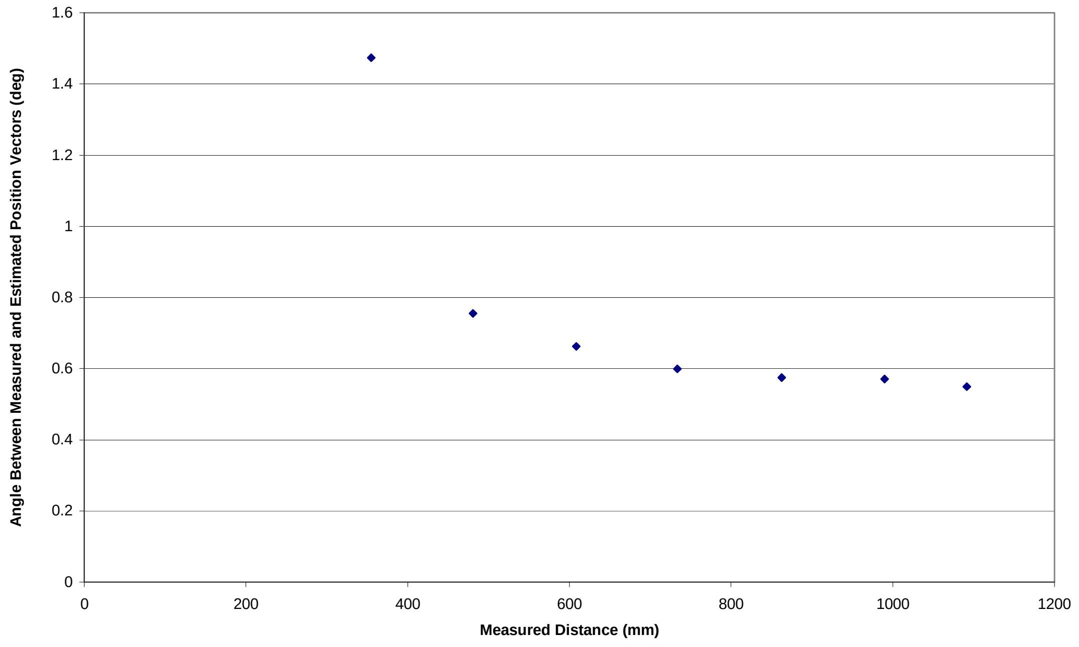

Figure 11 Effect of distance on error on positioning vector angle

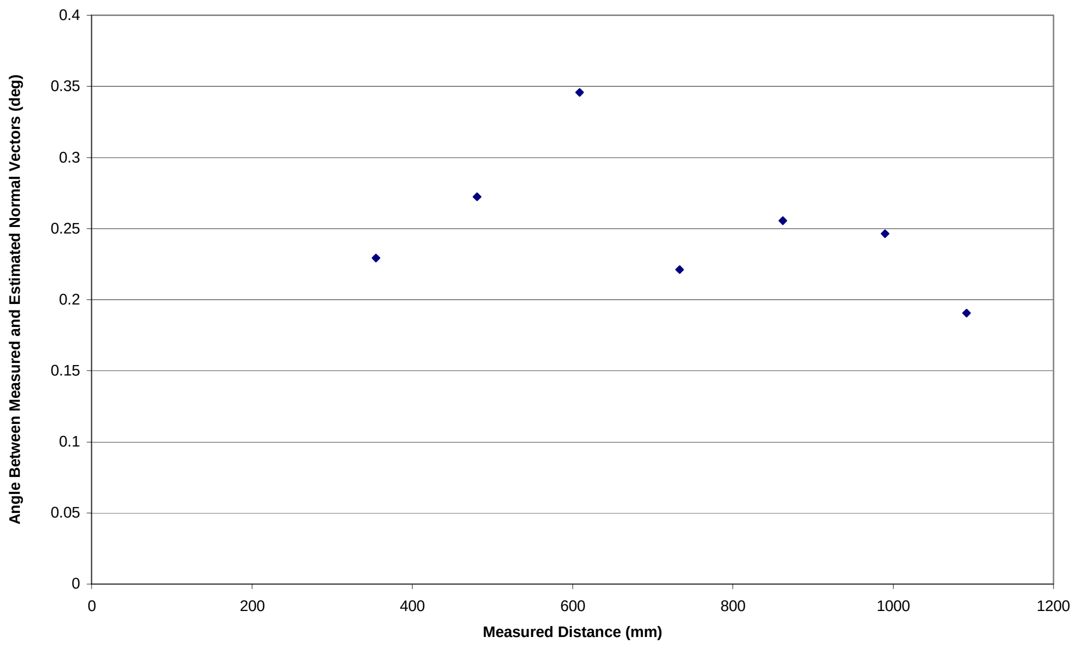

Figure 12 Effect of distance on error on normal vector angle

5.4.2 Effect of Angle

Another interesting issue was whether the relative angle of the checkerboard (angle between checkerboard and image plane) with respect to the camera would affect the accuracy of the system. The tests were performed by keeping the vertical centerline of the checkerboard at the same distance away from the camera, approximately 0.5 m. The angle was then varied, while making sure to keep the lens constant. The results can be seen in Figure 13, Figure 15 and Figure 14.

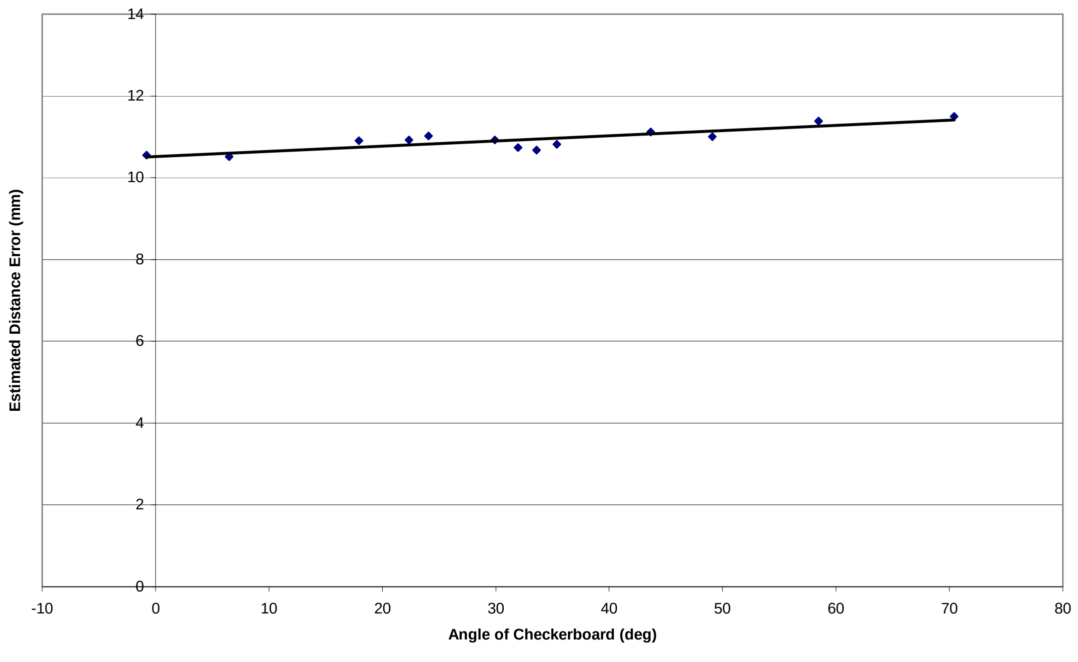

Figure 13 Effect of relative angle on distance error

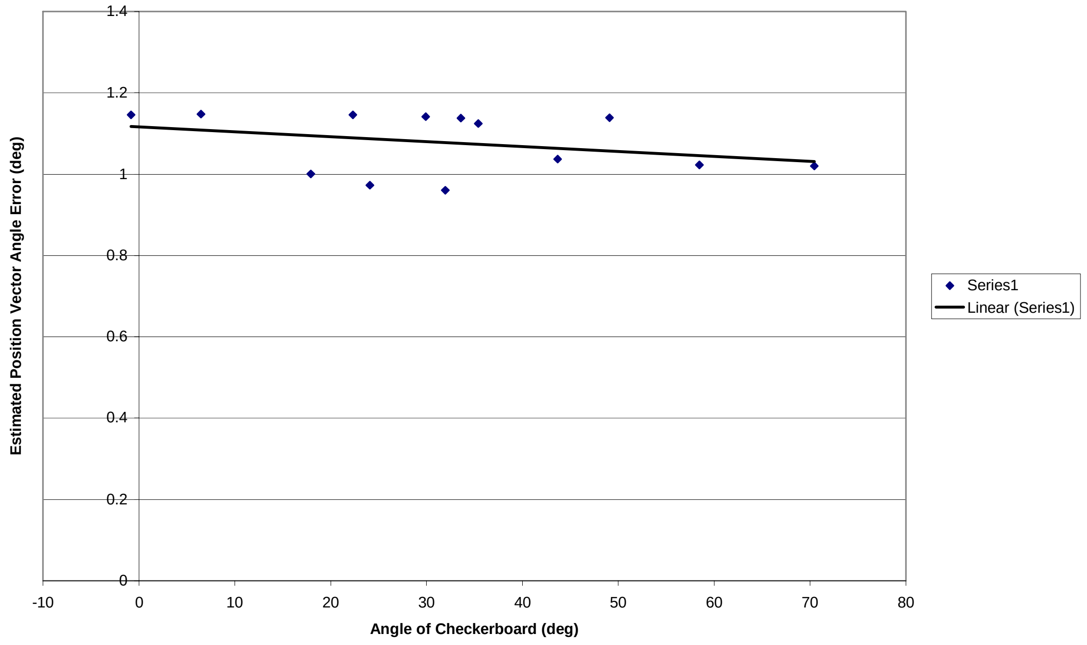

Figure 14 Effect of relative angle on the position vector angle error

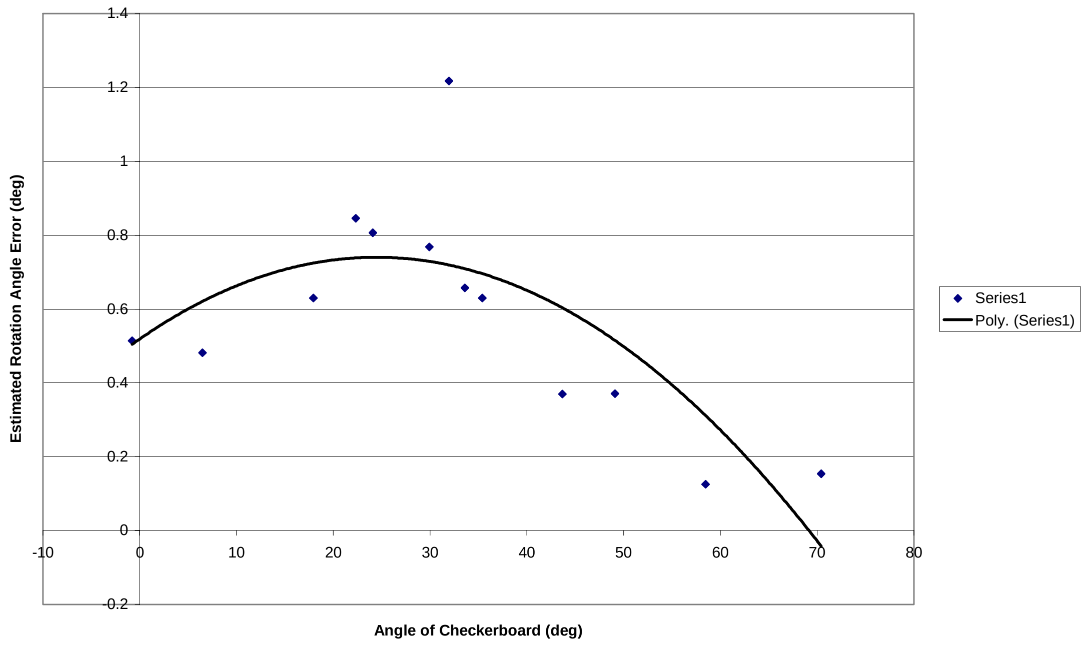

Figure 15 Effect of relative angle on rotation angle error

5.4.3 Lens effects

As the lens was found to have a considerable pointing tolerance, and could potentially shift during use, it was decided to determine how this would affect the performance of the system. Tests where performed where the lens was forced to extreme left and right positions, while the focus was kept constant for acceptable images in the 0.5 to 1 m range. Then the distance to the checkerboard was varied, moved along line parallel to the optical axis, and the angle kept constant (at approximately 19˚). The results can be seen in Figure 16.

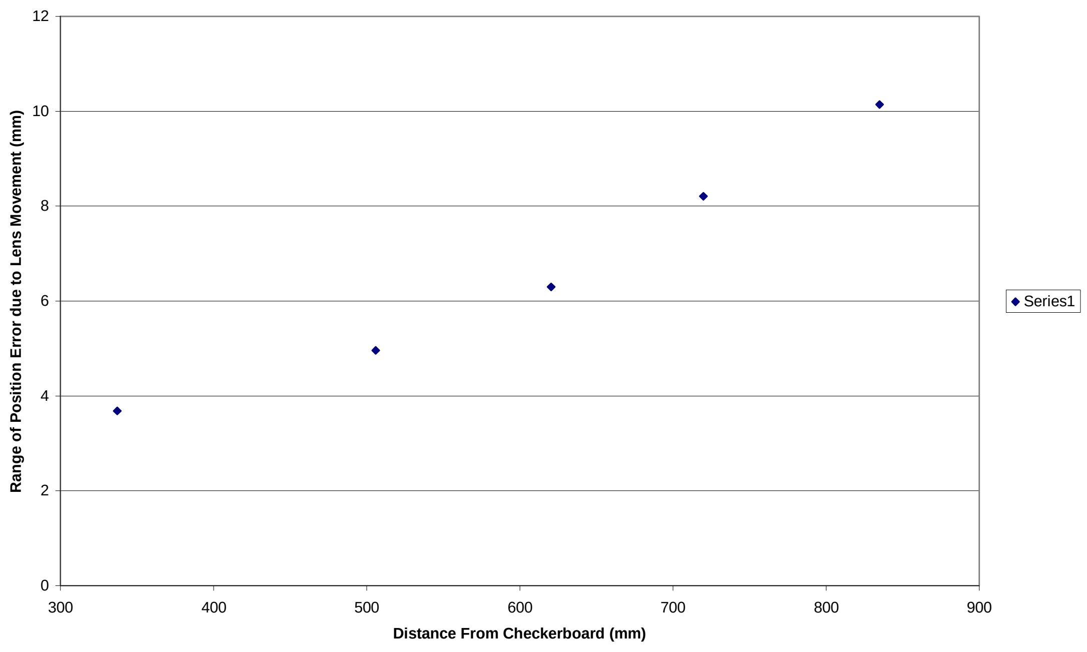

Figure 16 Range of position error due to lens movement

5.4.4 Effect of Number of Points Used

It was attempted to assess whether the number of points used would have an effect on the accuracy of the system. The checkerboard was kept at a constant distance and angle relative to camera (approximately 0.5 m and 41.5˚). The number of checkers visible was then limited by covering up sections of the existing checkerboard, as can be seen in Figure 17. Thus the testing was limited by the limited space for a checker board with the test setup used. And reducing the size of the checkers themselves would also be problematic, as when the checkers become smaller, they are much more difficult to get accurate physical size measurements from. The setup also becomes more affected by accuracy and resolution of printing device used.

The maximum number of corners in the checkerboards used was limited to 5 by 4 and 4 by 3, and produced very mixed results. There was very little effect on the direction of the positioning vector, on the order of 0.002˚. Some variation in the magnitude of the positioning vector was seen, but limited to about 0.7 mm with the current selection of points. It was difficult to detect any noticeable trends, and more data points for larger number of corners would be needed to draw any conclusions. It can be expected however, that the accuracy of the system will increase with more points.

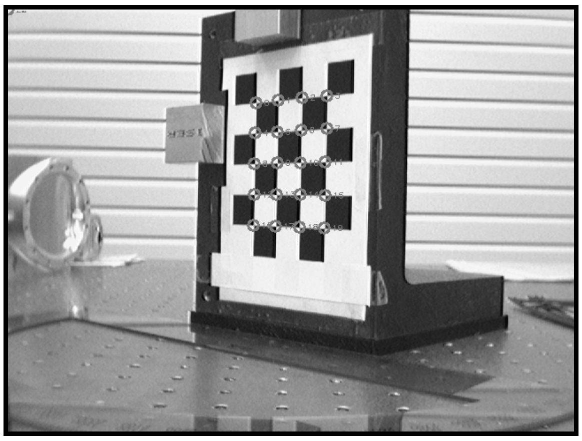
Figure 17 Example of 4 by 3 Checkerboard

5.4.5 Effect of Checker Size Inaccuracy

The author also wanted to investigate the effect of slight inaccuracies in the physical measurements of the checkerboard used, to assess whether this could be a contributing factor to the accuracy of the tests performed. This was done by introducing an offset of the accurate measurements by up to 0.3 mm in positive and negative direction, out of 32.9 mm checkers. These were then used by the algorithm on a fixed checkerboard, where position, angle and lens were kept constant during testing (approx. 0.5 m away, 18.0˚). The results can be seen in Figure 18.

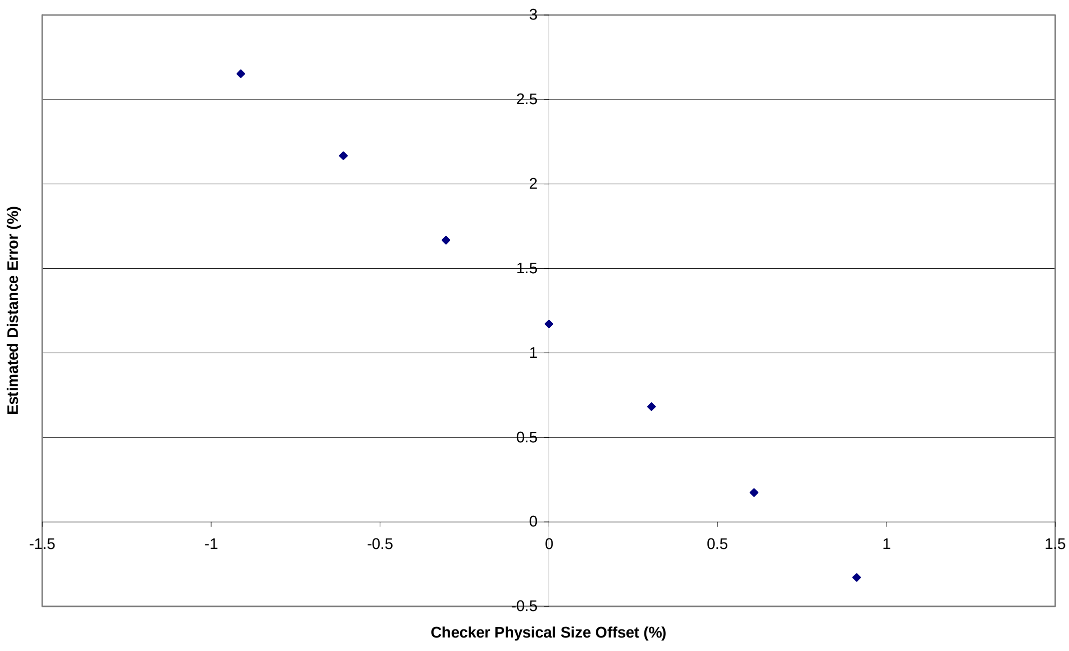

Figure 18 Effect of physical checker size inaccuracy at 18˚

5.4.6 Rover Integration

In addition to the characterization of the visual localization system, integrated testing was performed with Gregorio Monge and his camera object tracking system on the rover itself. First the camera system was dismounted form the rover, as seen in

Figure 19. In this configuration the camera could be moved relative to the checkerboard and the response be seen from the servos, with no risk of positive feedback during debugging.

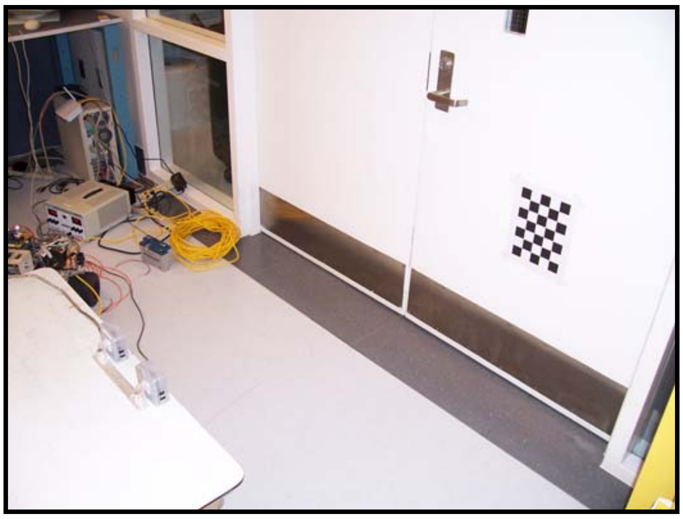

Figure 19 Initial rover servo/vision testing

Once the system was confirmed to be operating satisfactory, the cameras were mounted back on the servo stand on the rover, and full testing of the system could be performed. The tests proved early on the capabilities of the vision system in recognizing the checkerboard at varying distances and angles, and also to provide accurate feedback on its location. The update rate of the vision system was fast enough to enable accurate and responsive tracking, and Gregorio Monge's control system worked very well. More detailed info on the tests performed can be found in the before mentioned authors report. Some issues did arise however, regarding the suitability of the current camera setup to accurately position the rover. Especially the current inability to lock the cameras in on a set position with respect to the servo stand, as well as the accuracy of the stand itself may prove to degrade the performance of the system.

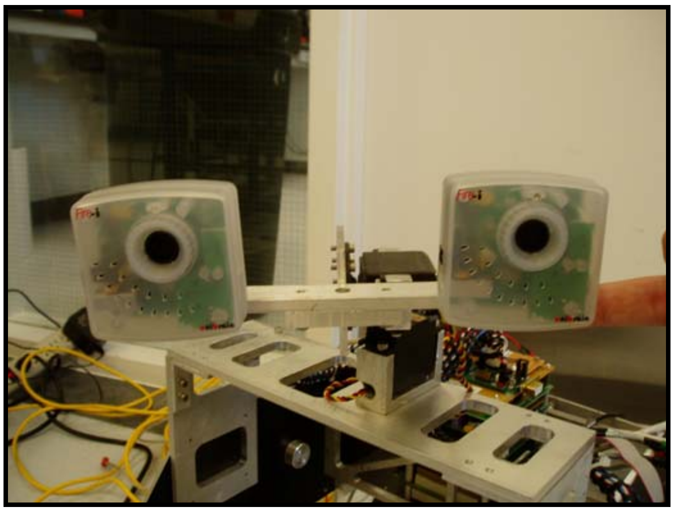

Figure 20 Integrated rover servo/vision testing

5.5 Discussion

An initial goal of the testing performed was to find the minimum allowable pixel distance between points on an image of the checkerboard for positive recognition. This would allow for the specification of the size of each checker in a checkerboard needed for a given camera system at a certain distance and angle. This again could be related to the size of the overall checkerboard for a given number of checkers, which would depend on number of points needed for the desired accuracy. This was determined to not be feasible with current test setup, which had a operating range of only approx.1 m. One could reduce the checker size, but inaccuracies in the physical measurements, and limitations of printing devices, would make this impractical.

The effect of distance gave mixed results. While the distance error was reasonably constant as the checkerboard moved back, the rotation of the checkerboard gave more spread in the results, albeit both gave slight trends towards reducing errors with distance. This was surprising, but was further confirmed by the results for the angle error of the position vector defining the checkerboard. The general negative trend could be due to image distortions that affect a close up more than the smaller image projection of the furthest position, or the fact that the position vector angle is easier to accurately estimate if the start and end position is further apart. All also showed a deviation from the general trend with the first point at around 350 mm. Whether this is due to errors in the data acquired, or part of a more complex trend is not fully understood, and should be investigated further.

The results from varying the relative angle of the checkerboard with respect to the camera were more clear cut. A consistent but minor increase in the distance error with increase in angle was found and an opposite trend for the angle error of the position vector exists. The more interesting result here came with the error in the normal vector of the checkerboard, defining the orientation of the board with respect to the camera. As can be seen from Figure 15, the error seem to maximize in the region between 20˚ and 35˚, while it approaches zero as the angle approaches 70˚ to 80˚. It could be that the added perspective of viewing the checkerboard almost sideways increases the ability of the algorithm to accurately define the rotation of the board. The trendline added is merely suggestive however, and more data points should be gathered before making a final conclusion on this.

From Figure 16 it can be seen that the inaccuracies of the lens does affect the ability of the camera to accurately establish the position (x coordinate assessed for left and right centered lens) of the checkerboard. As the distance increases the trend is towards a larger range of deviation from the measured position. This brings up the matter of camera calibration. The sideways motion may also alter the intrinsic parameters slightly, but probably only the image centre coordinates. This should be assessed by calibrating camera at extreme sideways travel of lens, and comparing results.

The tests performed used the initial intrinsic parameters gathered during earlier testing as described in section 4. However, the focal length could also have been slightly different in the tests performed later due to a different focus setting, and there is no way of assessing the current focus setting of the camera used. Also, the lens had to be fastened when doing initial setup of the FARO arm to accurately determine the optical axis, which further complicates the procedure. And although tests performed at similar distances so should be reasonably accurate, ideally the camera should be recalibrated every time the focus is changed. Another method is to use a camera where the focus is set to infinity and to set the aperture small enough to image the closest objects desired.

6 Conclusion

A system for allowing a rover to accurately position itself with respect to a static checkerboard using a vision system on a 2 DOF stand has been developed. The vision localization system has been designed to run real-time on the rover camera, and to provide accurate positioning feedback to a camera tracking system for keeping the checkerboard in sight and for estimating the position of the rover. The system has been tested using very accurate measuring tools, and been found to perform well. The algorithm was found to be able to estimate the distance to the checkerboard within about 10 mm at over 1 m distance with the setup used.

Furthermore, the effect of a change in the relative angle of the checkerboard with respect to the camera was found to only significantly affect the error in the rotation of the checkerboard with increasing precision at high angles. Thus the checkerboard should be able to give reasonably accurate localization results from most angles. The poor tolerance of the lens used for the camera was found to introduce added positioning errors however, which would have to be taken into account when using the camera for positioning. Another find was that errors in the estimated distance varied linearly with the accuracy of the measurements for the physical points of the checkerboard.

Finally, the vision localization system was finished early so it could be integrated with a camera object tracking system developed by Gregorio Monge. The combined system was tested on the rover, and performed very well in tracking a moving checkerboard at reasonable rates. A simple positioning system has thus been created, capable of reasonable accuracy and semi-autonomous tracking of the localization target.

References

[1] Openc Source Computer Vision Library [Internet], Intel Corporation, http://www.intel.com/technology/computing/opencv/. Accessed 04/10/06.
[2] Trucco, E. And Verri, A., Introductory Techniques for 3-D Computer Vision. Prentice Hall, New Jersey, 1998.
[3] Fire-i™ Digital Camera [Internet], UniBrain, http://www.unibrain.com/Products/VisionImg/Fire_i_DC.htm. Accessed 05/13/06.
[4] Debevec, P.E., Modeling and Rendering Architecture from Photographs. Ph.D. Thesis, U.C. Berkeley, 1996.
[5] FaroArm - Platinum [Internet], FARP, http://www.faro.com/products/faroarm_platinum.asp. Accessed 05/19/06.

Appendix A - Camera Object Tracking Methods Investigated

Limitations
To positively identify an object, a given number of points must be recognized on the object. For the OpenCV checkerboard functions used here however, all points must be within the field of view of the camera and identified for the object to be discerned from the background. The recognition of object points is thus limited by the size of the board and its distance away from the camera. More specifically, the minimum pixel separation of points must be above a certain threshold for the points to be discerned by the algorithms used. Thus the issue of the minimum number of points in the checkerboard which still gives an accurate estimate of the position is of importance. The angle of the checkerboard relative to the camera is also a contributing factor.

Active Visual Object Tracking
A simple form of active visual object tracking can be performed by looking at how the object points found move in pixel coordinates between frames. By using the midpoint of the cluster of points, and wanting to keep this in the middle of each frame, an error in pixels can be found for each frame. This can then be used for a simple control system that outputs a given angle adjustment. It can only be used for 1 DOF tracking however, most suitably for yaw, as the kinematics of the complete camera stand is not taken into account. Thus any object must be in the plane of the cameras field of view rotating around this axis. However, some issues with regards to the kinematic configuration can be expected, as the camera frame is not situated directly on the rotational axis.

A better solution would be to fully model the camera stand, and thus enable tracking of objects that is not necessarily in the same plane as the rover. The use of the pitch servo of the camera stand means that there is a need for knowledge of the kinematics of the camera stand. The idea would then be to have the error calculated in joint space, and use this in the 2 axis control of the camera rotation. More specifically the difference of the current servo angles and the servo angles if the camera was pointing directly at the object would be the calculated error. To be able to perform this operation, the inverse kinematics must be found.

Inertial Pointing
Another option that would allow for tracking of an object is using inertial pointing. The inertial sensors to achieve this are already on the rover, and will be used for Gregorio Monge’s motion planning. The idea here would be to point the camera in the direction where the object is believed to be situated, based on the motion performed. Whenever the object is lost from view, a search of the visible half-sphere will be performed by the vision system to try and relocate it. This could also be applied to any active tracking systems implemented, and will be the default application of camera servo control, should none of the more advanced options be completed in time.

Appendix B – List of files for vision localization system

List of Files for Vision Localization System

common.cpp
common.h
rovloc.cpp
rovloc.h
calibpoints.cpp
calibpoints.h
calibrationparams.cpp
calibrationparams.h
calibrationstructures.cpp
calibrationstructures.h
calibsetupparams.cpp
calibsetupparams.h
extrinsicparams.cpp
extrinsicparams.h
findcheckerboardcorners.cpp
findcheckerboardcorners.h
findextrinsicparams.cpp
findextrinsicparams.h
physicalcalibpoints.cpp
physicalcalibpoints.h