A REPORT

ON

Automatic Calibration and Boresighting

By

Viraj Prabhu (2011A7PS044P)

AT

Tonbo Imaging, Bangalore

A Practice School II Station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

(January-June, 2015)

A REPORT

ON

Automatic Calibration and Boresighting

By

Viraj Prabhu 2011A7PS044P B.E. (Hons.), Computer Science

Prepared in the partial fulfillment of the
Practice School II Course

AT

Tonbo Imaging, Bangalore

A Practice School II Station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

(January-June, 2015)

Acknowledgements

I would like to thank Dr. Arvind Lakshmikumar, CEO of Tonbo Imaging, for providing me with the great opportunity to intern in his company. I would also like to thank Dr. Sindhu, PS coordinator, for her guidance and advice. I am grateful to Mr. Vishal Dugar, my project mentor for his constant guidance and valuable insights, as well as several employees of Tonbo Imaging who mentored, guided and helped me out on numerous occasions. Finally, I would like to thank Mrs. Rekha Anandrao, my P.S. instructor for all her help.

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI (RAJASTHAN)

Practice School Division

Station:…Tonbo Imaging……**Centre:**…Bangalore……

Duration: …January-June, 2015…… **Date of Start: …….**January 12, 2015……

Date of Submission:….June 04, 2015……..…….

Title of the Project: Automatic Calibration and Boresighting

ID No. / Name(s)/: 2011A7PS044P - Viraj Prabhu
Discipline(s) of the Student(s): Computer Science

Name(s) and: Mr. Vishal Dugar, Member of Technical Staff, Tonbo Imaging
Designation(s) of the expert(s)

Name(s) of the: Mrs. Rekha Anandrao
PS Faculty

Key Words: Camera Calibration, Bore-sighting, Fiducial, Camera Matrix Estimation, DM365, Pan-Tilt Unit

Project Areas: Computer Vision, Image Processing, Embedded Development

Abstract:
Automating camera calibration without compromising accuracy can bypass the need for tedious manual estimation of intrinsic and extrinsic camera parameters. To achieve this objective, a fully automated algorithm for accurate calculation of intrinsic camera parameters using specialized targets and a collimator setup by applying image processing techniques is presented. High calibration accuracies are observed for the computed Field of View, Optical Center and focal length metrics. A secondary system that can calculate the motor position corresponding to a given field of view by using the above algorithm is also presented.
Boresighting is the process of aligning a weapon's muzzle with its camera sight and is a vital step in firearm calibration. An existing boresighting application has been redesigned in a server-client model and implemented on a DM365 media processor, to enable accurate calibration and boresighting with the boresight tool while improving the security and user experience.

Signature(s) of Student(s): Viraj Prabhu
Date : June 4, 2015

Signature of PS Faculty
Date:

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI (RAJASTHAN)

PRACTICE SCHOOL DIVISION

Response Option Sheet

Station: Tonbo Imaging Center: Bangalore

ID No. & Name(s): 2011A7PS044P – Viraj Prabhu

Title of the Project: Automatic Calibration and Boresighting

Usefulness of the project to the on-campus courses of study in various disciplines. Project should be scrutinized keeping in view the following response options. Write Course No. and Course Name against the option under which the project comes.

Refer Bulletin for Course No. and Course Name.

Code No.	Response Option	Course No.(s) & Name
1.	A new course can be designed out of this project.	
2.	The project can help modification of the course content of some of the existing Courses	Image Processing (BITS F446)
3.	The project can be used directly in some of the existing Compulsory Discipline Courses (CDC)/ Discipline Courses Other than Compulsory (DCOC)/ Emerging Area (EA), etc. Courses	
4.	The project can be used in preparatory courses like Analysis and Application Oriented Courses (AAOC)/ Engineering Science (ES)/ Technical Art (TA) and Core Courses.	
5.	This project cannot come under any of the above mentioned options as it relates to the professional work of the host organization.	

Viraj Prabhu
Signature of Student
Date: June 04, 2015

Signature of Faculty
Date:

Table of Contents

1) Introduction

Camera calibration is the process of estimating the parameters of a pinhole camera model approximating the camera that produced a given photograph or video. Usually, the pinhole camera parameters are represented as a combination of *intrinsic* and *extrinsic* parameters – intrinsic parameters comprised of parameters such as the Field of View (FOV), Optical Center (OC), Radial Distortion Coefficient, and Scale Factor, and extrinsic parameters comprising of rotational and translational coefficients.

Whenever a camera preset changes, for example, its zoom value changes, it is essential to compute these parameters to enable important functions such as focusing. A popular approach to this problem employed by popular image processing libraries such as OpenCV [1] looks for a pattern in a set of input images of an object in different orientations and positions and runs a coupled optimization to simultaneously find both the camera matrix and distortion coefficients.

This method is however, rather inaccurate. In military applications, such error is unacceptable which necessitates manual calculations using Pan-Tilt unit (PTU) or other manual methods. To overcome this tedious, time consuming and human error-prone method of calibration, an approach for automatically calibrating cameras has been developed which makes use of the open source April Robotics Toolkit [2] and can estimate the camera parameters automatically and with high accuracy.

Bore-sighting is a method of adjustment to an optical firearm sight or iron sights, to align the firearm barrel and sights. This method is usually used to pre-align the sights, which makes zeroing much faster. A boresight device is used to

accomplish this. It consists of an optical head and a bore-diameter arbor which is inserted into the muzzle of the rifle. The optical head is then attached to the protruding end of the rod. A grid pattern on the optical head is then used to align the sights with the barrel. This process is naturally unfeasible in certain time and security-sensitive applications, such as in a combat zone.

Thus a bore-sighting application has been developed that automates the bore-sighting process. The aim of the project now was to redesign the application and deploy it in a server client model wherein user interactions on the client side are sent as commands and processed on the server side using a DM365 board, which was achieved completely. This has successfully abstracted the complexity of the application away from the user and improved the end-to-end usability, security and portability of the bore-sighting device.

2) Automatic Calibration

Camera calibration is the process of estimating the intrinsic and extrinsic parameters of a pinhole camera model approximating the camera that produced a given photograph or video.

The intrinsic parameters comprise of the following:

A) Camera Matrix, consisting of the camera focal length (f), optical center (C_x, C_y) and optionally, the scale factor (S_x). These are represented most commonly in the form of a 3x3 matrix thus:

$$A = \begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix}$$

B) Distortion Matrix, consisting of the radial and tangential distortion coefficients, most commonly represented by the coefficients of radial distortion and tangential distortion, represented thus:

$$\text{Distortion}_{\text{coefficients}} = (k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3)$$

These are used to undistort the image in the following manner:

$$x_{\text{corrected}} = x + [2p_1xy + p_2(r^2 + 2x^2)]$$
$$y_{\text{corrected}} = y + [p_1(r^2 + 2y^2) + 2p_2xy]$$

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

The extrinsic parameters, required to account for different locations and orientations of the camera in the real world, comprise of the rotation angles and translation components, represented by the rotation angles R_x, R_y, R_z, and the translation component T_x, T_y, T_z.

It is essential to compute camera parameters for various applications. For example, the focal length needs to be estimated for accurate focusing of the camera. Distortion coefficients introduced due to lens defects need to be estimated to undistort images. A popular approach to this problem employed by popular image processing libraries such as OpenCV looks for a pattern in a set of input images of an object in different orientations and positions and runs a coupled optimization to simultaneously find both the camera matrix and distortion coefficients[3]. The April Robotics toolkit also provides an interactive calibration approach known as AprilCal [4].

Such methods are time consuming and error prone. The algorithm developed executes in run time and accurately computes intrinsic camera parameters particularly the field of view, optical center and focal length across a range of cameras, both color and infrared, at various zoom values accurately using a collimator setup and specialized target by applying image processing techniques. A secondary algorithm has also been developed that can conduct an automated search for the motor position of a camera corresponding to a given field of view.

2.1 Creation of Ground Truth

To measure the accuracy of the automatic calibration algorithm, it is essential to have a good ground truth data set. This was achieved by calculating the field of view of the camera at different zoom values using the Pan Tilt unit and using an application that aligns the extremities of the camera's field with the same edge and uses the displacement to compute the field of view.

Another method to calculate the same values was also employed, wherein the actual distance between corners in pixels is used to compute the FOV. Thus ground truth data was obtained to validate the algorithm.

After collecting ground truth and understanding the requirements and scope of the project, the automation was realized by employing software from the April Robotics Toolkit developed at the University of Michigan known as April Tags.

2.2 April Tags

April Tags [5] is a visual fiducial system developed as part of the April Robotics Toolkit at the University of Michigan. The system employs the use of special targets known as April Tags, which can be detected and decoded robustly across a wide range of translation, sizes, rotation and illumination, and can thus be used for calibration. Targets can be created from an ordinary printer, and the AprilTag detection software computes the precise 3D position, orientation, and identity of the tags relative to the camera.

Figure 1

Figure 1 shows various April Tags belonging to different families. In a TagXXhY, the XX stands for the number of bits in the tag and Y is the minimum hamming distance between any two tags from the same family.

The system can be divided into two main parts: Tag detector and Coding System

i) Tag detector - This can further be broken down into four steps:

a) Detecting line segments –
Gradient (magnitude and direction) is computed at every pixel. Then, the pixels are clustered based on the computed gradients after low pass filtering to eliminate noise. Line segments are then fit to each connected component.

b) Quad detection –
A recursive depth first search is applied in a counter clockwise direction along the detected line segments to identify quads (rectangles) in the clustered image. A quad scoring is also performed to identify the best (most likely) quad to contain the tag.

c) Homography and Extrinsics estimation -
This step computes the 3×3 homography matrix [6] that projects 2D points in homogeneous coordinates from the tag's coordinate system (in which [0 0 1] T is at the center of the tag and the tag extends one unit in the x and y directions)

to the 2D image coordinate system. The homography is computed using the Direct Linear Transform (DLT) algorithm [7].

d) Decoding -

The payload of the quad is now decoded. Using the homography matrix computed in the previous step, the tag-relative coordinates of each bit field are translated to the image coordinates, and compared bitwise to the bank of tags for the best match within a threshold. If a match is found within the threshold, that tag id is printed as the tag id, otherwise no match is said to be found.

Figure 2

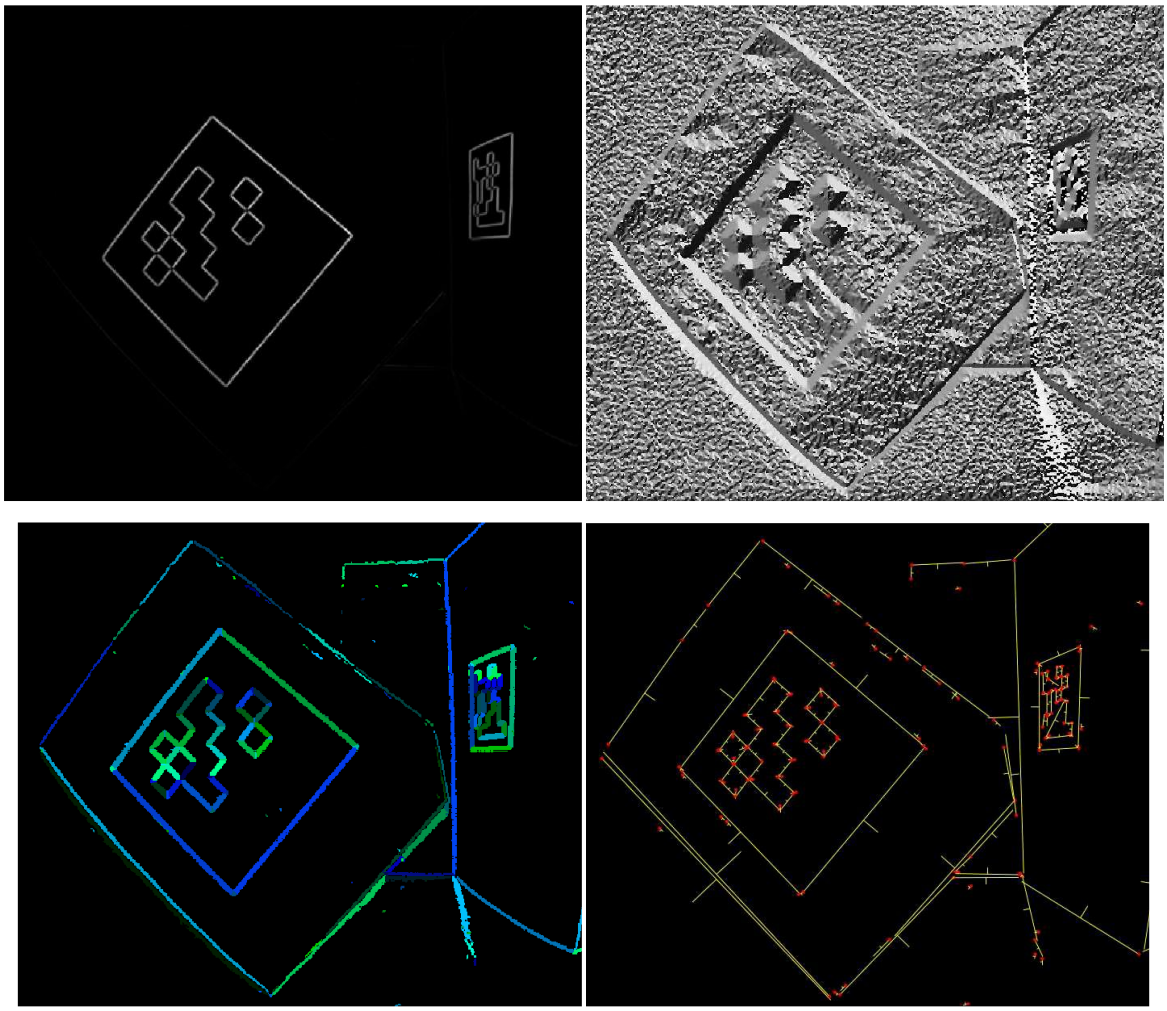

Fig. 2. Early processing steps. The tag detection algorithm begins by computing the gradient at every pixel, computing their magnitudes (first) and direction (second). Using a graph-based method, pixels with similar gradient directions and magnitude are clustered into components (third). Using weighted least squares, a line segment is then fit to the pixels in each component (fourth). The direction of the line segment is determined by the gradient direction, so that segments are dark on the left, light on the right. The direction of the lines are visualized by short perpendicular “notches” at their midpoint; note that these “notches” always point towards the lighter region.

ii) Coding System -

The decoding system described in the preceding section is prone to a high false positive rate. To offset its effects, the coding system has been developed. The goals of the coding system are as follows:

- Maximize the number of distinguishable codes
- Maximize the number of bit errors that can be detected or corrected
- Minimize the false positive/inter-tag confusion rate
- Minimize the total number of bits per tag (and thus the size of the tag)

These goals are often in conflict, and thus involve a tradeoff.

April Tags make use of visual tags known as lexicodes, which are characterized by two parameters - number of bits and minimum hamming distance between any two tags in a tag family. These visual fiducials are therefore a rectangle of different permutations of bits, satisfying the two parameters above. Since they need to be robust to rotation, the April Tags system tried to decode a payload for all four rotations of the tag.

In order to reject patterns that are too simple to reduce the false positive rate, the system tries to reach a target number of rectangles by adding the one that reduces the error the most, using a simple greedy approach. These complex patterns have been experimentally verified to be uncommon in nature and thus substantially reduce the false positive rate. Additionally, code words are tested in an order that maximizes entropy at each bit – this further reduces the false positive rate.

2.3 Implementation

The April Tags system is set up inside Cygwin on Windows, and works with files with a ppm extension. Images of an AprilTags target chosen from the 25h9 family are taken through a collimator-FCB camera apparatus at different zoom values (which are changes by passing commands through a serial communication port called RS232). These images will form the input to AprilTags.

2.4 Proposed Methodology

Setup: The experimental setup consists of the camera directed at the AprilTag target through a collimator such that the tag is visible clearly at maximum magnification. The target is also rotated inside the collimator head to ensure that the edges of the target do not get affine-transformed and thus a homography does not need to be computed – i.e. that ratio of sides remains preserved. Figure 1 shows the target as imaged through the collimator.

2.4.1 FOV Calculation

The application parses settings from an input XML file and images the target at the input zoom positions, by creating serial commands for setting the motor position and focusing the camera using the RS232 protocol. Snapshots are stored with their appropriate zoom label.

A preliminary run of April Tags on the input images showed poor results at low zoom values – this suggested the need for some preprocessing of the images. The preprocessing steps added in MATLAB achieve contrast enhancement,

conversion to black and white, binary thresholding and denoising of the images [8][9]. Figure 4 shows the output of the preprocessing phase.

Figure 3

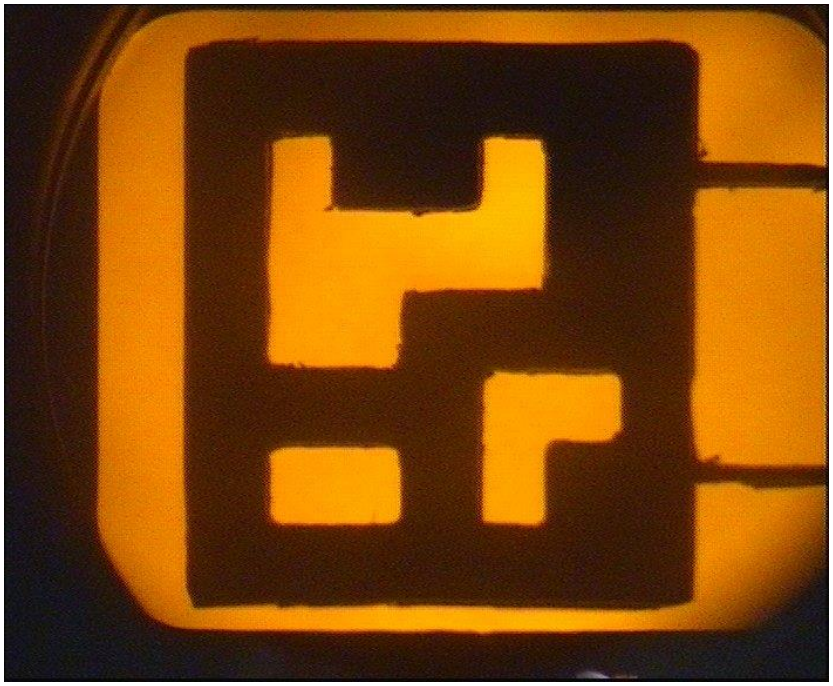

Figure 3 shows the target used in this paper imaged through a collimator at 28x zoom

These preprocessed images are passed to April Tags as input. At high and low zoom values, the images need to be sub scaled and up scaled respectively to enable detection. The former is achieved through passing appropriate flags to the algorithm. The latter is achieved by using the Lanczos rescaling. The algorithm is able to detect and decode the quads for all the images, as shown in Figure 5. The corners detected are further refined [10] and this completes the first phase of automating calibration. The refinement in shown in Figure 6.

To compute the Field of View, a mathematical relation is used that makes use of the length of an edge of the target (known), length of the same edge in pixels and the constant angle subtended by a target per unit length in the collimator.

Figure 4

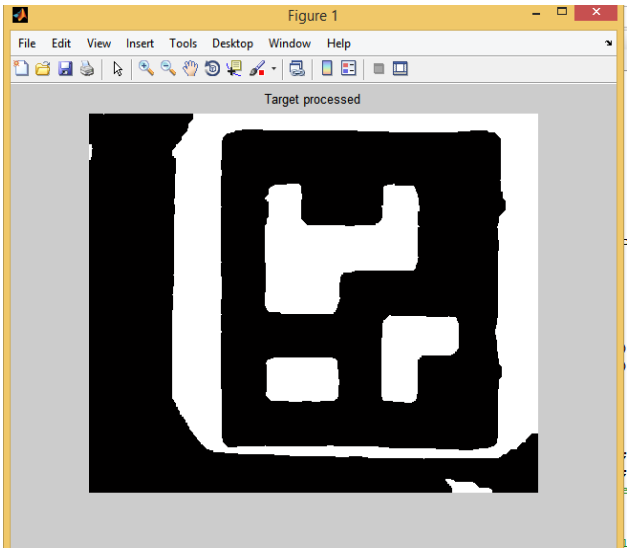

Figure 4 shows the result of preprocessing the target after contrast stretching, denoising and Otsu's thresholding, followed by morphological operations.

Figure 5

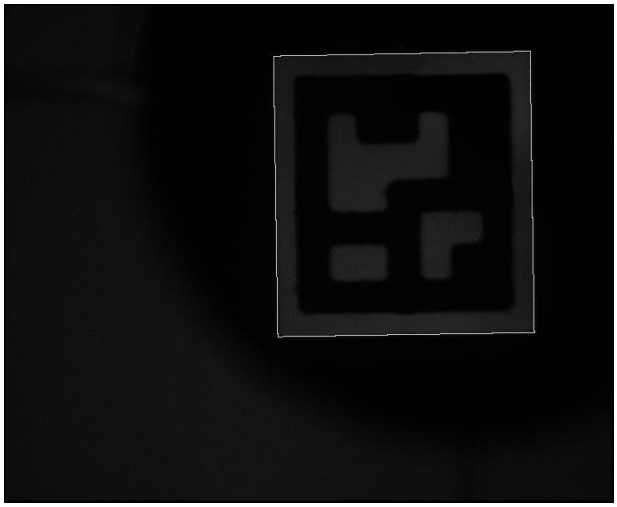

Figure 5 shows the results of the quad detection phase of the AprilTags application on the tag used in this paper

Figure 6

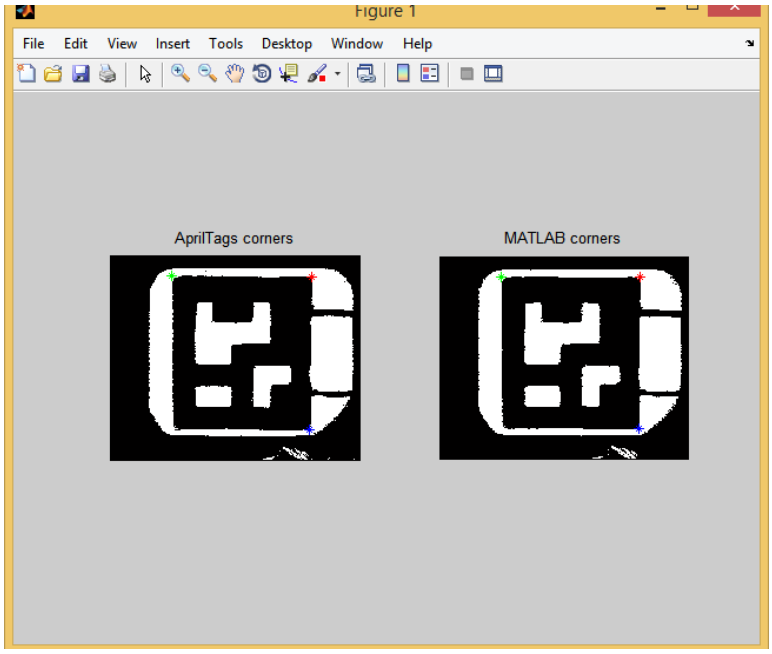

Figure 6 shows the results of the corner refinement stage of the automatic calibration algorithm

2.4.2 Automated Optical Center Calculation

This step calculates the optical center coordinates of the camera lens at its current motor position. The images obtained earlier and the corresponding corner detections are used in this step.

The image coordinates can be approximated to diverge from the optical center along straight lines with increasing magnification – this property is utilized to compute the global optical center of the camera, using the corners detected in the previous algorithm across the range of zoom values.

Additionally, a local optical center is computed for each pair of zoom values to account for lens defects. A coupled optimization is run in a window [11] around the global optical center over all a range of scale factors and optical center coordinates to find the one that best predicts the ground truth corners. Thus the local optical center for each pair of zoom values is computed.

2.4.3 Application: Automated Motor Position Calculation

A search is conducted to find the motor position corresponding to an input field of view using the above system for field of view computation. The search space is reduced repeatedly until the field of view within a chosen tolerance of the target field of view is obtained, and the corresponding motor position is returned as the answer. Checks are added to abandon the search if the target field of view does not lie in the range of the camera's field of view.

2.5 Results and Discussion

The discussed algorithm for accurate computation of intrinsic camera parameters has been implemented and tested and high accuracies very close to ground truth have been observed on the final targets that were manufactured in metal with very accurate dimensions. The algorithm has been tested across a range of cameras, both color and infrared. The editing phase of the algorithm ensures that cases when AprilTags fails to detect the target due to insufficient or inaccurate preprocessing can be resolved with a little user intervention. Apart from this, the entire process is automated and executes in real time.

The secondary system for calculating the motor position corresponding to an input field of view using the FOV calculator to within a user defined degree tolerance has been tested and demonstrates high accuracy.

3) Boresighting

Boresighting is the activity of aligning a firearm muzzle and sight. An application to achieve this has been developed, and my project involves the redesign of the code and its redeployment using a server client model. All the functionality is to be transferred onto the DM365 board [7]. The user will perform interactions on the client side, which are encoded and sent to the board through a serial communication link, and decoded and processed on the board.

To achieve this task, a video precision boresight tool shown in Figure 7 has been created that can align the bore and sighting system of a weapon, for a target at 10m to 100m. It consists of a self-centering mandrel that is secured inside the bore of the gun and a camera head that fits into the mandrel, which can be rotated around the axis of the barrel. Different mandrels are available for various weapon calibers.

The video of the target is transmitted to a computer to check misalignment of the gun with the calculated target line determined by the weapons targeting system. The boresighting is carried on in the following manner. The mandrel is first inserted and secured firmly inside the bore, following which the camera head is inserted into the mandrel and tethered. Two connections to the camera are required by the boresighting application – one to view the target from the camera's video out and one to communicate with the camera via serial commands.

First, the target is placed a distance from the boresight tool and the zoom preset corresponding to the position of the target is selected by the user to be loaded by

the application [12] [13]. The focus can also be toggled between auto and infinity mode.

The baseline distance of the weapon sight system from the muzzle axis along both X and Y axes is entered into the application in meters.

To perform boresighting, the following procedure is carried out:

1) First, the muzzle or target are moved so as to align the ideal sight reticule with the center of the target.
2) The actual sight is adjusted so that its axis aligns with the center of the target as well.
3) Once both the ideal sight reticule and the actual sight axis are aligned with the center of the target, the boresighting is complete.

In case the actual sight reticule is not calibrated, a calibration needs to be performed to recalibrate it. A test can be performed to check if calibration is required, as follows:

1) The target center is aligned with the muzzle reticule by moving either.
2) Then the boresight tool is rotated by 180 degrees. If the reticule has moved, then a calibration is required.

The calibration is achieved in the following way:

1) Once the calibration command is sent, the target/muzzle are adjusted until the target center is roughly in the center of the image.
2) The center of the target is marked as the first point
3) The boresight tool is rotated by approximately 70 degrees, and the center of the target is marked as the next point.
4) This process is repeated to mark the target a total of 5 times
5) Once 5 such points are marked, the new reticule for the muzzle axis is automatically calibrated and a reticule is drawn at its computed position. The corresponding sight reticule is also calculated and drawn on the same image.

The above process is show in the flowchart in Figure 8 below. The calibration workflow above is the main feature that has been implemented on the DM365 processor in this project.

Figure 7

Figure 7 shows the electro-optical video boresight tool for which the boresight application has been developed.

DM365: The TMS320DM365 digital media processor based on DaVinci technology from Texas Instruments enables images at up to 720p H.264 at 30fps in digital video designs with fewer constraints on video format support, constrained network bandwidth and limited system storage capacity. It supports multi-format HD video, provides speeds up to 300 MHz and supports production-qualified H.264, MPEG-4, MPEG-2, MJPEG and VC1/WMV9 codecs.

These codecs are driven from video accelerators offloading compression needs from the ARM core so that developers can utilize the most performance from the ARM for their application. The full suite of codecs supported on the DM365. This ensures interoperability as well as product scalability.

Along with multi-format HD video, the DM365 enables seamless interface to most additional external devices required for video applications. The image sensor supports CCD, CMOS, and various other interfaces such as BT.656, BT1120. It also offers a high level of integration with HD display support including, 3 built-in 10-bit HD Analog Video Digital to Analog Converters (DACs), DDR2/mDDR, Ethernet MAC, USB 2.0, integrated audio, Host Port Interface (HPI) and Analog to Digital Converter[14].

RS232: RS232 is a serial communication standard that formally defines signals connecting between a computer terminal and data communication equipment. It is used commonly in computer serial ports and the characteristics and timing of signals, their meaning and physical size and pin out of connectors are defined in the standard. Though it suffers from low transmission speed, large voltage swings and large size of connectors and has been replaced largely by USB, RS-232 devices are still often used in certain industrial machines, networking equipment and processors such as the DM365.

3.1 Implementation

The overall objective of the application was to extract the essential functionality of the original C++ application and to redesign it in the form of a Finite State Machine in embedded C, having various parameters that transition from one state to another based on user interactions. Figure 1 shows a snapshot of the C++ implementation of the application.

An Interface Design Document was created to standardize the commands used by the application to interact with the DM365. Handlers for each command were added to the command line interface of the application, which in turn are

decoded on the board side and change the state of the system, and trigger appropriate responses.

The RS232 protocol was used to send and receive serial commands to and from the board. On the local machine, serial commands defined in the Interface Design Document are sent by the user over the RS232 link, and are decoded and parsed on the DM365 processor by handlers implemented in software using a switch-case structure in the command line interface of the DM365 software, called u-boot. Checksums are computed on both ends to ensure data integrity. Depending on the command id, the corresponding function is passed the parameters of the command and these values are passed through multiple intermediary methods, from the command line interface to the system server and finally to the AV server which consists of polling threads that run from system startup to finish and control functionality such as the on screen display (OSD), key scan thread, etc. Most of the functionality utilized in this project was implemented in the video software OSD thread.

Figure 8

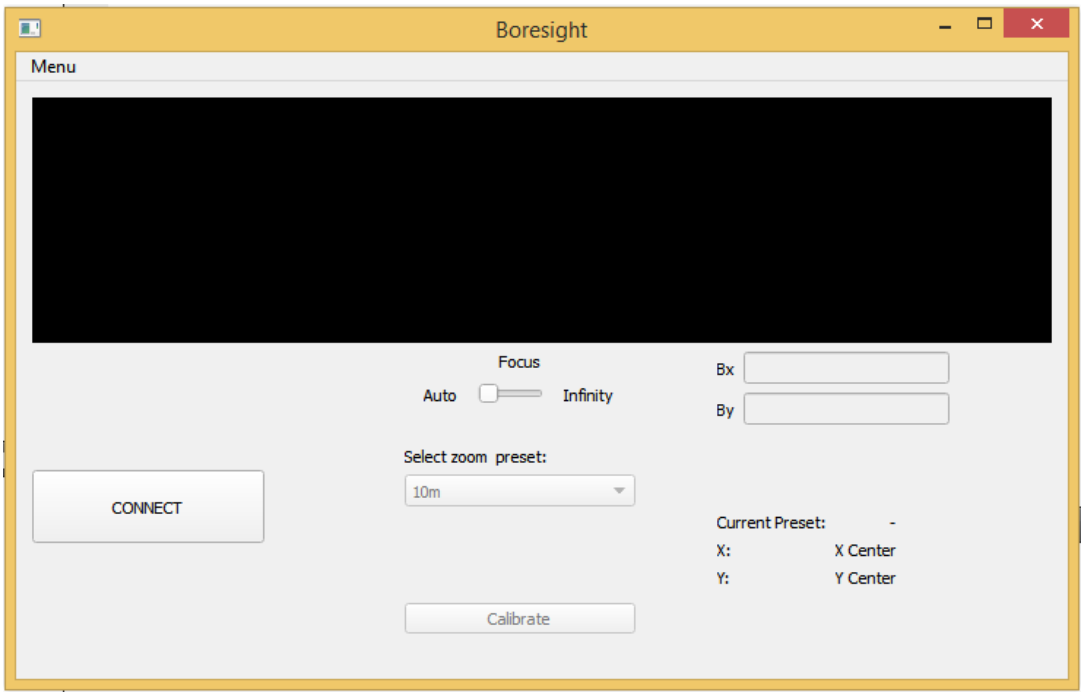

Figure 8 shows the GUI of the original C++ implementation of the boresighting application

Various functions were added to the application. On the board, code to create a new sight reticule was added, along with functionality to toggle the reticule type between a cross and a mil-dot. An enable for the sight reticule was added and a switch to toggle between auto and infinity focus modes on the camera. Additionally, commands were implemented to load presets for focal length, field of view and zoom positions from text files and to set the camera to those values.

The most important feature is the calibration procedure which uses ellipses fit to points provided by the user as well as the parallel displacement between the line of sight of the gun and camera to calculate the muzzle center as well as the sight center. Several commands were added to implement the entire workflow of the calibration, such as selecting points, moving the reticules, setting sight baselines along both axes and performing, saving and exiting the calibration.

After the commands were implemented, extensive testing was done to test the application under various inputs along the entire calibration workflow. Error checks were added to restrict access to certain functionalities during calibration that could cause errors, in accordance with the requirements specified in the IDD. Finally, the code was documented and checked in using git.

3.2 Boresight DLL

A dynamic link library (DLL) is a kind of shared library on Windows platforms which contain code, data and resources. DLL's provide a mechanism for shared code and data, and thus the creator can upgrade the DLL without having to re-link and re-compile the application importing it. DLL's execute in the memory

space of the calling process. Thus dynamic linking provides a way for a process to call functions that are not in its executable code.

DLL's on windows have import libraries with .lib extensions which are linked during compilation and containing compilation information for the contents of the DLL. During runtime, the DLL itself is imported as it contains the implementation of the functions.

The above boresight functionality was also built as a DLL as per client requirements so that they can be easily linked by an application and used for boresighting. The boresight.lib file contains the compilation information and is linked at compile time, while boresight.DLL is loaded and runtime. The header boresight.h exposes the *perform_calibration(int[], int[], int)* method that is passed the x and y coordinates along with the number of points and returns the computed muzzle position to the calling application.

The application loads camera presets from a file, a sample line from which is shown below:

Preset No.	Motor pos.	Hor. FOV(mrad)	Ver. FOV(mrad)	f (px)	mrad/px (X)	mrad/px (Y)
1	10560	103.673	84.823	6784.53	0.147	0.147

3.3 Results and Discussion

The boresight application has been implemented on the DM365 processor and all its functionality has been tested extensively. Error handling and documentation has been added to ensure failure tolerance and code maintainability. The calibration workflow has been tested under a wide range of

inputs and can be used reliably for calibrating the muzzle reticule. The software is now thus more secure and portable as it has been deployed in the desired server client model, and will form a part of the next release of the boresight tool.

4) API for Pan-Tilt Unit

An additional project was assigned to develop an API to control and operate a new Pan-Tilt Unit that is being designed and developed at the organization. An example of a PTU is shown in Figure 9. To this end, a C++ class has been developed inside Qt Creator with functions to operate all the functionalities that the Pan Tilt Unit will have. These include functions to open, close and query the PTU port using the RS-232 protocol, and functionality to toggle stabilization mode. The GUI of the PTU application used to control branded PTU's is shown in Figure 10. The mode of the PTU can also be changed between the following modes:

i) Position mode: In this mode, the PTU is passed pan and tilt angles and velocities. It then pans and tilts to the respective angles at the respective input velocities.
ii) Slewing mode: In the mode, the PTU is passed a slewing velocity and it pans continuously in full circles at this velocity.
iii) Tracking mode: In the mode, the PTU is continuously passed live pan and tilt angles and velocities, which it moves to. This mode is used while tracking a moving target.

The stabilization switch is used to choose whether the PTU should automatically stabilize itself after implementing the command to it. Since the Pan-Tilt unit it still under development, the code is yet to be tested.

Figure 9

Figure 9 shows an image of a Pan Tilt unit. The unit can pan upto 360 degrees at speeds upto 300 degrees/second, and generally are provided with RS232 interfaces.

4.1 Results and Discussion

The C++ API development for the Lorral PTU has been completed and tested on the unit, and all the commands have been found to work as intended.

Figure 10

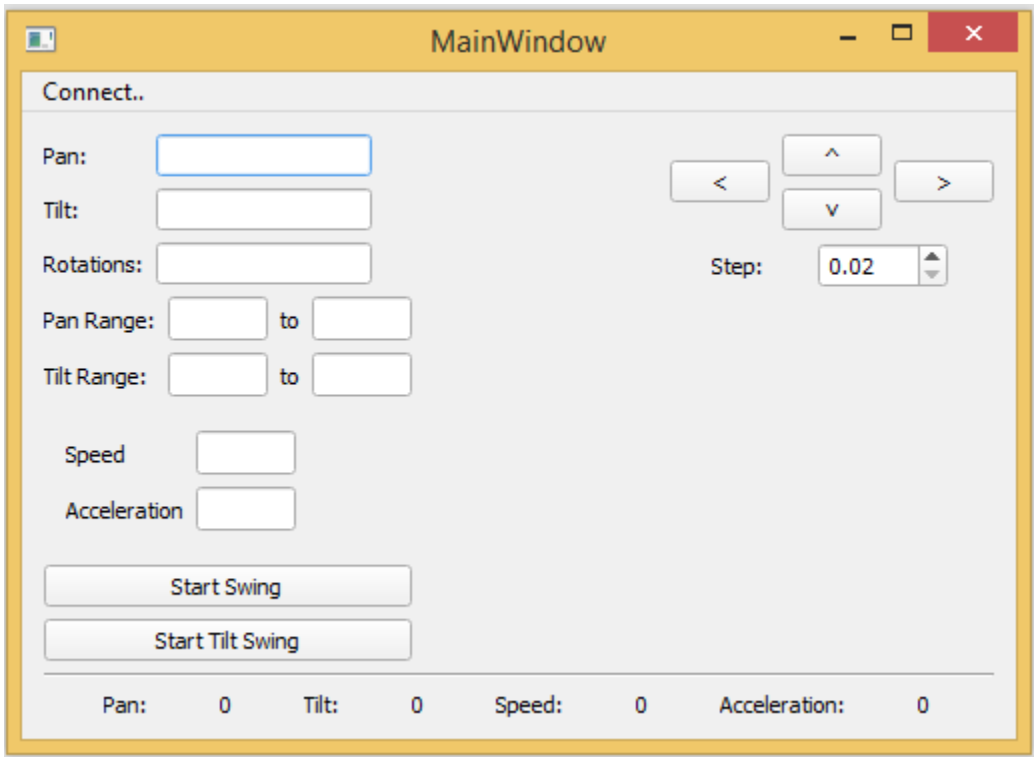

Figure 10 shows the GUI of the original C++ PTU control application

5) Conclusions and Recommendations

Thus, the objective of automatic calibration of a dynamic zoom camera at different presets has been realized – the estimated FOV tallies with the ground truth data within a 1% tolerance. Steps need to be incorporated for making the preprocessing more robust and ensure correct thresholding in a wide range of real world scenarios, so that the input to the April Tags algorithm is noiseless and easy to decode. Additionally, experiments with different dimensions of target need to be conducted to work across the entire range of the camera's zoom values.

Information from the shapes inside the quad containing the tag can also be incorporated somehow to better the estimate – presently the decode step serves as a false positive elimination test and helps to reject spurious quads – it can also possibly be employed to estimate the FOV in the same manner the outer edges are. Once these aspects are incorporated, the result will be a robust, useful and portable application that can greatly bypass tedious manual calculations for calibration without compromising accuracy.

The boresight application has been completed along with all the required functionality described in the Interface Design Document. It can be used to accurately calibrate the camera sight and muzzle and has been tested extensively. Error checks and documentation have been added to provide security and maintainability. Finally, the application is more secure and portable than its predecessor, and thus the project objective has been realized.

The API for the new PTU has also been developed that can be used to operate it and switch between position, slewing and tracking modes and enable stabilization. The API has been tested on the PTU and all commands have been found to work as intended.

6) References

[1] “Camera calibration With OpenCV - The Open CV Online Documentation”, *http://docs.opencv.org/doc/tutorials/calib3d/camera_calibration/camera_calibration.html,* 2014

[2] Olson, Edwin., “The April Robotics Toolkit”, *http://april.eecs.umich.edu/*, 2014

[3] Z. Zhang, "Camera Calibration", Chapter 2, pages 4-43, in G. Medioni and S.B. Kang, eds., *Emerging Topics in Computer Vision*, Prentice Hall Professional Technical Reference, 2004.

[4] A. Richardson, J. Strom, and E. Olson, “AprilCal: Assisted and repeatable camera calibration,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, November 2013.

[5] E. Olson, “AprilTag: A robust and flexible visual fiducial system,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2011, pp. 3400–3407.

[6] P. Ranganathan and E. Olson, “Gaussian process for lens distortion modeling,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2012.

[7] Ranganathan, Pradeep, and Edwin Olson. "Locally-weighted Homographies for Calibration of Imaging Systems." *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014.

[8] J Canny. 1986. A Computational Approach to Edge Detection. *IEEE Trans. Pattern Analysis and Machine Intelligence* 8, 6 (June 1986), 679-698.

[9] Rafael C. Gonzalez and Richard E. Woods. 2006. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[10] Harris, C. and Stephens, M. A combined corner and edge detector. *In Fourth Alvey Vision Conference*, Manchester, UK, pp. 147-151, 1988.

[11] Wilhelm Burger, Mark J. Burge (2009). Principles of digital image processing: core algorithms. *Springer. pp. 231–232.*

[12] FCBEH6500 Technical Manual, *https://pro.sony.com/bbsc/ssr/cat-camerasindustrial/cat-cihighdefinition/product-FCBEH6500/,* 2012.

[13] FCBEX995E Technical Manual, *https://pro.sony.com/bbsc/ssr/product-FCBEX995E/,* 2011.

[14] Texas Instruments Inc., "TMS320DM365 Digital Media System-on-Chip (DMSoC) Datasheet", *SPRS457E–MARCH 2009–REVISED*, June 2011