Stereoscopic Imaging for Slow-Moving Autonomous Vehicle

Project Proposal

By:
Alex Norton

Advisor:
Dr. Huggins

December 15th, 2011

Introduction

The objective of Stereoscopic Imaging for Slow-Moving Autonomous Vehicle, SISAV, is to develop a system that can provide an accurate terrain map to be used for navigating an autonomous vehicle. The system will use two digital cameras to perform the stereoscopic imaging to obtain a terrain map of the objects in front of the cameras. This map will be used by the control algorithm running the autonomous vehicle to determine the direction in which to move. The system will have two modes of operation: calibrate and run. Calibration mode involves calculating the intrinsic and extrinsic parameters of each camera by using a calibration rig with known geometry and easily detectable features. Run mode involves sending a signal to have the cameras acquire images, download and process the images, and then compute distances to objects in the field of view of the cameras. These distances are then used to develop a terrain map for use by the control system of the autonomous vehicle.

Goals

- Learn theory of 3D stereoscopic imaging
- Investigate existing software (OpenCV or MATLAB)
- Control cameras
- Calibrate cameras
- Take and store images
- Process images for objects (e.g. edge detection)
- Correlate objects
- Compute distance to objects
- Compute terrain map

System Description

The system will consist of two digital cameras, a mount for the cameras, and a laptop. The two cameras will be attached to a stable platform that, in turn, will be attached to the vehicle. At the command of the software on the laptop, the cameras will simultaneously acquire images which will be downloaded to the laptop via two USB connections. The software will then identify the pertinent objects via edge detection, correlate the detected edges, and finally compute distances based on the disparity map. These distances will be used to generate a terrain map the vehicle control system can use for navigation

There will be two modes of operation: calibration mode and run mode. During calibration mode, a calibration rig will be used to obtain the extrinsic and intrinsic parameters of each camera[1] which will be used to correct for distortion in images captured during run mode. After the cameras are calibrated, the system will enter run mode, which implements the process of image acquisition and processing for 3D information as described earlier.

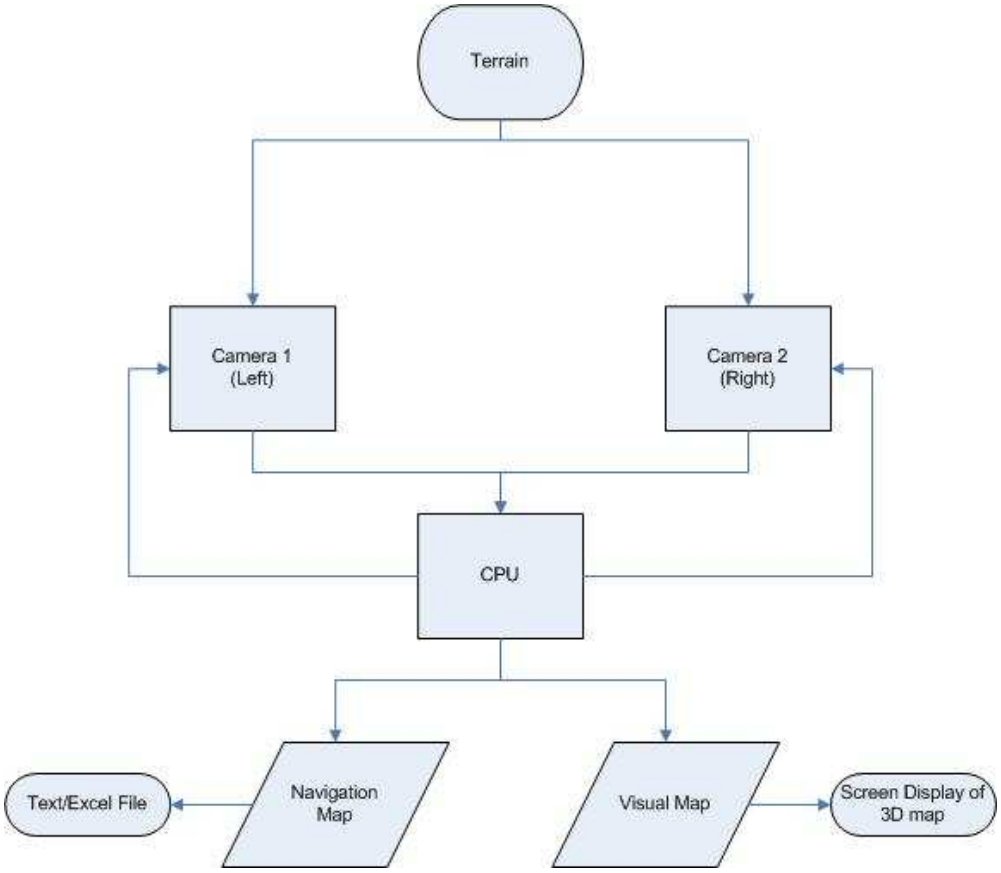

Figure 1: System Block Diagram

As shown in figure 1, the system will be made up of 2 subsystems: the camera subsystem and the laptop subsystem.

Camera Subsystem

The camera subsystem**,** shown in Fig. 2, will consist of two digital cameras mounted on a stable platform. The cameras will convert photons of light into binary data each time they receive signals to capture images. The data will be 8-bit arrays, three from each camera, containing values from 0 to 255 for the RGB values of each pixel. The color information will then be converted to grayscale using a built-in MATLAB or OpenCV function. A value of 0 will correspond to black and a value of 255 will correspond to white. This data will be sent to MATLAB or OpenCV to be processed further. The cameras function in an identical fashion in both the calibration mode and run mode.

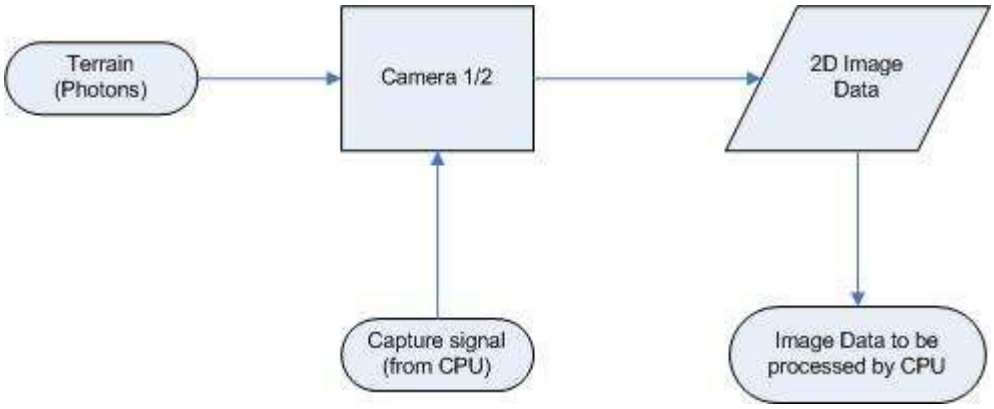

Figure 2: Camera Subsystem

Laptop and Software Subsystems

The laptop subsystem, shown in Fig. 3, will run the necessary software to capture and process the images from the cameras and generate the data for input to the navigation software of an autonomous vehicle. The computer vision software, running on the laptop, will simultaneously acquire images from the cameras which will be downloaded to the laptop via a USB connection. The software will then identify the pertinent objects via edge detection, correlate the detected edges, and finally compute distances based on the disparity map. These distances will be used to generate a terrain map the vehicle control system can use for navigation

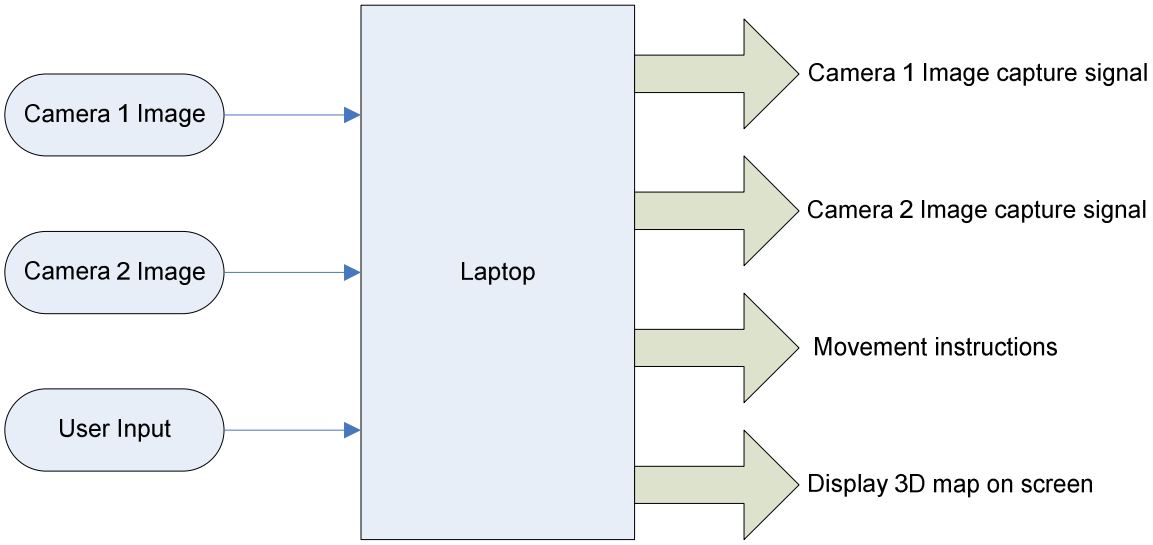

Figure 3: Laptop Subsystem

Operational Modes

There are two modes of operation, calibration mode and run mode. These modes are described in more detail below.

Calibration Mode

Calibration mode will be the initial mode of operation for the system. Its primary function is to correct for deviations of the actual camera system from an ideal pin hole camera system. The deviations are due to internal (e.g. lens distortion) and external properties of the physical cameras. Once the system is powered on, the software will wait for user input to enter calibration mode. In this mode, the software will prompt a user to place a calibration target (e.g. a chessboard target) in a known position and orientation and then the system will acquire an image from each camera. This will be repeated in additional positions and orientation as needed by the calibration software algorithm to be used[2]. Once an appropriate number of images are acquired, the calibration will determine the calibration matrices to be used to correct distortions in the images acquired in run mode. The flow chart for the Calibration mode software is shown in Fig. 4.

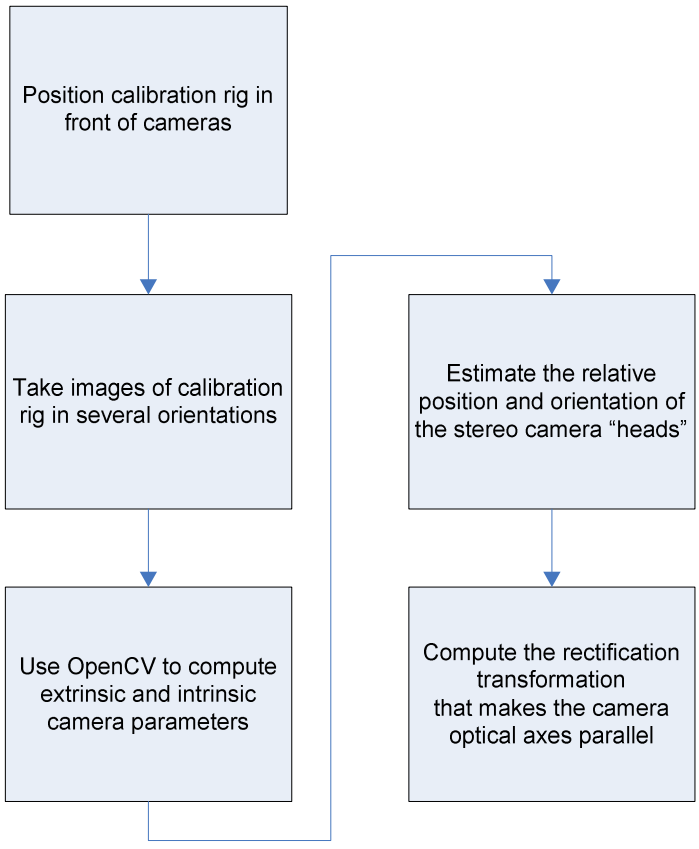

Figure 4: Calibration Mode Flow Chart

Run Mode

Run mode is entered upon successful calibration of the two cameras. In this mode, the user can set up the computer vision software to respond to commands from the user or from an automated prompt from the navigation control software. In both instances, the computer vision software acquires the images from the cameras. The software will then identify the pertinent objects via edge detection, correlate the detected edges, and finally compute distances based on the disparity map. These distances will be used to generate a terrain map the vehicle control system can use for navigation. In addition, the run mode has an option that will allow the distance information to be displayed on the laptop screen for analysis by the user. Run mode is exited by closing the computer vision software. The flow chart for the run mode software is shown below in Fig. 5.

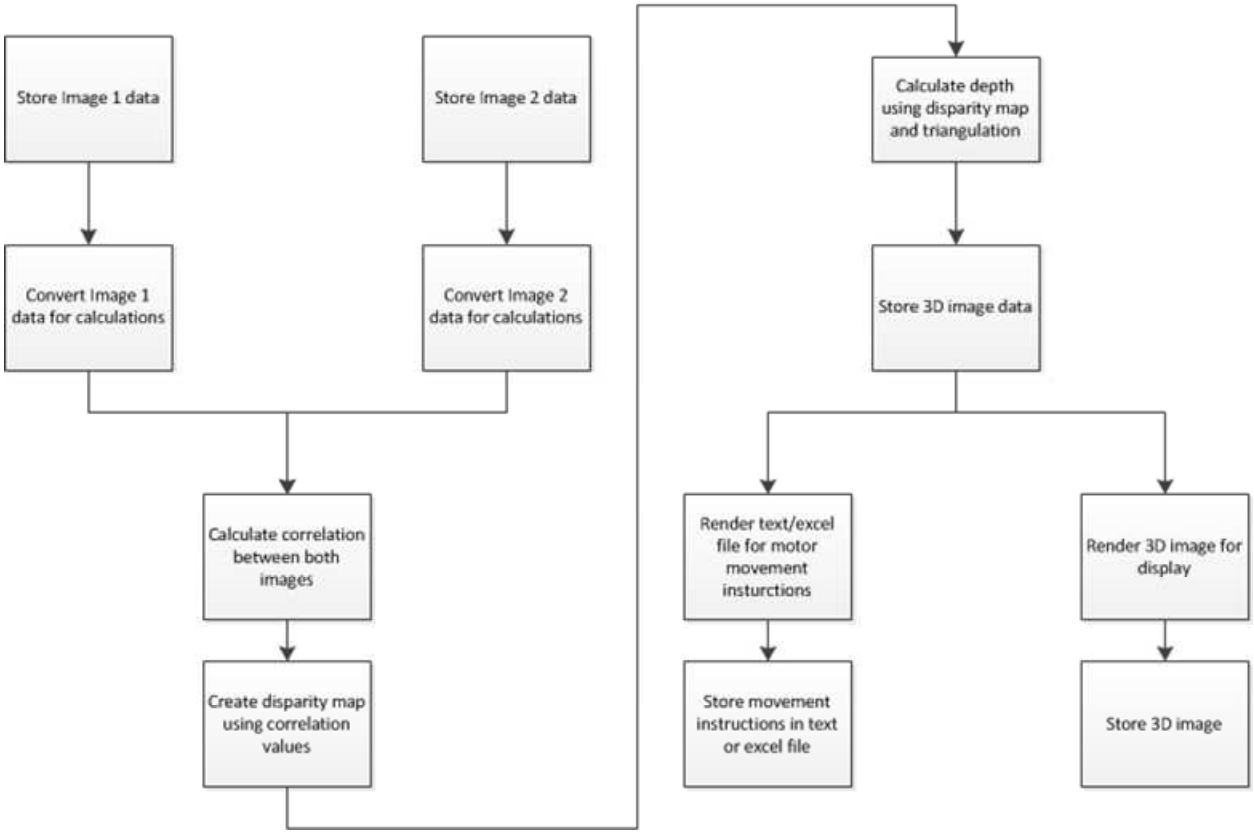

Figure 5: Run Mode Flow Chart

Camera Subsystem Requirements

- The cameras shall have a field of view of at least 45 degrees
- The cameras shall have a depth of view from 1 meter to 10 meters
- The cameras shall be calibrated and focused after startup to ensure accurate image data

- In order to maximize speed for the system, the cameras shall output images at a resolution of 320x240
- The cameras shall be secured to a mount to ensure they do not move out of alignment during operation
- The cameras shall not have any face-tracking software built into them
- The cameras shall interface with the laptop via USB connections
- The cameras shall be compatible with Windows 7.

Laptop Subsystem Requirements

- The laptop shall have at least two USB ports to interface with the two cameras
- In order to run MATLAB, the laptop shall have at least 1024 MB of RAM, 5 GB of memory, 64-bit Windows(Windows 7, Vista, or XP), several TCP ports, and an Intel or AMD x86 processor supporting SSE2 instruction set
- In order to run Microsoft Visual Studio 2010, the laptop shall have2 GB of RAM, 5 GB of memory, 32-bit or 64-bit Windows, a 1.6 GHz or faster processor, a 5400 RPM or higher hard disk drive, a DirectX 9 capable video card running at 1024 x 768 or higher-resolution display, and a DVD-ROM drive

Computer Vision *S*oftware Requirements

- The software shall be split into two modes of operation, calibration mode and run mode

Calibration Mode Requirements

- Once the system is powered on, it shall wait for user input to enter calibration mode
- The calibration mode shall be compatible with a chessboard type calibration rig[2]
- The calibration mode software shall compensate for internal and external distortion effects.
- The calibration shall compensate for internal and external distortions of the camera system so distance information calculated in run mode is accurate to 5% in specified operating range.
- The calibration mode software shall transmit appropriate parameters to the run mode software.

Run Mode Requirements

- Run mode shall be entered upon successful calibration of the two cameras

- The cameras shall receive signals from the navigation control software of an autonomous vehicle or a user to acquire a pair of images for processing.
- The software shall complete all image processing within 5 seconds of receiving the images from the cameras
- The run mode software shall determine distances to objects with 5% accuracy
- The run mode software shall complete an image acquisition and computation cycle in less than 5 seconds.
- The run mode software shall present distance information as a text file for use by navigation control software.
- The run mode software shall be capable of displaying the detected edges, disparity map, or distance information on the monitor based on user input.

Spring Schedule

Table 1: Schedule of tasks to complete in spring 2012

Tentative Schedule for Spring 2012		
Weeks	Alex Norton	Matthew Foster
1	Assemble camera setup	Assemble camera setup
2	Configure calibration rig	Ensure OpenCV runs correctly on lab computers
3	Begin writing OpenCV code for calibration mode	Begin writing OpenCV code for run mode
4	Continue writing OpenCV code for calibration mode	Continue writing OpenCV code for run mode
5	Continue writing OpenCV code for calibration mode	Continue writing OpenCV code for run mode
6	Continue writing OpenCV code for calibration mode	Continue writing OpenCV code for run mode
7	Test and debug calibration mode code	Continue writing OpenCV code for run mode
8	Test and debug calibration mode code	Continue writing OpenCV code for run mode
9	Test run mode code with calibrated cameras	Test run mode code with calibrated cameras
10	Debug calibration mode code	Debug run mode code
11	Debug calibration mode code	Debug run mode code
12	Test and debug complete computer vision code	Test and debug complete computer vision code
13	Test and debug complete computer vision code	Test and debug complete computer vision code
14	Prepare for final presentation	Prepare for final presentation

Equipment List

- Two Logitech Quickcam Express webcams
- Compaq Presario CQ60 laptop
- Mathworks Matlab
- Microsoft Visual Studio 2008
    - OpenCV

- Equipment to be ordered:
    - Two webcams compatible with Windows 7 and Linux

Patents and Standards

Patents

Although there are many patents related to stereoscopic imaging and autonomous navigation, these are the ones most related to our project.

Table 2: Related Patents

Patent Number	Brief Description
6728582	System and method for determining the position of an object in three dimensions using a machine vision system with two cameras
6137893	Machine vision calibration targets and methods of determining their location and orientation in an image
7680323	Method and apparatus for three-dimensional object segmentation
5383013	Stereoscopic computer vision system
6392688	High accuracy stereo vision camera system
6807295	Stereoscopic imaging apparatus and method
6661449	Object detection device for autonomous vehicle

Standards

Applicable standards for our project are those related to the JPEG and PNG image formats, USB 2.0 and 3.0, and OpenCV versions 2.1 and 2.3.

The JPEG standard can be viewed at http://www.stanford.edu/class/ee398a/handouts/papers/Wallace%20-%20JPEG%20-%201992.pdf
The PNG standard can be viewed at http://www.libpng.org/pub/png/spec/iso/index-object.html#1Scope.
The USB 2.0 and 3.0 standards can be viewed at http://www.usb.org/developers/docs/.
Documentation for OpenCV 2.1 can be found at http://opencv.willowgarage.com/documentation/cpp/index.html
Documentation for OpenCV 2.3 can be found at http://opencv.itseez.com/.

References

[1]Gary Bradski and Adrian Kaehler. “Learning OpenCV”: Internet: http://www.cse.iitk.ac.in/users/vision/dipakmj/papers/OReilly%20Learning%20OpenCV.pdf, 2008 [Sept. 20, 2011]

[2]Jean-Yves Bouguet. “Camera Calibration Toolbox for Matlab”: Internet: http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/example.html, 2010 [Nov 13, 2011]

[3] Martin Peris. “OpenCV: Stereo Camera Calibration”: Internet: http://blog.martinperis.com/2011/01/opencv-stereo-camera-calibration.html, 2011 [Oct 5, 2011]

[4] Digital-Tutors. “Stereo 3D Disparity Maps”: Internet: http://www.digitaltutors.com/dtlabs/?p=645, 2010 [Nov 13, 2011]

[5] Mathworks. “System Requirements - Release 2011b”: Internet: http://www.mathworks.com/support/sysreq/current_release/, 2011 [Nov 28, 2011]

[6] Microsoft. “Visual Studio 2010 System Requirements”: Internet: http://www.microsoft.com/visualstudio/en-us/products/2010-editions/professional/overview, 2011 [Nov 28, 2011]

Appendix A
Pinhole Camera Model

The section below concerning the equations to determine the 3D position of objects using stereoscopic imaging is taken from the Project Proposal of the team that previously worked on this project, Adam Beach and Nick Wlaznik. These equations are derived for a system with the cameras mounted in line a horizontal axis, similar to how the SISMAV system will have cameras mounted in line on a horizontal axis.

The equations to calculate the 3D position of an object in Cartesian coordinates using locations of the object in two camera images are shown below. This technique is known as stereoscopic imaging. Figure 10 shows the setup of the cameras and the coordinate system to ensure the validity of these equations. The two cameras are placed so there is an upper and a lower camera, and the lower camera must be centered on the x-y axis. The x-axis is assumed to be vertical, the y-axis is horizontal and perpendicular to the line of sight from the center of the cameras, and the z-axis is horizontal and parallel to the line of sight from the center of the cameras. The positive z-axis is pointing toward the objects to be viewed.

$$X = \frac{X_D * d}{X_D - X_U} \qquad \text{(Eq. 1)}$$

$$Y = \frac{Y_D * d}{X_D - X_U} \qquad \text{(Eq. 2)}$$

$$Z = \frac{d * f}{X_D - X_U} \qquad \text{(Eq. 3)}$$

In the above equations, d is the distance between the centers of the cameras and f is the focal length of the cameras. It is assumed f is the same for both cameras. X_D is the distance in the x-axis between the object in the lower camera and the center of the lower camera. If the object in the lower camera is above the center of the camera the distance is positive while if the object is below the center of the camera the distance is negative. X_U is the same as X_D applied to the upper camera. The distance in the y-axis between the object in the camera and the center of the camera will be the same for both cameras, so $Y_D=Y_U$. If the object in the cameras is to the right of the center of the cameras (facing the back of the camera) the distance is positive while if the object is to the left of the center the distance is negative. Figure 10 shows the sign convention for the variables X_D, X_U, and Y_D described above. Equations 1 and 3 were found on the Cooper University website, www.ee.cooper.edu, in the week 5 lecture notes for EE 458. The equation to calculate Y was derived using Equation 3 and the equation for a line in the y-z plane from the origin to an arbitrary point in 3D space.

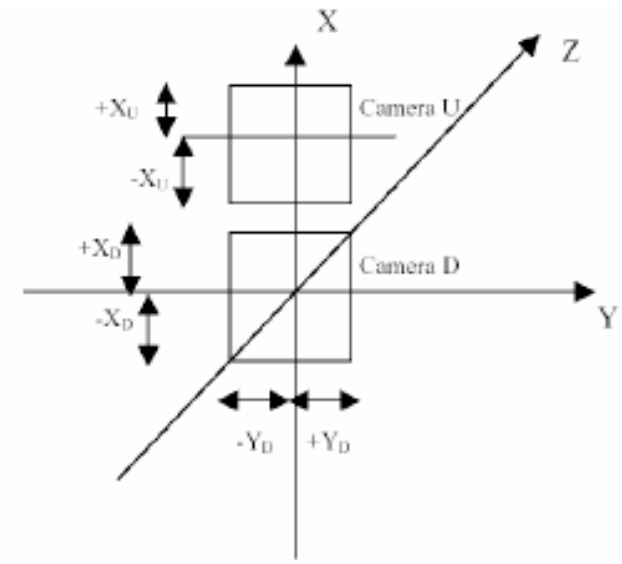

Figure 6: Setup of system to ensure validity of design equations