Stereovision Camera for 3D Strain Tracking of Ulnar Collateral Ligament: Preliminary Camera Calibration & Software Validation

Vaishali Shah  
*Washington University in St. Louis*

Leanne Iannucci  
*Washington University in St. Louis*

Spencer Lake  
*Washington University in St. Louis*

Follow this and additional works at: [https://openscholarship.wustl.edu/mems500](https://openscholarship.wustl.edu/mems500)

**Recommended Citation**

[https://openscholarship.wustl.edu/mems500/104](https://openscholarship.wustl.edu/mems500/104)

This Final Report is brought to you for free and open access by the Mechanical Engineering & Materials Science at Washington University Open Scholarship. It has been accepted for inclusion in Mechanical Engineering and Materials Science Independent Study by an authorized administrator of Washington University Open Scholarship. For more information, please contact digital@wumail.wustl.edu.
Stereovision Camera for 3D Strain Tracking of Ulnar Collateral Ligament:

Preliminary Camera Calibration & Software Validation

Vaishali Shah
Introduction

The overall objective of this project is to establish a method of taking three-dimensional strain measurements of ulnar collateral ligaments (UCL) using a stereovision camera during dynamic mechanical loading. Stereovision involves using two image sensors to acquire simultaneous 2D views of an object and rectifying these images together in order to create a 3D reconstruction of the scene [1]. The reason we aim to use a stereovision setup instead of performing point tracking using a simple single-lens camera is mainly due to improved accuracy of measurements from the depth information gained in 3D. The 3D reconstruction is able to give information about where a point of interest is in xyz-space whereas 2D imaging can only give xy-planar information when performing point tracking. For tracking bead placement on a tendon, depth sensitivity is crucial when performing strain measurements particularly if the tendon is in a complex, non-planar loading environment like the UCL. Additionally, we chose to use a commercially available stereovision camera (FujiFilm FinePix Real 3D W3) as the hardware already incorporates two synchronized sensors instead of having to perform manual synchronization during acquisition or post-processing of data from multiple cameras.

Matlab was chosen to perform analysis of the 3D videos because it contains an in-suite computer vision toolbox, which includes stereovision camera calibration. It allows the user to take simultaneous images of a calibration frame from each camera (in the cases of multi-camera stereovision system), or in this case each lens of the 3D camera and compare the two images in order to calibrate the camera. When the two views of the same scene are compared, the output will be a 3D point cloud where each point matches to a pixel in one of the two images. In this
toolbox, the user can also rectify the two stereo images which projects each image onto a common plane so that each corresponding point can be matched (Figure 1).

This image shows the left and right views from two different lenses (top two images). These are then combined to create a composite image (middle left image) and a image disparity map (middle right image). The 3D point cloud is shown as the bottom image. [2]

Figure 1

In other applications, stereovision and the computer vision toolbox in Matlab have been used for purposes in such as detecting human movement in surveillance videos and determining their distance to a camera or in robotics for allowing the robot to have spatial awareness [2]. In this project, the toolbox and stereovision camera calibrator app is used to get the stereo parameters for the camera setup. The stereo parameters themselves are constants that describe the relationship between the two sensors in space with respect to each other, as well as are properties of each of the two individual lenses imposed edge distortions. Knowing these parameters allows the user to correctly place and triangulate a point in 3D space in reference to the sensors. By obtaining the three dimensional coordinates of multiple points in space, the user can then determine the distance between those points in 3D, which will ultimately allow a true 3D strain measurement to be obtained when tracking points on a tendon.
**Semester Process**

**Literature Review**

At the start of the semester, I read through Matlab’s stereovision camera documentation online. I also read about the calibration app that Matlab offers through its computer vision toolbox. This was very helpful in writing the code later because I was able to fully understand all the functions of the toolbox, including the stereovision camera calibrator and how to implement it. I also read online more about what stereovision actually is and other applications of it as well as how it relates to human vision.

**Calibration Code Development**

After this, I went through the code that Leanne had previously written for this project and took notes from each line of code and how it worked in conjunction with the project as a whole. The structure of her code was how I based mine, and I worked closely off of her previously written code during this whole project. The way that the code is written is separated into two programs, a calibrator program and a testing program. The calibrator program utilizes Matlab’s stereovision vision toolbox. In this app in Matlab, images are taken of a checkerboard (the calibration frame) of a known size in order to calibrate the camera as a whole. They then can be uploaded into the app pairwise for the two sensors. The way that I used this feature was I set up a checkerboard against a blank wall and would move the camera around at different places around the checkerboard object. When the camera was moved around, I made sure to keep the checkerboard in the frame of the camera while imagining a bounding box around the checkerboard the size and depth of the size of the checkerboard (Figure 2). I made sure to take a variety of angles around the checkerboard in order to get a more accurate calibration of the sensors.
These images show the different angles taken around the checkerboard object. A variety of angles were taken, and each of these images has an image pair for the other lens.

Figure 2.

Since the output of videos from the 3D camera is just a single video in a 3D-AVI format, an online software called StereoSplicer was used to break the video apart into left and right camera videos [3]. Then in the calibration program in Matlab, I imported both left and right videos and separated the frames of each video to isolate 2D image matched pairs. Later in the semester when I started working on the validation process, I went back through the calibration program and had each image save to a folder specifying the zoom setting out of 10 and the camera, left or right, which it came from. After the program saves the images from the video frames, it calls the stereovision app from Matlab and each image pair is analyzed by the program. If the checkerboard was out of frame in either view, or the image was too blurry, the program would reject the pair. Once the acceptable pairs were loaded into the program, the calibration process took place. During the calibration process, the code finds each corner of the checkerboard in one view and matches it in the image from the other view, rectifying the two images. It repeats this process for all of the corners of the checkerboard. Based on these observed points, it calculates the stereo parameters of the sensors, indicating how the sensors are positioned to one another in space. It performs this calibration process for all of the inputted matched pairs, updating and increasing the accuracy of the stereo parameters for each pair. It
then, for each pair, projects onto the images a calculated location of the corner it had previously detected. It takes the distance, in pixels, between each projected point and measured point, called the reprojection error. It then calculates the mean reprojection error for all of the points in view and assigns it to the matched pair. The user then could manually eliminate any matched pairs with abnormally high reprojection errors so that any outliers were also thrown out of those frames being calibrated. After all of this, the program would re-calibrate the stereo vision camera without the eliminated matched pairs and export the stereo parameters into the workspace of the current Matlab session. In a single calibration, the original amount of frames would be around 1000, and the amount rejected would be around 400 frames. Each time I took a calibration video, I tried to make the camera as still as possible to minimize the amount rejected. After conferencing with Leanne and Dr. Lake, I decided to switch to moving the checkerboard instead in front of the camera in order to have a stiller video with less rejected stereo image pairs. This was successful in reducing the amount of rejected pairs by the software. In later calibrations, I will continue using this method in order to have more pairs read through the software, which will also reduce the mean reprojection error.

*Point Tracking Code Development*

The next program was the “testing” or point tracking program. Based on code written by Leanne previously, this program was able to take in the image of an object with multiple distinct points and determine the distance between those points. The image is taken in and first undistorted according to the exported camera parameters from the calibration program. This undistortion process accounts for any edge effects in the image (pin-cushioning, fish-eye, etc.) due to the lens. Once all the images are undistorted, they are converted to a black and white image, and a sliding threshold is used to select the grayscale level that distinguishes the two points from
the background in the photo. The user slides the threshold bar to blacken the background and leave only the white dots standing on the image. Then the two points are clicked on by the user with the mouse and the centroid of the white area is calculated by the program. By finding the coordinates of the centroid, the two dimensional distance in pixels between the two points can be calculated. With the 2D coordinates of each centroid along with the stereo parameters for the camera pair, these can be put into the triangulate function in order to get the 3D coordinates of each of the points. The distance can be calculated from these 3D coordinates and eventually strain can be calculated too.

**Validation**

After these two programs were working with no errors, I moved on to validation. The accuracy of camera to calculate the distance between two points was evaluated was using a phantom test object. In this case, the test object was a piece of white paper with two black dots on it with a known distance between them. First, I performed a calibration on one of the ten discrete zoom settings of the camera (Figure 3).

This figure shows the output window during the calibration process. One side shows a graph of the mean reproduction error across all image pairs (left) and one side shows the frames in relation to the position of the lenses spatially (right).

**Figure 3**

During this calibration on the first zoom setting, about 500 image pairs were taken from the frames of the video. Most of the accepted frames were clustered about 200-300 mm away from
the two lenses. The mean reprojection error was around 0.61 pixels. This was considered a successful calibration as according to the online Matlab documentation, a calibration with a mean reprojection error under 1 pixel is acceptable [2].

In the calibration window of the stereovision Matlab program, the distance from the sensors to mass of frames taken of the checkerboard is known and can be determined from the window (Figure 2). We defined the calibration volume as the region bounding all of the calibrated frames in space. We wanted to test the accuracy of the camera to detect distance between points both within and outside the calibration volume, so knowing where this volume is in space with respect to the camera is important for the validation process. This is so that we can get more insight into the accuracy of the calibration itself, the distance the camera should be placed away from the object of interest, as well as which zoom setting is optimal for a particular testing condition. We can define a bounding box around the calibration volume so we know where the test object should be placed for validation testing (Figure 4).

![Figure 4]

Each letter in this figure corresponds to a plane around the center of the calibration volume. The test object will be moved to each of these planes so that an image can be taken at each plane and each vertex of the bounding box. A calibration can be run on these images, and the graph plotting distance from the bounding box and error can be plotted for each zoom setting. This is a schematic of the bounding box around the test object in line with the camera.
Discussion/Future Work

For next semester’s work on this project, I am planning to move further into validation testing. For each zoom setting, I will be taking videos in and outside of the calibration volume with the goal of creating disparity graphs. This will show the relationship between distance from the center of the calibration volume and the error in measured distance. These graphs will help show which areas around the volume are especially prone to error and which zoom settings are already optimized for getting accurate distance measurements.

After doing this process for each of the ten zoom settings, I will move on to more experimental-type settings to test the program’s reliability. This will include planar orthopedic tissues like bovine tendons before eventually moving onto the ulnar collateral ligament. Points of interest to track like the dots on the sheet of paper can be placed on these tendons, and the distance between these points can be measured while the tissues are being mechanically loaded. From the distances measured, strain can be then calculated.

As well as a more dynamic testing setting, my goal for next semester is to also create a written procedure that fully details how to use this software so that the code is versatile. It will lay out which parts of the code should be changed for each computer utilizing the code as well as what to name the folders that Matlab creates through this code. I will also comment these into the code as well to make it more convenient. It will also specify the optimal distances found during the validation process of where the camera should be placed in relation to an object, how many frames should be taken of the object, and where the frames should be taken from. This is so that anyone in the lab can use this software in the future and feel comfortable with the results that it measures.
Overall, this project this semester has been mostly focused on learning about Matlab, the tools it offers and putting together the code in order to start the calibration and validation process of the 3D stereovision camera. Next semester, my goal is to continue validation of phantom objects for all discrete zoom settings of the camera as well as in a more realistic setting in order to create a working manual of the software for general lab use.
References:

   
   *Machine Vision and Applications*, Springer-Verlag, 2 Mar. 2000,


2. “Computer Vision Toolbox.” *MATLAB & Simulink,*


5. Leanne Iannucci and Dr. Spencer Lake