3D Robotic Vision Using Movement

By Marc van Zyl

Seth Bishop

Table of Contents

[**Introduction**](#_twdkxrrnmyk) **2**

[**Experimental Design**](#_ym9hqvrljvdq) **3**

[Problem](#_wysvl1niqppo) 3

[Hypothesis](#_rzf35vxvg0id) 3

[Control Group](#_dle2wa7uqss5) 3

[Experimental Group](#_6k9o4zw6rmyu) 4

[Independent Variable](#_2equw58viy1o) 4

[Dependent Variable](#_ec9px68m27vw) 4

[Procedure](#_glcuxmqu96up) 5

[Experimental Setup](#_aj8ykdox5yxw) 5

[**Calibration procedure**](#_64d93cldf9ac) **7**

[Data Gathering procedure](#_3ij34og0ft00) 11

[Analysis Procedure](#_pgvtx7t5yfnu) 11

[Stereo Camera](#_wzirwyftqove) 11

[Optical Flow Camera](#_4xitjagtvd1p) 13

[Data analysis](#_8wcyocq7jlrr) 13

[**Results**](#_91ta064976jh) **14**

[Stereo Camera Results](#_jnfy2if4tl60) 14

[Sparse Optical Flow Camera Results](#_kytw32ro5wk3) 15

[Dense Optical Flow Camera Results](#_b0b7uw4nwj5) 15

[**Discussion**](#_yp9b7fmp9shi) **18**

[**Conclusion**](#_git3rtl3gl0y) **18**

[**Future work**](#_w3z4ffbsnw0v) **19**

# 

# Introduction

In 2018, 25,000 children were reported missing or lost. Almost 92% were runaways, and many are found dead from the elements. If a child is not found within the first 24 hours, their chance of survival is halved, and by 48 hours the odds are extremely against them. At the moment, the only solution to search for children in the wilderness, especially those in forested or cramped areas, is to organize hundreds of people to make a chain across the forest. This is extremely costly, and they cannot cover enough area to be very effective. One of the biggest contenders to replace search chains are drones. They are fast, can cover large amounts of area, and can figure out what area they have missed. This can be done autonomously. In fact, there are drones like the DJI that can go through forests using distance detection. The main problem, however, is that distance detection requires a lot of processing power, so the drones move slower than the human chains, and they run out of battery much more quickly than they need to be effective, as they need to go out and come back, effectively halving battery life. Most distance detection uses stereo vision, like our own, with two cameras next to each other. If one camera could be eliminated, and leave a camera that only requires movement to see the distance to an object, the speed of processing is doubled and the battery is improved. Moving is not an issue - drones are doing that all the time anyway. Drones could become viable solutions to a problem that kills children under 14 everyday - people like myself. I could never imagine the terror, so I decided to try and save those that do experience such feelings. I need to first determine the accuracy of such an endeavor, which my science fair project is based around.

# Experimental Design

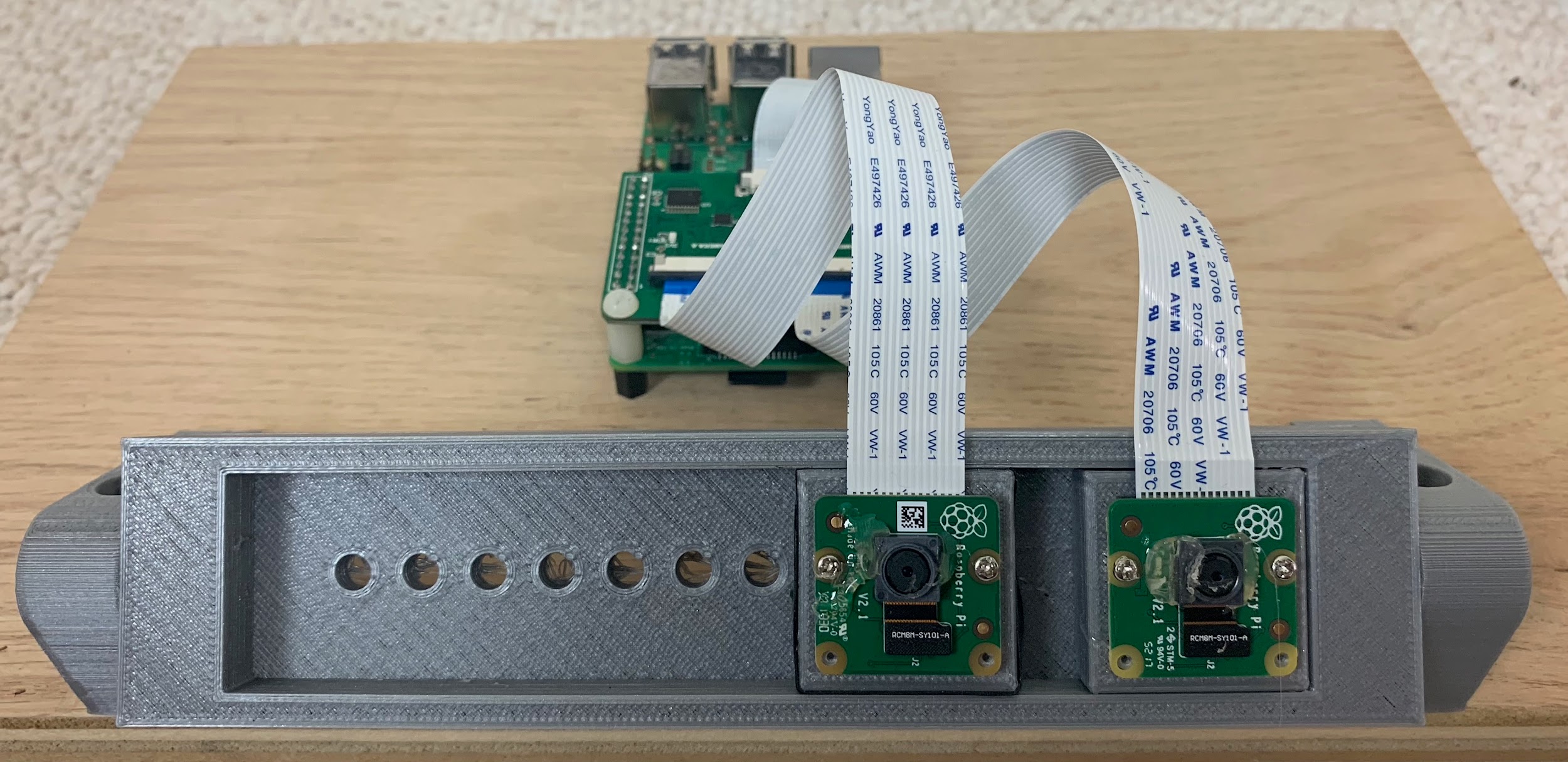
## Problem

My problem is that 68 children are reported missing every day in the US, and 80% of them are lost and endangered. The only solution currently effective is chains of hundreds of people that comb through forested areas, which is costly and prone to mistakes.

## Hypothesis

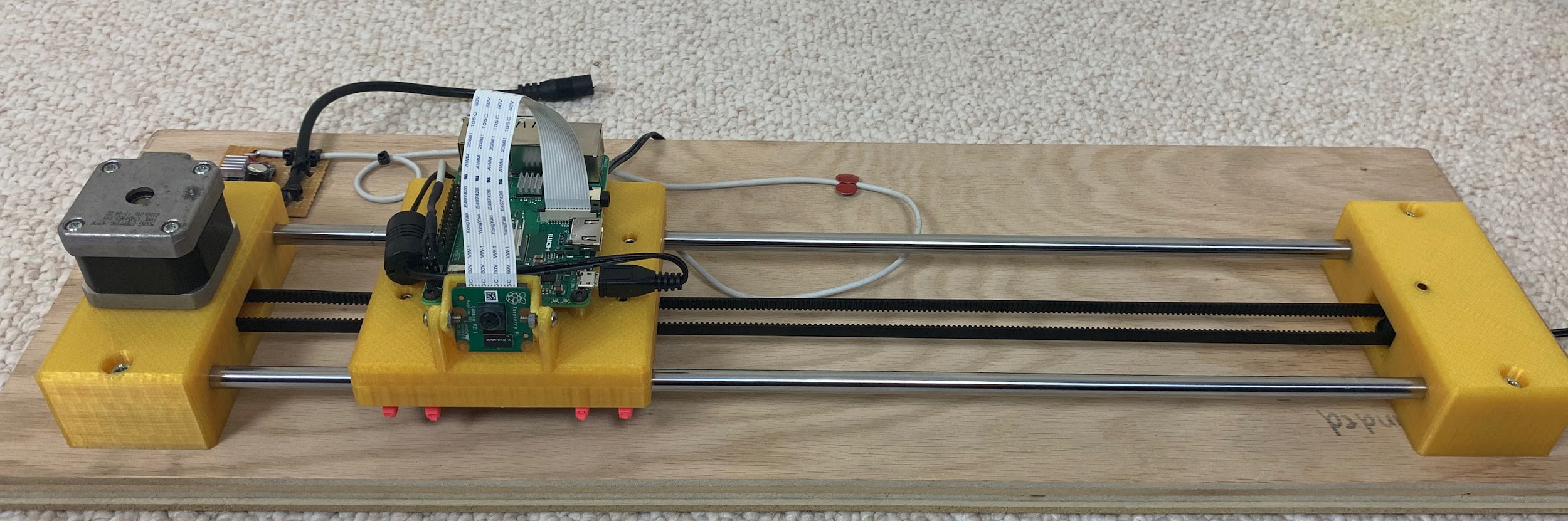
If optical flow applied to depth perception is more accurate and less computationally intense than stereo cameras, it will be a useful alternative or replacement.

## Control Group

My control group is a stereo camera setup that will use the same logic in programming and the same setup. It is adjustable, so the cameras can be set closer or further apart to gather more data and to test more variables. It uses a Raspberry Pi, like my experimental setup, and it uses the same cameras as well. It is a standard type of distance detection that many drones and other devices use.

## 

## Experimental Group

My experimental group is a camera mounted on 2 rods and is moved by an accurate stepper motor. It uses bearings to move on the rod and is moved by a belt that the stepper motor controls.

## 

## 

## Independent Variable

My independent variables are:

* On the control group, the distance between cameras, and on the experimental group, the distance between each picture,
* The distance to an object that is being used for accuracy testing.

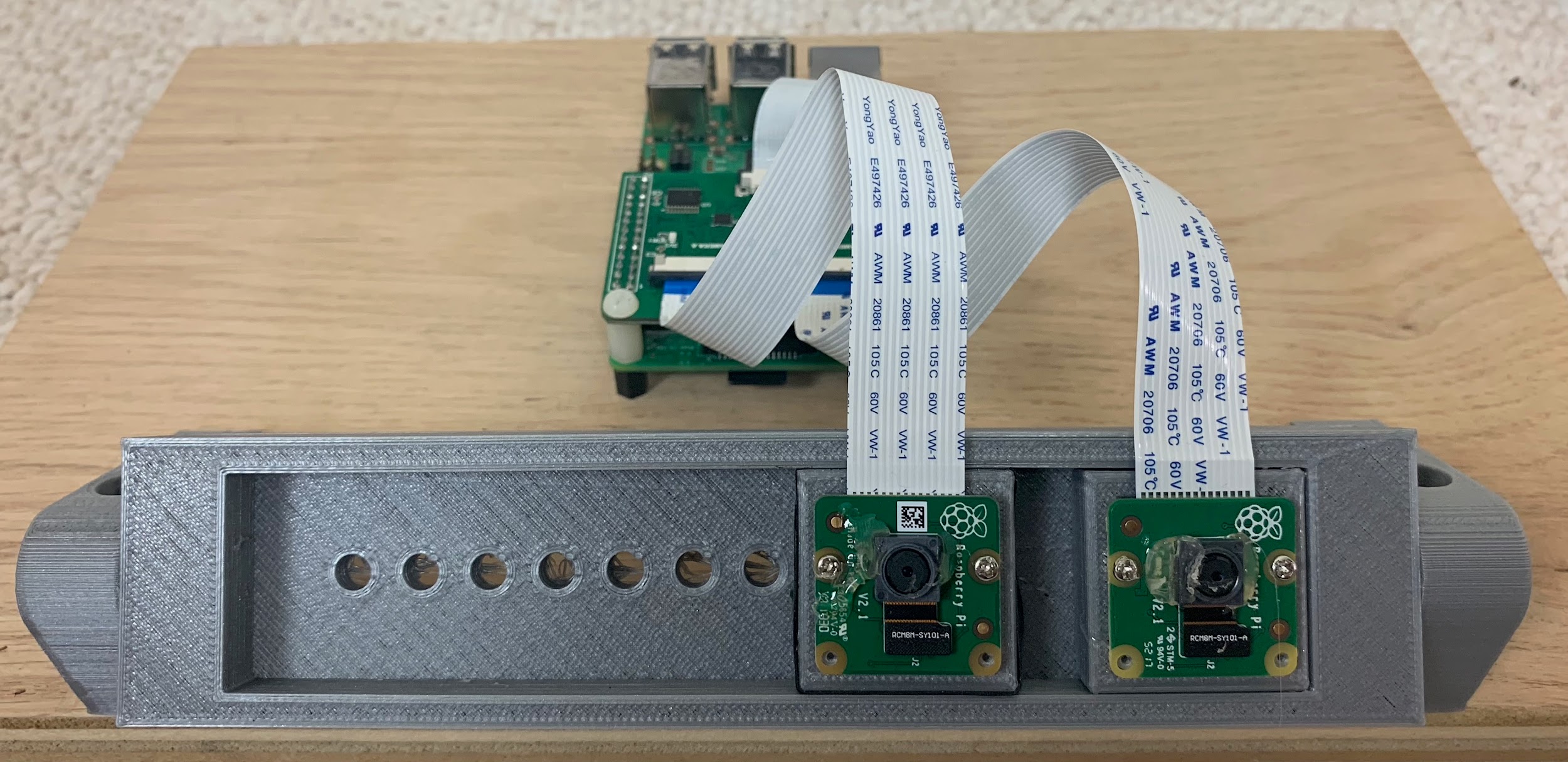
## Dependent Variable

My dependent variable is the accuracy of both setups in determining the distance to objects.

## Procedure

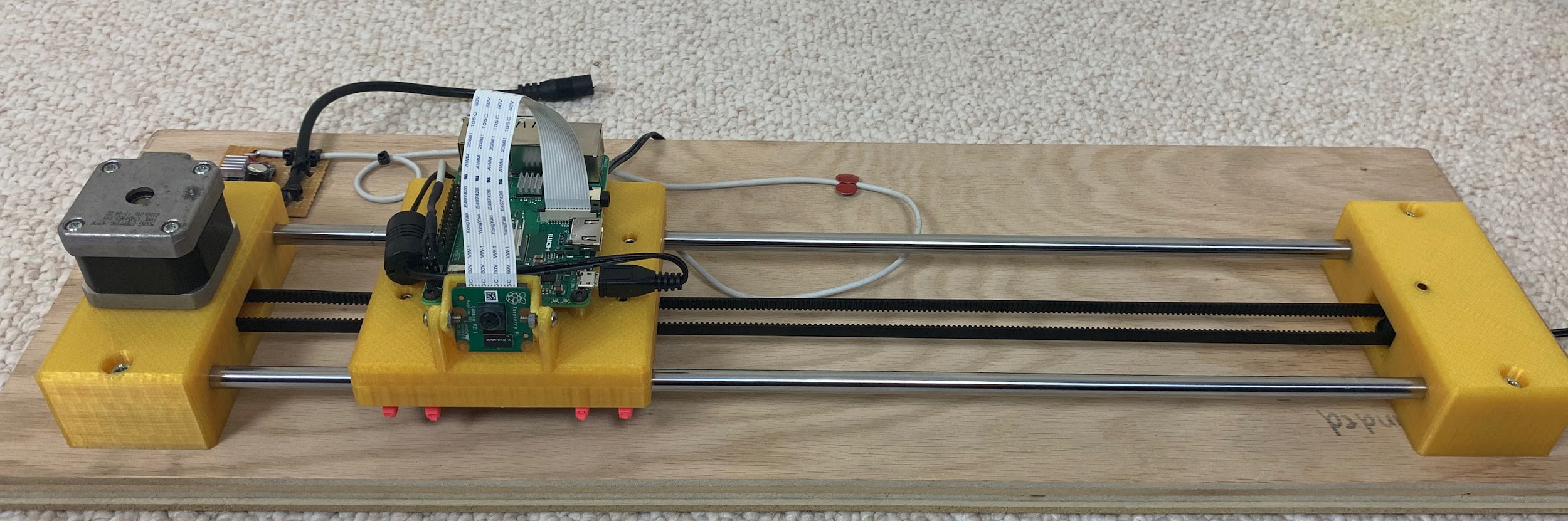
## Experimental Setup

1. Build Stereo Camera using:
   1. Raspberry Pi 3 B+
   2. Arducam Multi Camera Board
   3. 2 x PiCamera V2.1 8M color cameras
   4. 3D Printed stereo camera mounts



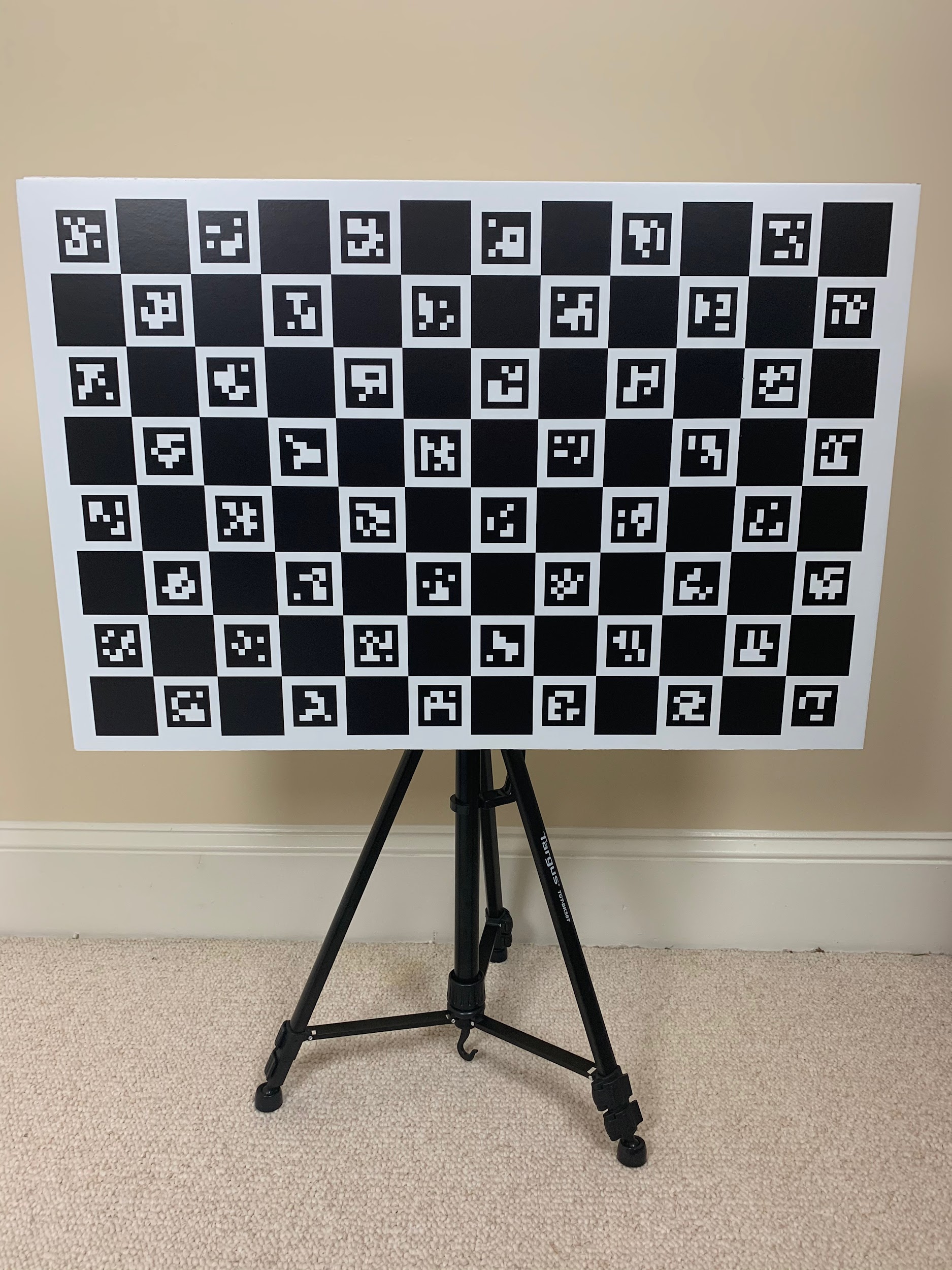
**Figure: Stereo Camera setup**

1. Build the Optical Flow linear slide using:
   1. Raspberry Pi 3 B+
   2. 1 x PiCamera V2.1 8M color cameras
   3. 3D Printed parts
   4. Linear slides and bearings
   5. Nema 17 Stepper motor
   6. Polulu DRV8825 High Current Stepper Motor Driver Carrier



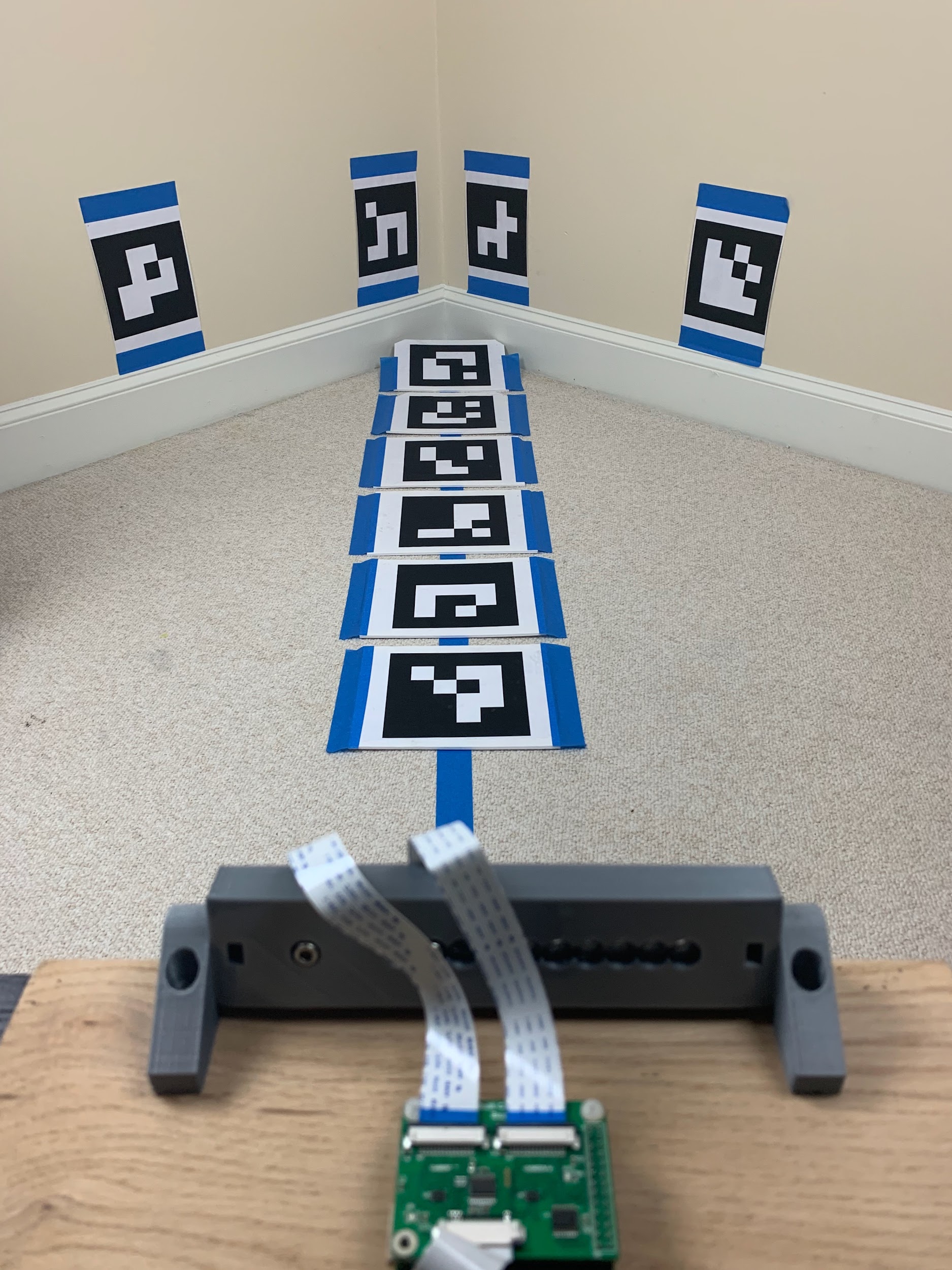
**Figure: Optical Flow experimental setup**

1. 30” x 20” CharUco calibration board on foam board and mounted on a tripod



**Figure: Checkerboard ArUco camera calibration board**

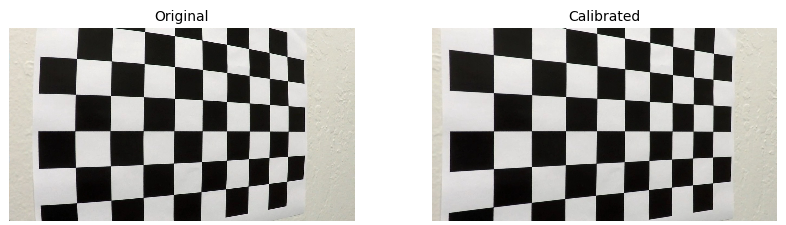
1. 10 x 8”x8” ArUco targets carefully laid out and measured in the target scene



**Figure: The target scene with AruCo markers mounted at regular intervals.**

## Calibration procedure

1. Stereo camera:
   1. Take 40 stereo images of the CharUco calibration board in various poses and distances from the camera
   2. Use these images to perform individual camera calibration in order to remove distortions resulting from the camera design (see figure below). This process delivers the camera matrix (mtx) and the distortion matrix (dist).



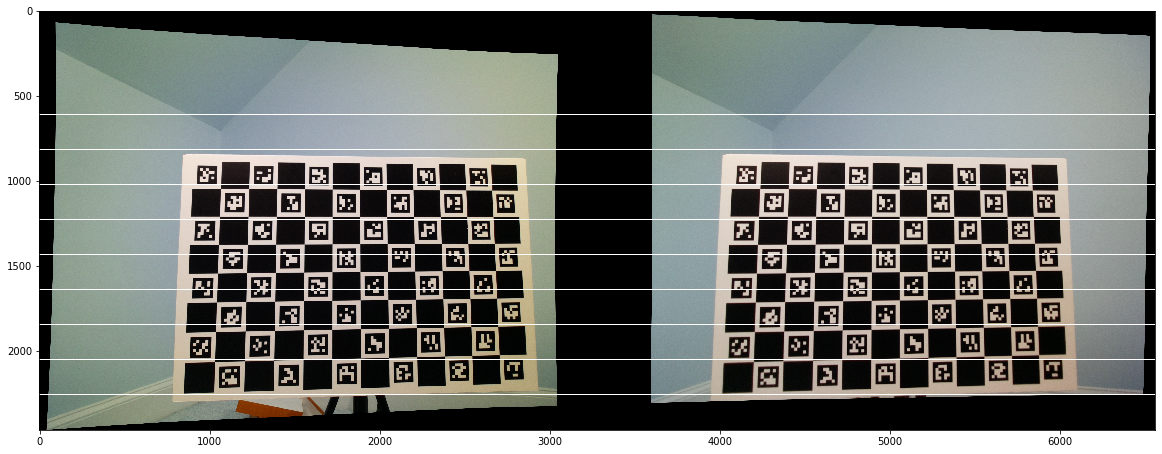
**Figure: Original and Calibrated image showing how the distortion is corrected and the bowed lines are straightened.**

* 1. After applying the camera calibration matrices from the step above find the matching ArUco markers in the left and the right cameras (see image below)

# 

**Figure: Matching the points in the left and right image. This process can be automated because the ArUco markers can be uniquely identified**

* 1. Apply the OpenCV stereoRectify algorithm to the points matched in the step above to determine the mapping function that will rectify the images (see figure below). The black borders around the images are an indication on how the images have been distorted to equalize and align the pictures. StereoRectify returns two mapping functions that can be applied to any Left-Right image pairs to rectify them. This concludes the Stereo camera calibration

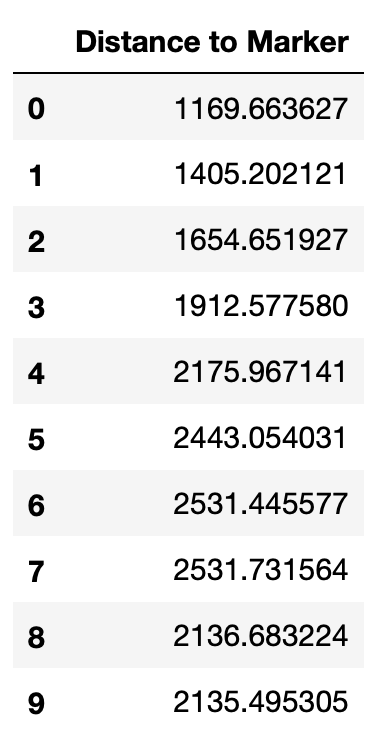


**Figure: The rectified images - the boards in both images are exactly the same size and are aligned so that the same points are on common horizontal lines**

1. Optical Flow camera:
   1. Take 40 images of the CharUco calibration board in various poses and distances from the camera
   2. Use these images to perform individual camera calibration in order to remove distortions resulting from the camera design (see figure below). As before, this process delivers the camera matrix (mtx) and the distortion matrix (dist) for the optical flow camera.
2. Optical Flow linear slide:
   1. The ratio between stepper motor steps and the distance travelled by the linear slide was determined by careful measuring. The result was 40 steps/mm.
   2. From this the velocity of the slide can be determined by knowing the frequency of the step signal created by the Raspberry Pi.

where is velocity in mm/s and is frequency in Hz.

1. Scene calibration:
   1. Mount the ArUco marker on foam board and lay them out in the scene.
   2. Measure the cartesian coordinates *(x,y,z)* of the center of each marker to an accuracy of +/- 2 mm
   3. Place the camera and measure the position of the camera lens in the same cartesian coordinates *(x,y,z)Camera*
   4. Using a Python program the distances from the camera to each marker can be calculated. The results were:



## Data Gathering procedure

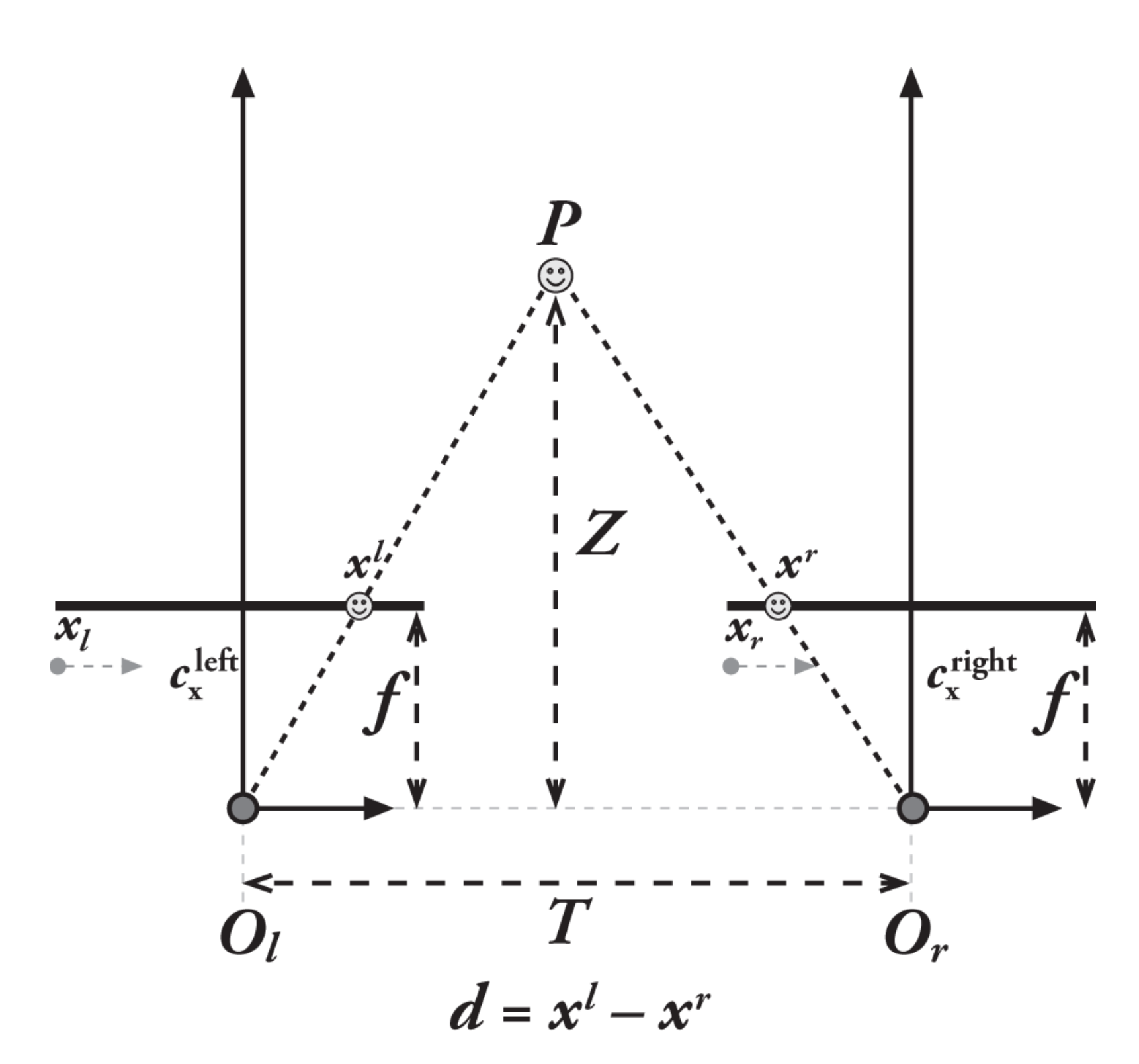
1. Stereo Camera:
   1. Take 50 stereo pictures of the scene moving and rotating the camera a few mm between each picture to get a diversity of images. This process was automated by a small program on the Raspberry Pi
2. Optical Flow Camera:
   1. Write a small program on the Raspberry Pi to automate the following:
      1. Enter a desired velocity
      2. Enter the test name
      3. Enter the number of runs
      4. For each run:
         1. Start the recording
         2. Move the camera at the desired speed
         3. Stop recording and save the file
         4. Move the slide back
   2. Run the program for four runs at with *v =* [15, 30, 45, 60, 75] mm/s.

## Analysis Procedure

### Stereo Camera

1. Apply the rectification maps to the Left-Right image pairs
2. In each image find the four corners of the AruCo markers (using a corner detection algorithm in the ArUco library) in camera pixel coordinates.
3. Optimize each corner to sub-pixel accuracy.
4. Find the location of the center of each ArUco maker by interpolating between the *x* and *y* values of the four corners.
5. Find the pixel disparity between the center of each marker in the Left and Right image pairs.
6. Find the distance to the marker center using the following equation (where *Z* is the distance, *T* is the distance between the cameras and *f* is the focal length of the camera):

The geometry of the solution is shown in the figure below:



**Figure: Calculating distance from pixel disparity *d = xl - xr***

### Optical Flow Camera

1. For each experimental run split the video file frames into individual images (the images were captured at 15 frames per second)
2. Apply the camera calibration matrices to undistort each frame
3. In each frame find the four corners of the AruCo markers (using a corner detection algorithm in the ArUco library) in camera pixel coordinates.
4. Optimize each corner to sub-pixel accuracy.
5. Find the location of the center of each ArUco maker by interpolating between the *x* and *y* values of the four corners.
6. Using a simplified version of the optical flow equation (motion is linear and all points will only move in the *x*-direction) find the distance to the center of each ArUco marker by applying the following formula to successive frames:

where

### Data analysis

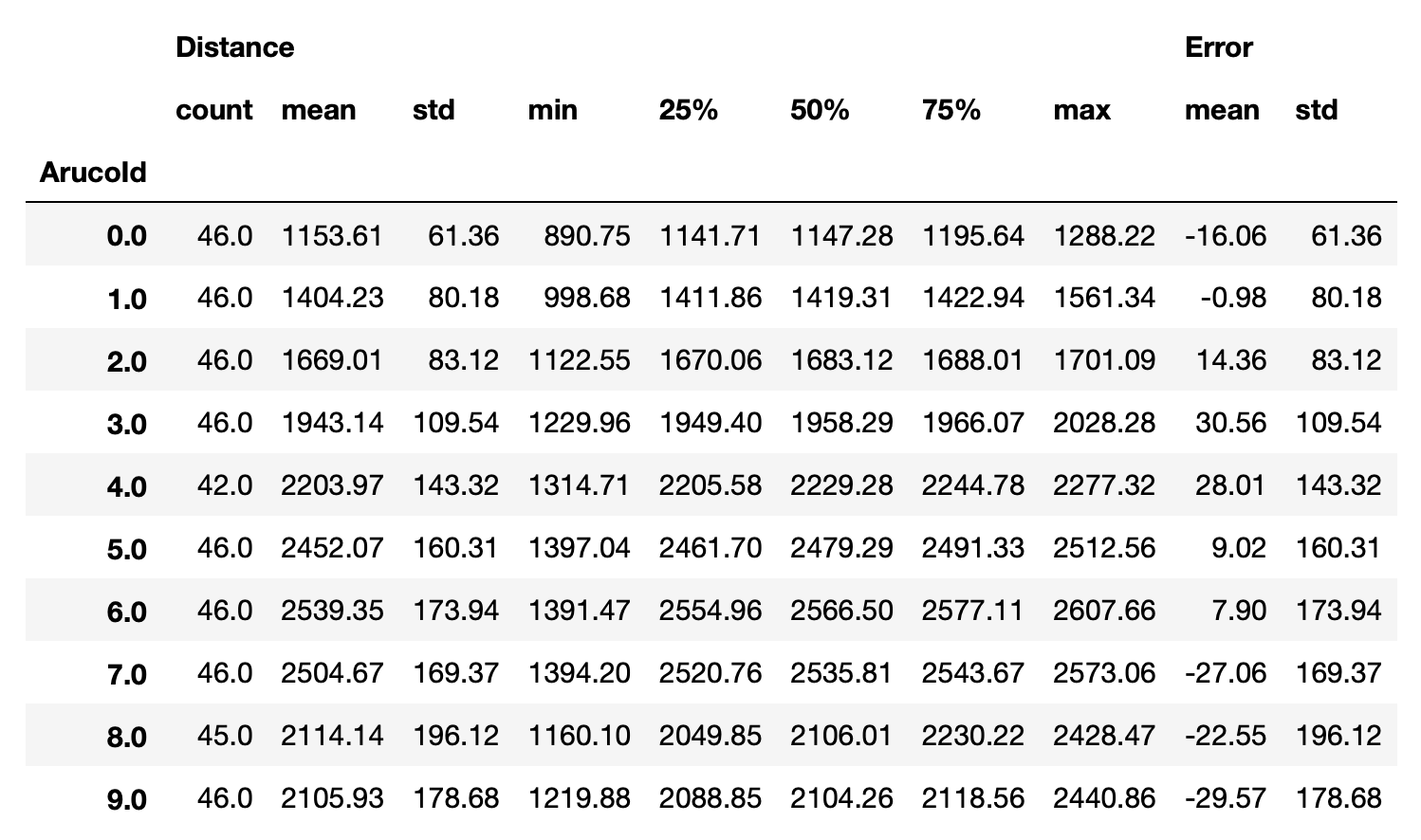
1. For each of the Stereo and Optical Flow setups:
   1. Calculate the differene between the calculated distance and the actual distance to each target. This is the *Error.*
   2. For each target calculate the:
      1. *mean(Error),*
      2. *StdDev(Error)*,
      3. *min(Error)* and
      4. *max(Error).*

# Results

The results of the experiments are illustrated below:

## Stereo Camera Results

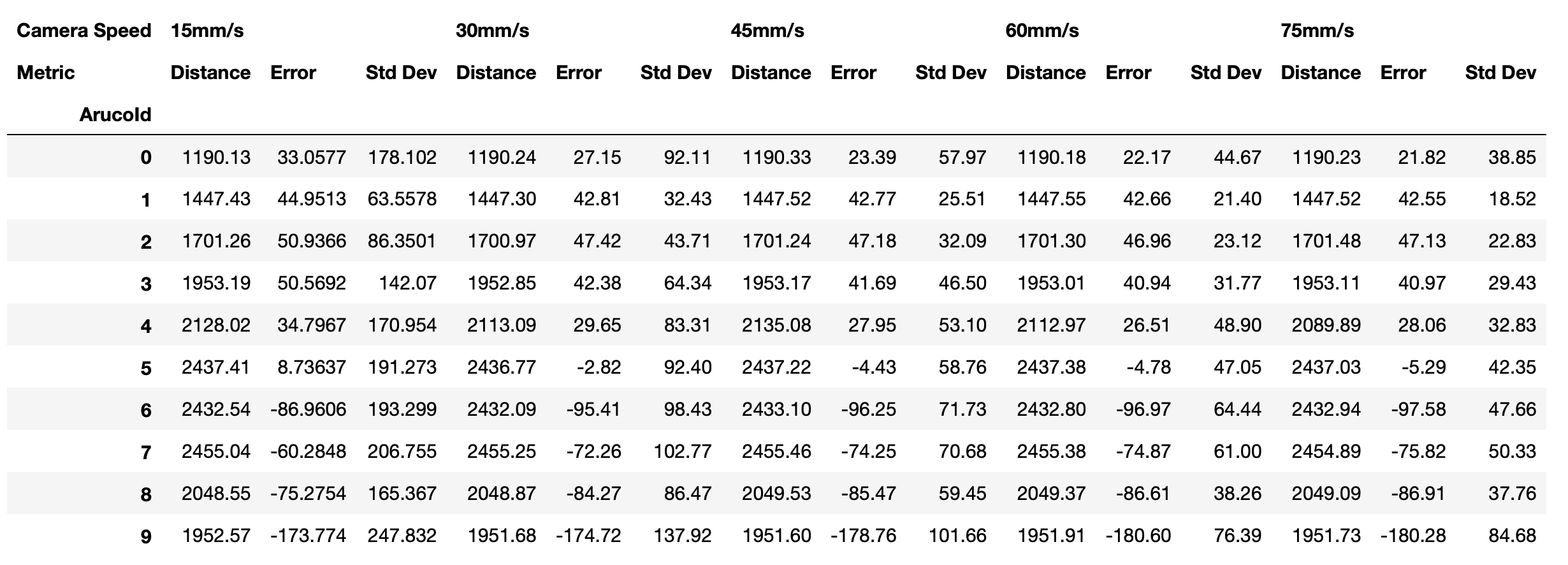
**Table: Measured distance and Error statistics for Stereo Camera (n=42,45,46)**



## 

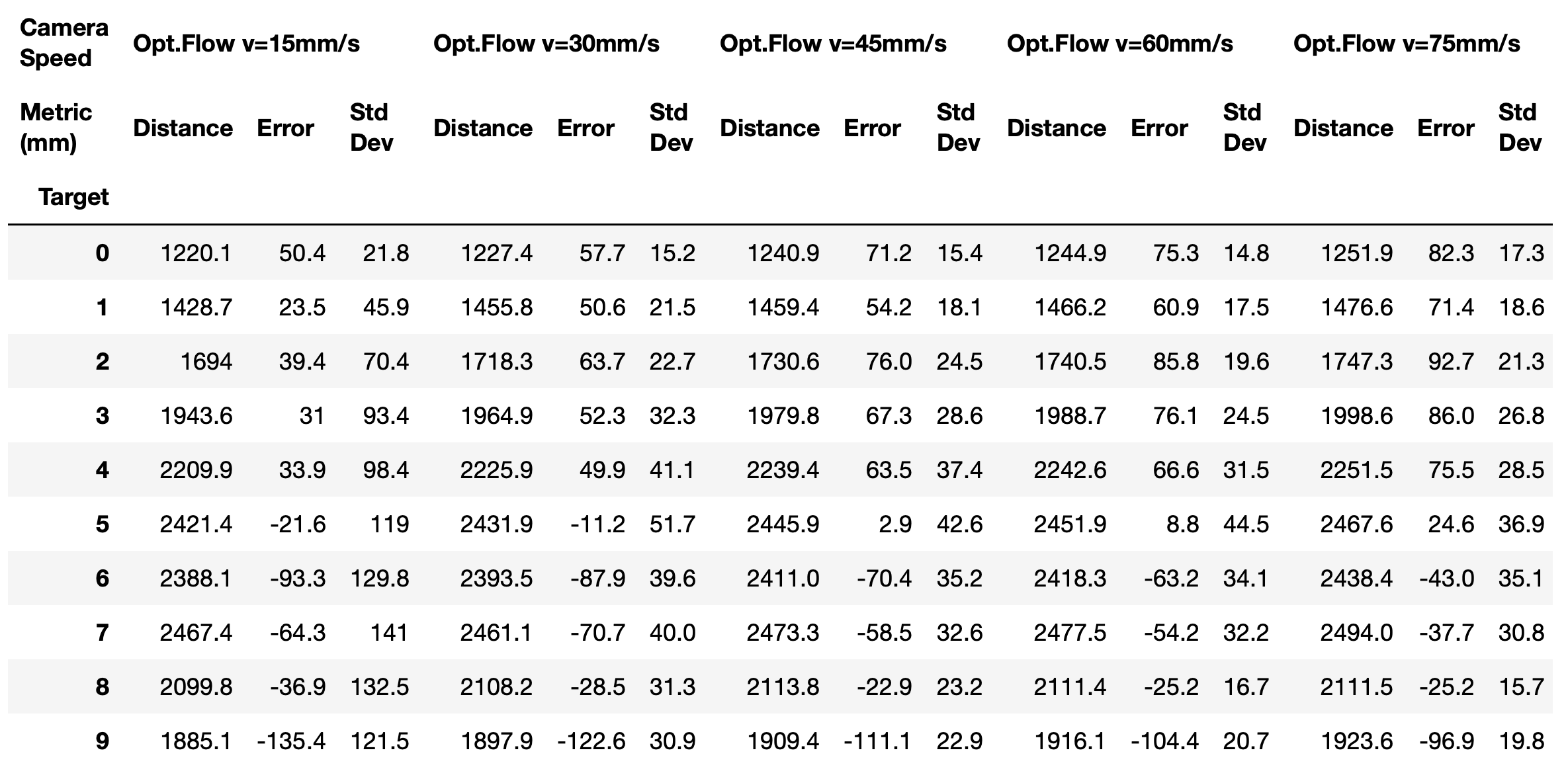
## Sparse Optical Flow Camera Results

**Table: Measured distance and Error statistics for Optical Flow Camera (n=4 x 50-300 frames) for *v =* [15, 30, 45, 60, 75] mm/s**

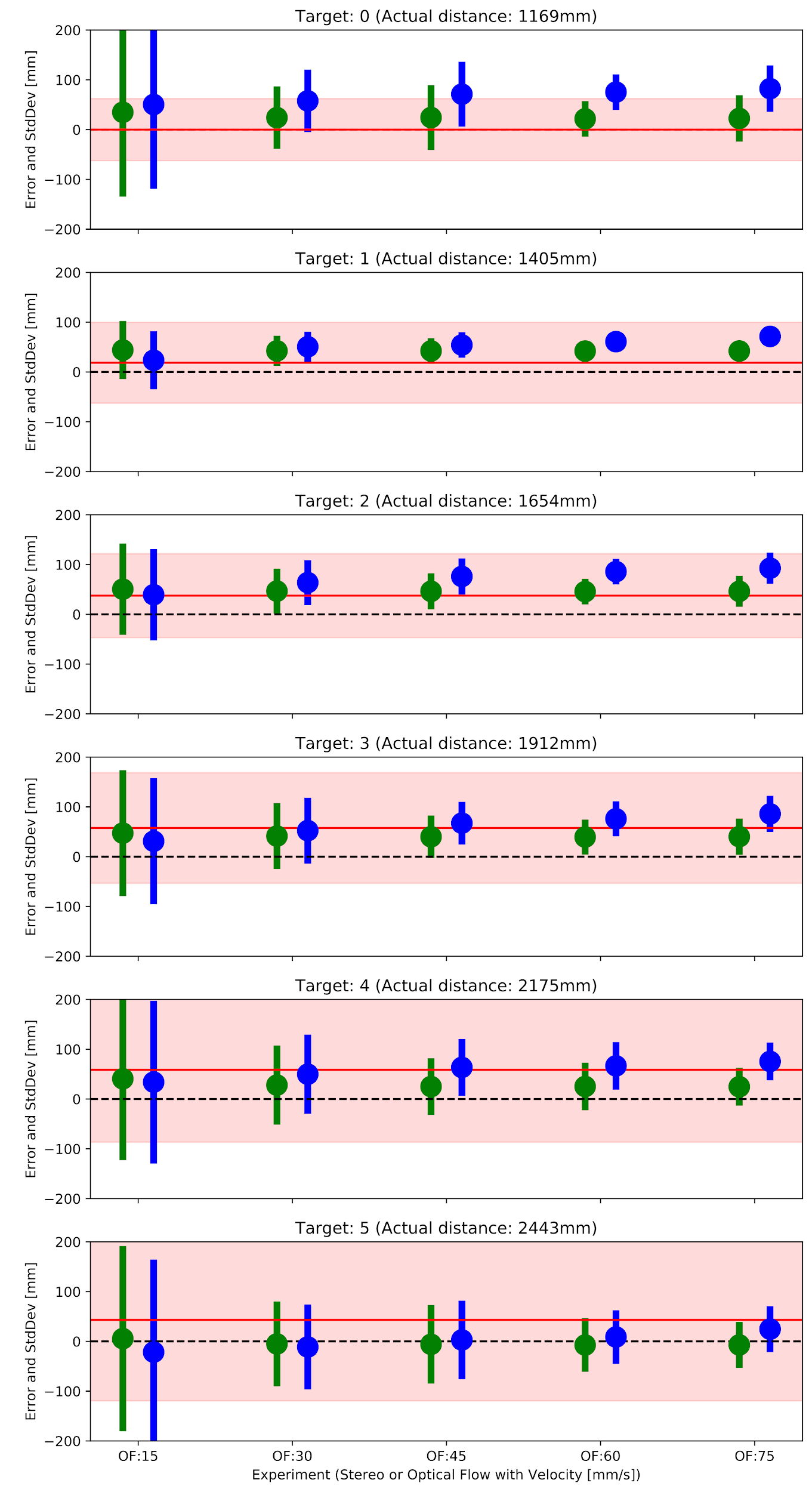
****

## Dense Optical Flow Camera Results

**Table: Combined results for Stereo Camera and Optical Flow Camera with *v =* [15, 30, 45, 60, 75] mm/s**

****

**Error and Standard Deviation of Error Between Experimental Results and Actual Target Distances**

****

**Figure: Final results showing measurement error for Stereo and Optical Flow Cameras for different velocities *v =* [15, 30, 45, 60, 75] mm/s**

# Discussion

The results show some interesting insights:

1. The accuracy of both systems was surprisingly good.
2. The stereo camera was more accurate than the optical flow camera for 3 of the 6 markers at *v* = 15mm/s, however,
3. The accuracy of the optical flow camera decreased very significantly at greater travel velocities. The reason for this is probably because at greater velocities the disparity between frame is larger and more accurate triangulations were achieved.
4. While the average accuracy of the stereo camera was quite good, the standard deviation of the error was on average greater than 130mm. This was only better than the optical flow camera at *v* = 15mm/s. The optical flow camera performed significantly better at higher velocities.

# Conclusion

For the special case of linear camera motion, the experiment confirms that Optical Flow algorithms can active comparable accuracies to traditional Stereo vision systems with much lower variance in measurements. While the overall accuracy of Stereo vision was greater at the very close distances, Optical Flow was more accurate at the longer distances. This is probably due to the Optical Flow algorithm using multiple images from the video stream to update measurements over time. This points to some interesting possibilities for robotic vision in the future. Dense optical flow is particularly interesting due to its ability to map a full scene and refine its accuracy over time.

# Future work

Future research will be to enhance the algorithm to account for linear and rotational motion of the camera and to enhance the dense optical flow algorithm to identify separate objects and convert the scene into a 3D map.