Camera Calibration Using Tsai's Algorithm

Hunter Morera[1]

I. ALGORITHM DESCRIPTION

The algorithm that we are using in this project was first discussed by Roger Tsai in his 1987 paper where he begins by addressing some of the problems with the current camera calibration algorithms available at the time. Some of the problems he addresses are that prior algorithms cannot consider the lens distortion present in the camera, and in many of the previous algorithms the focal length is assumed as a given parameter [1]. Tsai's algorithm is a four step solution that requires multiple inputs. To begin the algorithm must be supplied with a list of points that consist of a locations real world coordinate $[x_w, y_w, z_w]$ and the pixel locations in the corresponding image $[x, y]$, where the world coordinates are defined by the distance from the origin of the three intersecting planes used for calibration. The algorithm then solves for the rotation and translation matrices, from this the focal length of the camera can be determined [1]. Once these parameters have been solved for it is possible to then finally the lens (radial) distortion parameter can be solved for. The algorithm then returns the intrinsic parameters of f_x & f_y the focal lengths, as well as the lens distortion coefficient k_1, and the uncertainty scale factor s_x. The extrinsic parameters are also returned where each element of the 3x3 matrix are a function of the euler angles, yaw, pitch, and tilt [1]. These values of intrinsic and extrinsic parameters can be used to un-distort the images taken by this camera which will compensate for the distortion that the lens applies in taking the photo. This method of camera calibration has been shown to be very robust and is still used as the primary method of camera calibration in many applications over 30 years after its publication.

II. IMPLEMENTATION USED

As discussed in class on multiple occasions we were not required to code this algorithm by hand and were welcome to use others implementation. After a few failed attempts with using python packages available via pip, I took Dr. Sarkars advice and searched out the C implementation. This implementation was written by Manuel E. L. Fernandez & Marcelo Gattass in 2006, given a set or real world and image points it determines the cameras intrinsic and extrinsic parameters[2]. After speaking to Dr. Sarkar in office hours about exactly what we were responsible for with this project which is to test the robustness of this implementation as if we were planning to by it from some company. Therefore I began to dig through the code to find exactly what and how the parameters needed were used. I found that this implementation was originally designed to be used with a few different cameras, who's camera array values were already defined. However, as the goal was to calibrate my own camera I first went about finding my cell phones actual camera and the specs of the sensor array it uses. I was able to research and find that my Iphone 7 uses a Sony IMX315 camera, from this I was able to use the Sony website to determine the length and width of the sensor array, as well as the size of each individual sensor of the array. This was the information I needed to add a new camera to the code I found and set all of the parameters with those I found for my phone. As well I will note that this code came with no readme file so I had to determine how to compile and run it from scratch to even test it. Now that this was done I constructed the calibration object by printing out three checkerboard pattern and used a cardboard box to construct an object that had three distinct planes each of which was a checkerboard pattern, and I measured out the size of each square to determine the real world coordinates of different parts of the object. I took three photos from different angles and distances in order to test the robustness of this code[2].

III. RESULTS

As discussed above I wanted to test this codes robustness by taking multiple photos from different angles and distances. As well to make the test of the algorithm a bit more rigorous I did not use the same points in each of the three images to do the calculations. I did now After acquiring the photos I began to extract the real world and image coordinates by hand. I did try to use the extract checkerboard corners function that is implemented in OpenCv but had no luck with that. The code I was using read the data from a text file and output the intrinsic and extrinsic parameters accordingly. At first my results for the intrinsic parameters we varying widely between images, so I went back to the drawing board and re-computed some of the calculated parameters of the sensor array and I was able to fix the problem. After this my intrinsic parameters, my K matrix, was consistent in its values to within 1mm of difference. From the previous research I did on my camera I knew that my actual focal length as per Sony was 3.9mm. My results of this value from the calibration code were as follows. Fig 1a produced an approximation of 2.8mm, Fig 1b of 3.9mm, and Fig 1c of 4.0mm. This is very close to the ground truth of 4mm and I believe the error that is seen in my results originate from one ore more things. As well my distortion parameters were pretty consistent between the three images,

[1]H. Morera is a PhD Student of Computer Science and Engineering, University of South Florida, 4202 E Fowler Ave, Tampa, Florida, USA hmorera@mail.usf.edu

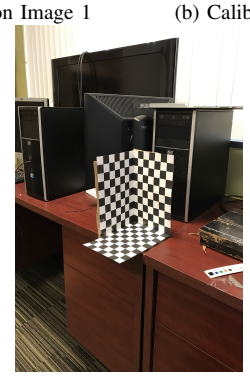
the results of the three were -0.3, -0.4, -0.5 respectively. Firstly is that my printouts of the checkerboard pattern did not fill the page perfectly, the edges of the paper had to be trimmed. In an attempt to keep from cutting too much I used a paper cutter to allow for a more accurate and clean cut. However, there was still some human error there and some squares on the edge were not exactly 2.5cm as they should have been. To compound this error it was also hard to tape the grids in such that they matched up perfectly so that each square was exactly 2.5cm times the number of squares away from the origin. I believe that these errors along with the approximation of marking pixels by hand in an image was the reason for the 1mm difference between the three images. Other than this I was surprised at how robust the algorithm was despite this, 1mm difference between three images is very small when all of the sources of possible error are considered. All of the results described were acquired with just four points on each of the three planes in the image. I believe if I had increased this number to include all of the grid square corners in the image my consistency between images would have improved. The code I used had a test file that had one hundred data points for each of the three planes to do the calculations[2].

IV. LESSONS LEARNED

I can't say that I learned as many lessons with this project as other, simply because I was not required to code the actual Tsai algorithm. At the same time I am very thankful for that. I read through the white paper that Dr. Sarkar put up online and it is very dense, as well I dug through the code I used for this project and it is very advanced. However, I do feel some of it seems a bit more complicated because it is being done using c and there are a lot of pointers and memory allocations that would not necessarily be needed in other languages. Despite not actually writing the code I certainly still learned some lessons. The first is that using a cellphone for this assignment was probably not the best idea, it took me a lot of time to track down which camera sensor is actually used in my Iphone7, it is not something that apple or many other companies advertise when they give the specs of a device. This is in contrast to a traditional camera from Sony or Nikon where they have very detailed in depth specs with the exact model of sensor they use and what the parameters of such sensor is. I also learned that those sensor parameters are very important to the accuracy and precision of the results that the algorithm provides. Prior to pulling my sensors measurements off of Sony's website I tried to use what I found on Wikipedia and they were clearly way off because my estimated focal length was in the fraction of a mm size. As well as I discussed above I learned that the quality of the calibration object is very important, I believe with a higher quality object that I could achieve sub-millimeter precision between multiple images. I also came to find that the image center to use for my camera is shifted a bit to what one would think the center values should be, but with some trial and error I was able to narrow down a consistent value that produced the above results over the three images. Another thing I observed which I noted above is that some of the error may be due to the fact that I am only using four points to define each plane, as opposed to the original implementation which used over one hundred points for a single plane, it is possible that with more points per plane my deviation between images would be reduced. Finally I want to discuss some failed attempts which I tried, I expected them to fail when I tried but I figured it was worth a shot. I tried to estimate by only using points on two of the three planes, this produced results of a focal length of less than a quarter of a millimeter and a distortion factor that was way off from the consensuses of the correctly defined images. As well I tried to only use two points per plane which caused for erratic results similar to those when you only define two of the three planes. These were expected failures but I wanted to personally verify these things that Dr. Sarkar explained would completely throw the measurements off. Therefore I would say that despite the fact that I was at liberty to use a preformed "library" to complete this project, I still feel like I learned a lot and re-enforced the content and concepts that were explained in class.

(a) Calibration Image 1

(b) Calibration Image 2



(c) Calibration Image 3

Fig. 1: Calibration Targets

REFERENCES

[1] R. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses," *IEEE Journal on Robotics and Automation*, vol. 3, no. 4, pp. 323–344, 1987.

[2] M. Fernandez *et al.*, "Ansi c implementation of classical camera calibration algorithms: Tsai and zhang," *https://webserver2.tecgraf.puc-rio.br/ mgattass/calibration/*.