Final year Report

REPORT ON
3D RECONSTRUCION PROJECT

Shrinal Thakkar
Computer Science
Lakehead University
Thunder Bay, Canada
Sthakka1@lakeheadu.ca

Supervised by
Dr. Shan Du
Assistant Professor
Lakehead University
Thunder Bay, Canada
sdu2@lakeheadu.ca

Contents

Abstract

One the of the leading technology is 3D reconstruction which have been a challenging at very points, in this report we will talk about a system build on python programming language generating a 3D model from stereo vision. The system uses any regular camera which in first phase gets calibrated extracting camera information like focal length distortions in camera, intrinsic and extrinsic parameters of camera etc. In phase 2 we will use those parameter to create a disparity map and using that map we will generate a 3D cloud which will we transfer into a 3D point cloud using a transformation matrix and reprojecting points into 3D and creating a point cloud. Finally, we will use MeshLab to visualize our 3D model.

Keywords: 3D reconstruction; 3D model; camera calibration; depth map; point cloud
Python programming language.

Acknowledgment

I would like to express my deep and sincere gratitude to my supervisor,   Dr. Shan du, Department of Computer Science, Lakehead University. I could not have imagined having a better advisor and mentor for my master's Project. It has been a real pleasure working with him. I would also like to thank my family, friends for supporting me spiritually throughout this Journey.

1. Introduction.

3D reconstruction is a blooming field day by day newer technologies are created which are working on a 3D rather than old school 2D space. As being human we have a 3D view but when we work with computers, they lose a dimension in the projection process, for a computer it's hard for to estimate true 3D geometry of an object [1]. In 1960s , Ivan Sutherland define 3D modeling can be simply defined as the process of building 3D digital visual representation of an actual object using specialized computer software. Which in that software we can , flipped, exploded, or manipulated in all sorts of ways on the screen[1] .One of the greatest advantages of 3D modeling is that the data points can be anything to do with the object itself or external factors, such as weather and possible interactions with other objects including humans/users[2]. While 3D modelling offers very accurate designing of objects, they need tremendous data like width, height, refence points and many more. But while working with a 3D reconstruction those variables are unknown and we face a problem of calculation using mathematical equation.

For a computer an image is just a 2D array, but we live in a 3D world where we have volume, affordances, and are spatially arranged with objects occluding each other. The ability to reason about these 3D properties would be useful for tasks such as navigation and object manipulation. As humans, we perceive the three-dimensional structure of the world around us with apparent ease [1]. As a computer it has been a challenge to see the world like human and it has been shown to be a very difficult task, and have been studied for about 50 years in the 3D reconstruction community in computer vision, which has made significant progress. Especially, in the past two decades, the dominant research focus for 3D reconstruction is in obtaining more accurate depth maps
or 3D point clouds.

Even though we have these advances, the task of having a computer interpret an image at same level as a two-year-old (for example, counting all the objects in a picture) remains elusive. Even after we have great information like a depth map, we still cannot manipulate an object because there is no high- level representation of 3D world., we would

[1] cadcrowd.com/blog/3d-modeling-overview-history-industry-applications

like to argue that 3D reconstruction is not just a low-level task. Obtaining a depth map to capture a distance at each pixel is analogous to inventing a digital camera to capture the color value at each pixel. The gap between low-level depth measurements and high-level shape understanding is just as large as the gap between pixel colors and high-level

semantic perception. Moving forward, we need a higher-level intelligence for 3D reconstruction. One of the main uses of 3D reconstruction is the automatic generation of 3D representations of object that are difficult to model, which helps us with to speed-up the model generation process for graphics applications. This technique can be utilized indoors and outdoors scenes, with some improvement [2]. This era 3D reconstruction has seen one as one of the focus of development in computer as he 3D reconstruction's application are limitless, which ranges from medical application, game development to image processing.

For a 3D reconstruction are two method definition, either a method is passive or an active method. A passive method does not involve interaction with the object, whereas active methods use contact or a projection of some form of energy onto the object. Whereas, an active method that require some sort of energy projection (such as, lasers or structured light) or use the relative motion between one or more camera and objects, to obtain 3D information on the objects shape, Examples range from moving light sources, colored visible light, time-of-flight lasers to microwaves or 3D ultrasound. See 3D scanning for more details. Even though this methods are very different in some case we have seen them working them in together and they share the same steps which are required which are: -

- Image Obtaining.
- Camera calibration
- Feature extraction
- Stereo correspondence
- Reconstruction

This report will highlight the system which undertakes a passive method for reconstruction. Here I have created a stereo vision reconstruction which like our human vision as in working on 2 images one as our right and one as left. Those two images will work as our input from our camera which is calibrated by us using Find cheeseboard algorithm. Then using a Semi-Global Block

Matching (SGBM) a feature matching algorithm we create a disparity map. We transfer into a 3D point cloud using a transformation matrix and reprojecting points into 3D and creating a point cloud. Finally, we will use MeshLab to visualize our 3D model.

2. Literature review and Background

Here, in this section we will go though some technology that are well known for 3D reconstruction or provide a good understand for this topic.

“Computer vision-based 3D reconstruction : A review”
3D reconstruction is used in various field being object reconstruction to model or cultural artifacts. The scientists are beneficial for these task in order to learn and keep the environment into 3D data due to the extinction [12]. A 3D reconstruction always starts with a some are some setup camera, some research some computer vision hardware such as stereo camera and Kinect. In addition to the vision setup, Kinect camera, a proved method which can be common vision setup that can be found in the literature review. In many cases stereo camera setup can be found among the literature review. In addition to the stereo camera, custom stereo camera is quite popular among the researchers by combining two equals web camera that positioned by period of distance [12].

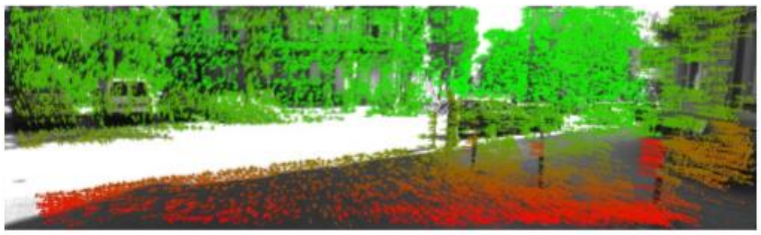
Some algorithms found in the literature review introduced the usage of single and multiple images approaches to perform 3D reconstruction where everyone has their own pros and cons.

- Single Camera : easy to calibrate, computationally efficient more compact. but they lack the depth information. It requires prior knowledge from other sensor to determine the depth scale [13].
- Stereo Camera: In this technique the images captured either using 2 equals web camera [14] or any cameras. They are set by a defined distance. In addition to 2 images captured, an algorithm is used to generate depth map. However, stereo matching has several issue when the scene contains weekly textured areas, repetitive patterns or occlusions occur in both indoor and outdoor environments [15].
- Kinect / Structured Light / Time of Flight: Structured Light sensor is able to perform range detection; an accurate distance measurement is the output [16].

- Fusion: there are researchers who also tried possibilities of using fusion approach whereas combining depth map produced by Stereo and Kinect camera to achieve higher accuracy in depth map precision. To such development allows to produce better 3D Reconstruction object, rich in features details. Range cameras are low cost and ease to use to construct 3D point clouds in real time. One issue arise is that the transparent and reflective surfaces [17].

This paper gave us some light on some camera setups and some method which we can use in our project

“StereoScan: Dense 3d Reconstruction in Real-time” This paper was one of the most impressing, the system they proposed an approach to create a 3D maps from high-resolution stereo sequences in real time. In simple words they have proposed an application which can be used online for robots or cars which highly depend on real-time reconstruction of images [4]. They have two worker thread (two point of view) at real time these two threads provide images out of which they obtain Ego motion (Ego motion is defined as the 3D motion of a camera within an environment). estimates and disparity maps in parallel. While this thread is working of the positions of the objects present in the environment of the systems, the stereo matching and 3d reconstruction parts runs at low fps.
I really want to highlight these two images from paper


(a) Feature matching (2 frames, moving camera)

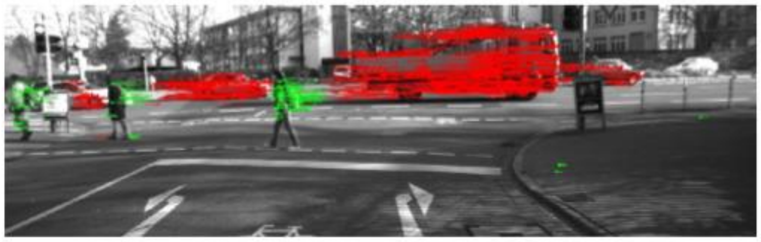

(b) Feature tracking (5 frames, static camera)

Figure 1.0

- a. First image gives output of a moving camera as you can see not only the objects which are moving creates a feature points, but the environment too creates too. This way it is easier to gather information about environment which we want to create as a 3D reconstruction.[4]

They created a 3d reconstruction pipeline consists of four stages: Sparse feature matching, ego motion estimation, dense stereo matching and 3d reconstruction. I would really like to turn towards this paper if I want to generate accurate dense 3d reconstructions from stereo sequences.

"Real-Time 3D Reconstruction and 6-DoF Tracking with an Event Camera" paper proposes a method which can perform real-time 3D reconstruction from a single hand-held event camera with no additional sensing and works in unstructured scenes of which it has no prior knowledge. This paper has a really nice way to put out how do we use a active methods like a event camera in 3D reconstruction. This one was very impressive too one the part I was very much impressed was that they were just using only one camera, but they were using a event camera(An event camera, also known as a neuromorphic camera, silicon retina or dynamic vision sensor, is an imaging sensor that responds to local changes in brightness) this camera is one of the expensive technology to work with.[6] The core of their system is three interleaved probabilistic filters, each estimating one unknown aspect of this challenging Simultaneous Localization and Mapping (SLAM) problem.[6] The system in this paper is highly depend on the camera and they have worked around this event camera for all the detailed they wanted e.g.

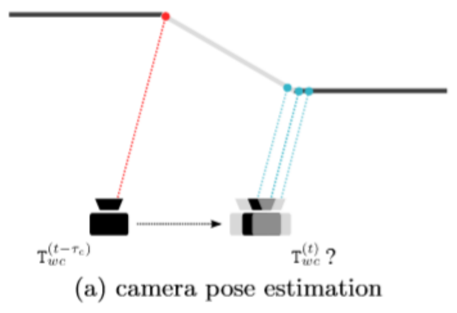


Figure 1.1

(a)based on the assumption that the current log intensity estimate (shown as the color of the solid line) and inverse depth estimate (shown as the geometry of the solid line) are correct, they find current camera pose, they compare this to previous events and estimate[6]

As we can see they have created a nice system but as my understanding they are creating their feature data from the event camera like the pose estimation then depth of the object. They created one of the first 6-DoF tracking and 3D reconstruction method purely based on a stream of events with no additional sensing, and it runs in real-time on a standard PC. This system was one of the most advance will using an active methods

“3D-Reconstruction of Soccer Scenes”By C. Malerczyk Dept. Visual Computing ZGDV Computer Graphics Center Darmstadt, GermanyH. Seibert Dept. Visual Computing ZGDV Computer Graphics Center Darmstadt, Germany
As my interest towards game develop, I found this system exciting. They can create a nice system which bridge 3D reconstruction wit gaming.The aim of this paper is to demonstrate the 3d reconstruction of human body poses and motions in the field of sports events.

A game scene is reconstructed as a 3dimensional model and the user can look at this model not only in slow motion, but also from arbitrary positions and viewing angles of his/her choice. He/she can navigate through the scene assuming various viewing angles and positions.[8] The application is very impressive as they are creating a world where a person can relive any moment of a game by, he/she is being a part of the event too. This type of systems can be divided into two type major domain

1. The reconstruction of sport scenes and motion recognition
2. The visualization of immersive 3D environments on stationary and mobile devices.[8]

For stage one we must create a 3D reconstruction of the environment in which the soccer match takes place or in which it is going to be visualized can mainly be performed prior to the actual match. This is one of an easy thing because the area remains same though the game. More than a sufficient geometric accuracy, the surface color (textures) is crucial for a realistic impression of the reconstructed scene.[8]

The main purpose of the three-dimensional reconstruction is to break through the barrier of standard two-dimensional television. For a 3d reconstruction short but important scenes of a soccer match are selected (clips-of-interest) to provide additional information like for example the trajectory of the ball or the detailed and natural reproduction of the movements of all player, which are involved in the original scene.

Pose Adaptation. The core technology of the reconstruction of sports scenes is a new developed video-based algorithm for the three-dimensional reconstruction of human movements from two-dimensional camera sequences. [8]

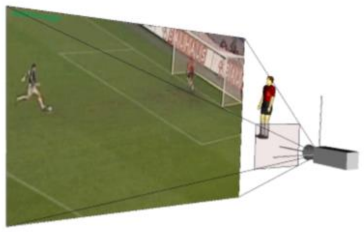


Figure1.2(Principle of the silhouette based 3d pose adaptation.[8])

A virtual copy of the original broadcasting camera with the same parameters (known by the calibration process of the camera) is used. The pixel-wise comparison of both silhouettes leads to the output value of a residual error function, which is minimized using a nonlinear optimization algorithm. For the online generation of the synthetic silhouettes during the pose adaptation process the open source scene graph system OpenSG [3] is used. There are stilling some process like Nonlinear Optimization, and Residual Function which are more like optimizing the model which can differ application to application. This paper really helped to understand some of the 3D reconstruction which can be used for gaming purpose.

Figure 1.3 Some reconstructed scene from FIFA world cup

"Real Positioning in Virtual Environments Using Game Engines" By Rosario De Chiara, Valentina Di Santo, Ugo Erra, Vittorio Scarano ISISLab, Dipartimento di Informatica ed Applicazioni, Again, I landed on this paper because of my game development side. This one is little far away from typical 3D reconstruction by I can be useful because I wanted to use game engines in, y project to help with 3d reconstruction because today they are on the most powerful application in computer graphics [7] Immersive virtual environments offer a natural setting for educational and instructive experiences for users, and game engine technology offers an interesting, cost-effective and efficient solution for building them. In this paper they describe an ongoing project whose goal is to provide a virtual environment where the "real" location of the user is used to position the user's avatar into the virtual environment.[7] Yet, this project was in process and yet I didn't find about the finish project I will reference to this one meanwhile.

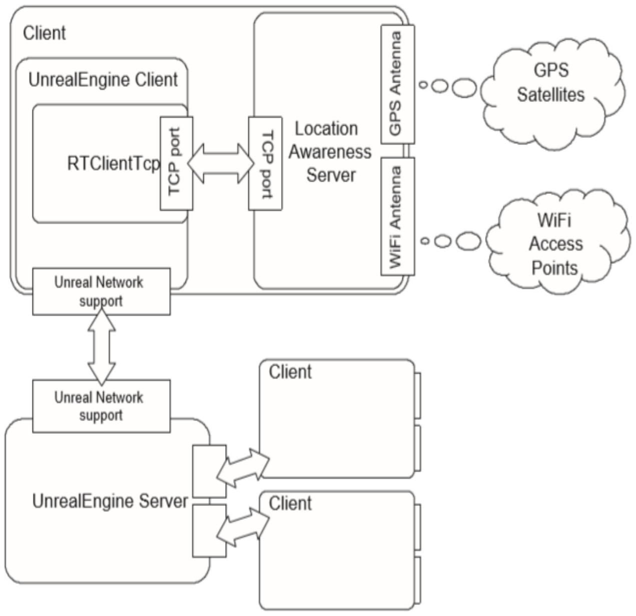


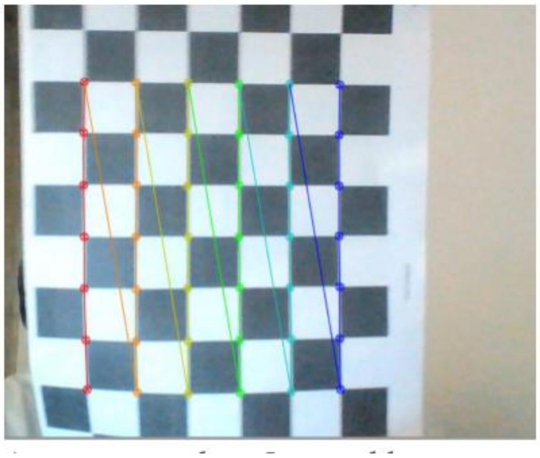
Figure 1.4 This is the architecture of the system.[7]

They have proposed to use to build with a system with a software I'm very much familiar with Unreal Engine. They have briefly described how they will manipulate Game engines to help them set up the game. System like this shows us that we still have a lot to do games not only which VR, but we can also focus it toward 3D reconstruction too. 3D reconstruction has not only endless methods but endless application too, yet it is one of those thing which still needs development.

3. Methodology and experiment

At this point we know how a 3D reconstruction system look like we have also investigated some preexisting system which are generate good results. In this section we are going to talk about the methods I have chosen for my system and why I chose it.

First as we need a calibration for our camera. As we are choosing to go forward with stereo reconstruction, we need our camera calibrated. As we can see how I can calibrate camera to detect a chess pattern and the points in they as we can see I have marked each points different why? because we can now measure distance



(a) No detection (b) chessboard detected

In the above image I have used same algorithm that we will later talk about it in later section, I used findChessboardCorners algorithm which will return the corners detected and a flag called ret that will be true if the algorithm was able to detect the pattern.
Not only this algorithm is very efficient it is also very modular and I wanted modular design for my project.

Stereoscopic vision is referring to us human being having ability to view with both eyes in similar, but slightly different ways. Retinal images are fused in the brain in a way that their disparities (or parallaxes) are transformed into depth perception, yielding a three-dimensional representation of the object in the observer’s mind. Having this vision allows us to see world in 3D but also help us to judge distance which helps us to build an ability to have true depth perception. the human’s ability to view the world through stereoscopic sight has given him/her a significant advantage over entities and animals in the wild that do not possess this capability[i]. Not only they gave us advantage our other animals, but it helped human to evolve faster.

There are many ways to reconstruct the world around, but it all reduces to getting an actual depth map. A depth map is a picture where every pixel has depth information. It is normally represented like a grayscale picture.

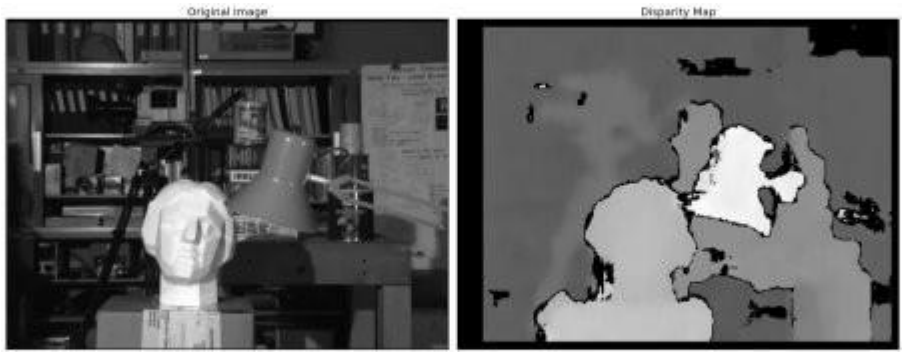


Figure 1.5(depth map from the Tsukuba dataset. Courtesy of OpenCV)

As mentioned before there are different ways to obtain a depth map and these depend on the sensor being used. A type of sensor could be a simple camera, but it is possible to use others like LiDAR or infrared or a combination. The type of sensor will determine the accuracy of the depth map. In terms of accuracy it normally goes like this: LiDAR > Infrared > Cameras. Depth maps can also be colorized to better visualize depth.

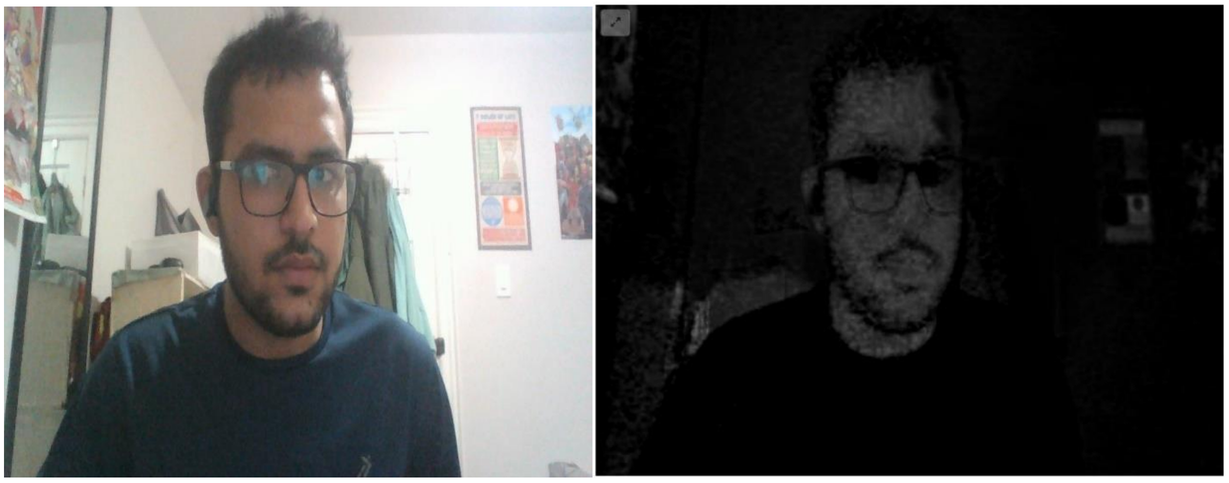


Figure 1.6 A selfie of me taking a depth selfie using an infrared camera

An infrared camera is a non-contact device that detects infrared energy (heat) and converts it into an electronic signal, which is then processed to produce a thermal image or video, on which you

can perform temperature calculations. But we can use systems like Kinect camera which in fact is an infrared, but it uses to measure depth. Kinect is Microsoft's motion sensor add-on for the Xbox 360 gaming console. The device provides a natural user interface (NUI) that allows users to interact intuitively and without any intermediary device, such as a controller[3]. As we can see how easy we can just generate a depth map using an active method like infrared camera, but everyone doesn't have a luxury of having one as they are very expensive or we have to work smart and use something which is very accessible like a phone camera. In that case I will do a stereo reconstruction.
The process of looking at a model from two different angles like our eyes we look for the similarities in both pictures to calculate the depth, this is called a stereo matching.

Stereo matching, also known as Disparity mapping, is an important subclass of computer vision. Curiosity, the mars rover, uses Stereo matching. So do Google's new self-driving cars, as well as quadcopters, helicopters, and other flying vehicles. Stereo matching is robust and fast because it only uses cameras. It is easy to set up yourself, and not too hard to get running. Stereo matching is a subclass of a wide range of algorithms to find depth. It's the simplest way to find depth with two rectified images.
Stereo matching works by finding corresponding points in rectified images. The lighter parts of the disparity map are closer, the darker are farther away. These maps are great because you can see details you may not have noticed before. On the right of the image, the painting is at a tilt, which I could not see that with just the plain image.

In order to do stereo matching it is important to have both pictures have the exact same characteristics. Put differently, both pictures shouldn't have any distortion. This is a problem because the lens in most cameras causes distortion. This means that in order to accurately do stereo matching one needs to know the optical centers and focal length of the camera.

[3] https://searchhealthit.techtarget.com/definition/Kinect

System architecture

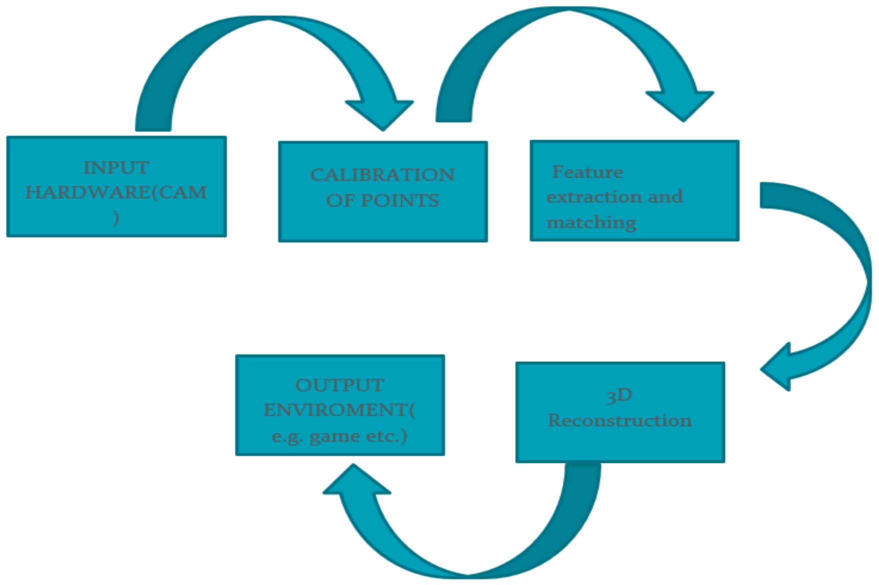

INPUT HARDWARE(CAM)
CALIBRATION OF POINTS
Feature extraction and matching
3D Reconstruction
OUTPUT ENVIROMENT(e.g. game etc.)

Figure 1.7

4. Design and Setup

As talk in before section how I choosing to stereo 3D reconstruction so we will be using a phone a handheld device for our project here.

Camera calibration

As a lens in camera may content distortion, which is very big issue for as it deceases our model's accuracy very much and so we need to current this. Before correcting though we need to know the internal parameters of the camera we are using. Here we a OpenCV algorithm comes in play. Two major kinds of distortion are radial distortion and tangential distortion. Radial distortion causes straight lines to appear curved. Radial distortion becomes larger the farther points are from the center of the image. As phase one of project we will calibrate our camera

Setup

For over system to properly calibrate the camera, it needs a good number of samples of the same pattern. Here we are not dealing with video where we analyze every frame as the user moves the pattern in front of the camera until the pattern is detected several times. Since we're not dealing with a video feed, we must take several pictures of the same pattern instead.
Here the sample we used for the system.

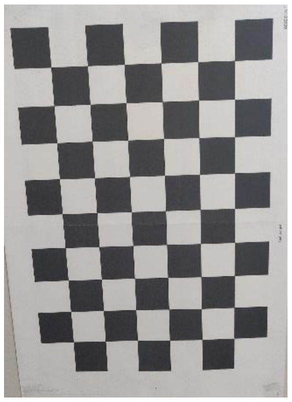


Figure 1.8

[1]

Now Camera setup, a setup is very important because we don't want any unwanted noise, distortion or any kind of extenuation disputations which can cause will we are extracting our parameters which are need for reconstruction. I noticed that the more stuff in the background the longer it took to calculate the camera matrix. Therefore, I used a white wall and pasted the chessboard pattern on it. Make sure it is perfectly flat.

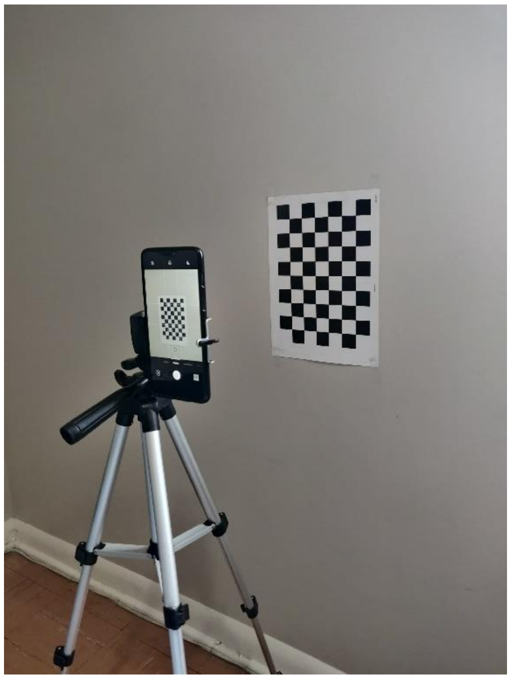


Figure 1.9

The fig 1.9 show my setup where I used a tri pod to get better results with better quality with much of noise. One the points of using tri pods was consist it merely one option we can skip but for consists I choose using one so I can focus on picture rather worrying about camera not focusing on chessboard. Once this is setup, we will start taking pictures.

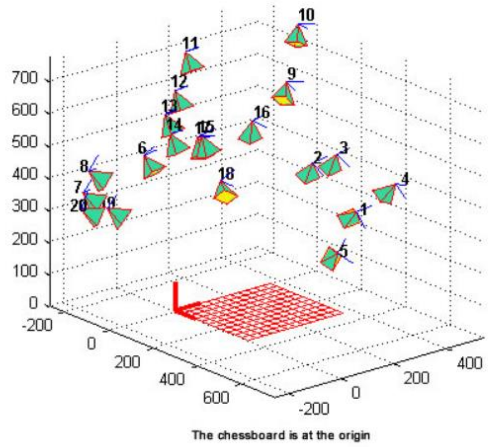


Figure 2.0

There no specific number that we are needed to take pictures as not all pictures will be suitable for detecting the pattern. I took 70.

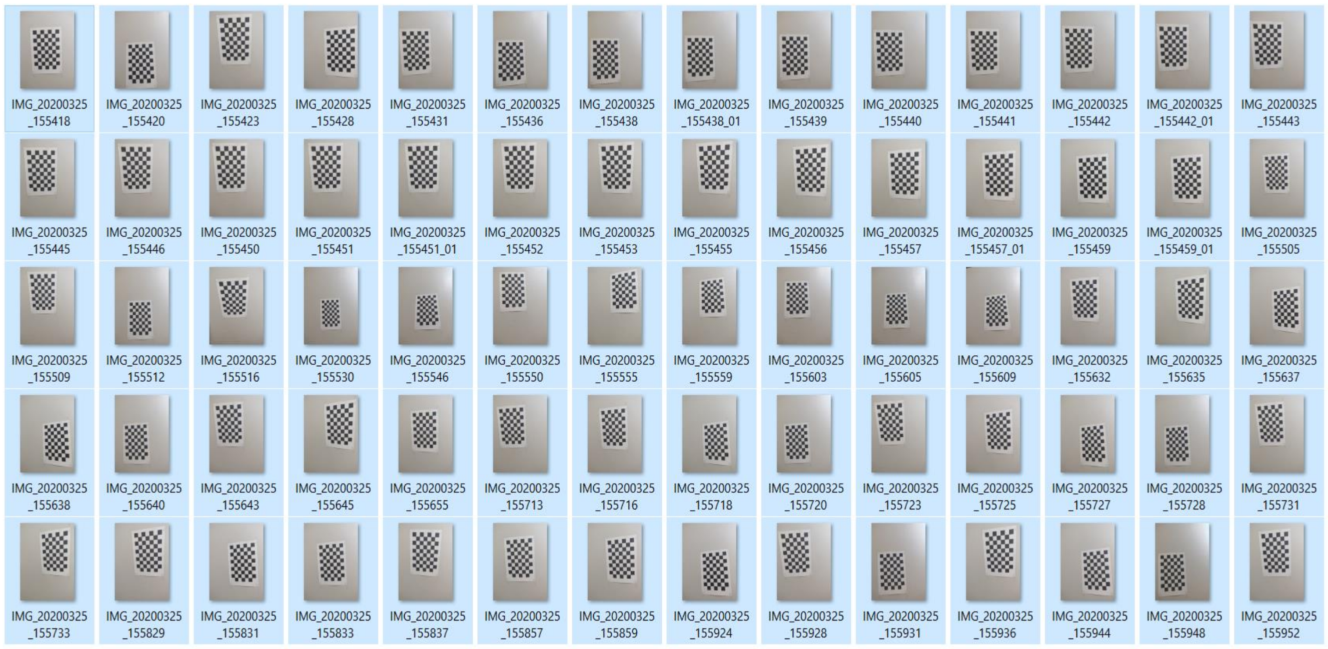
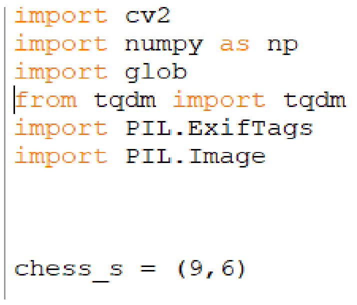


Figure 2.1

The above fig 2.1 is my picture which I took to calibrate my camera. This will conclude our step for camera calibration.

Code

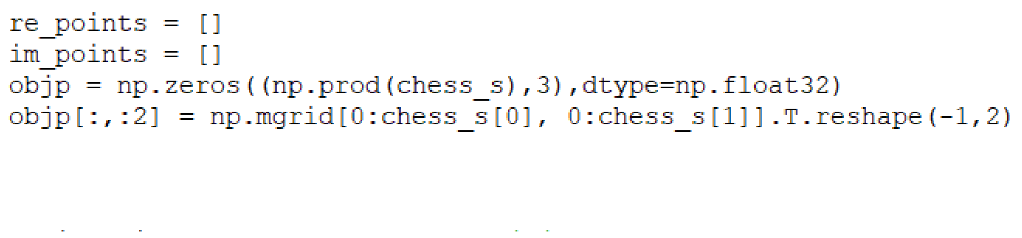
The first step would be importing library we need into ours. Using Python 3.7.1, OpenCV 3.4.1. as well as Numpy, Glob, tqdm and Pillow, and we also need to define our chessboard size in my its 9x6.



```
import cv2
import numpy as np
import glob
from tqdm import tqdm
import PIL.ExifTags
import PIL.Image



chess_s = (9,6)
```

Figure 2.2

Cv2(library in snippet 1.0) is an Open Source Computer Vision (OpenCV) which includes large number of algorithms that we can use in computer vision task like our task to calibrate camera. This library also has algorithms which are used to process image or videos to detect faces, identify objects, classify human actions, track moving objects, color detection, pattern recognition etc. this makes perfect library for our task. The NumPy and glob this are the library we will use to manipulate our data in our case those the image we capture using our camera.



```
re_points = []
im_points = []
objp = np.zeros((np.prod(chess_s),3),dtype=np.float32)
objp[:,:2] = np.mgrid[0:chess_s[0], 0:chess_s[1]].T.reshape(-1,2)
```

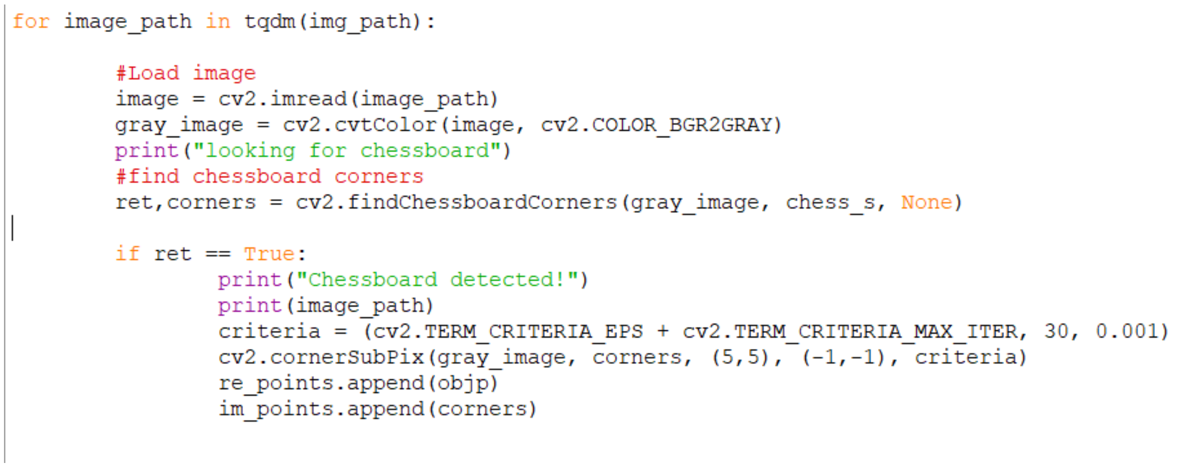
Figure 2.3

In above code we create a list where we will store points in manner of (x, y, z).

“re_points = []” will store 3D points in real world space.

“im_point = []” will store 3D points in image place.

Then we will use Glob method to iteratively go over images and I also added a extra function using tqdm a function which outputs the progress of out loop wit time.



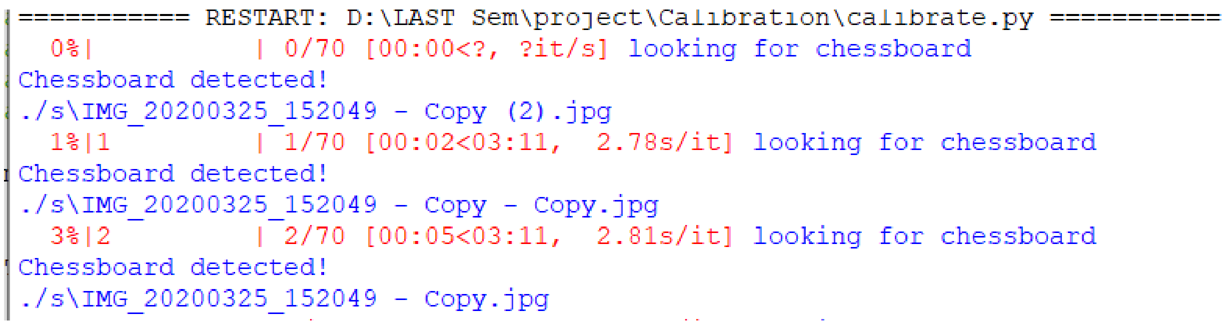
```
for image_path in tqdm(img_path):

        #Load image
        image = cv2.imread(image_path)
        gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        print("looking for chessboard")
        #find chessboard corners
        ret,corners = cv2.findChessboardCorners(gray_image, chess_s, None)

        if ret == True:
                print("Chessboard detected!")
                print(image_path)
                criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)
                cv2.cornerSubPix(gray_image, corners, (5,5), (-1,-1), criteria)
                re_points.append(objp)
                im_points.append(corners)
```

Figure 2.4

In above figure 2.4 the loop is main calibration work. There we are loading our images and one at a time and looking for the chessboard.

INSIDE LOOP

- we load over image into over system.
- Convert into grayscale (easier to work with and faster)
- Then we will be using findChessboardCorners algorithm.
- It will generate the corner which are detected
- RET a flag will also return stating if we found the pattern or no.
- To increase the accuracy, we have to define a "*criteria*" to run the algorithm.
- A criteria in this situation is *"criteria = (type, number of iterations, accuracy)",* so in over code we stated that take iteration and accuracy(refer to figure 2.4 line (*cv2.TERM_CRITERIA_EPS* + *cv2.TERM_CRITERIA_MAX_ITER*)
- And we selected 30 over 0.001 learning rate.
- Then we need to finite over search a window size for that "*cv2.cornerSubPix*" algorithm we will use which will focus on the points it will get the image, the corners a window size, zeroZone and the actual criteria. It will look for corners in window size.



```
=========== RESTART: D:\LAST Sem\project\Calibration\calibrate.py ===========
  0%|          | 0/70 [00:00<?, ?it/s] looking for chessboard
Chessboard detected!
./s\IMG_20200325_152049 - Copy (2).jpg
  1%|1         | 1/70 [00:02<03:11,  2.78s/it] looking for chessboard
Chessboard detected!
./s\IMG_20200325_152049 - Copy - Copy.jpg
  3%|2         | 2/70 [00:05<03:11,  2.81s/it] looking for chessboard
Chessboard detected!
./s\IMG_20200325_152049 - Copy.jpg
```

Figure 2.5

- You can see in figure 2.5 that we are successfully detecting those chessboards.
- After are gone though ever image we will invoke a function called "*cv2.calibrateCamera*" which helps us to extract the camera parameters. This will return camera matrix (K) distortion coefficients (dist) and the rotation and translation vectors (rvecs and tvecs).

Finally, once we have our parameter, we need somewhere to put them so we will use NumPy library again.

```
np.save("./camera_params/ret", ret)
np.save("./camera_params/K", K)
np.save("./camera_params/dist", dist)
np.save("./camera_params/rvecs", rvecs)
np.save("./camera_params/tvecs", tvecs)

exif_img = PIL.Image.open(img_path[0])

exif_data = {
        PIL.ExifTags.TAGS[k]:v
        for k, v in exif_img._getexif().items()
        if k in PIL.ExifTags.TAGS}


focal_length_exif = exif_data['FocalLength']


focal_length = focal_length_exif[0]/focal_length_exif[1]


np.save("./camera_params/FocalLength", focal_length)
```

Figure 2.6

In the above code snippet we can see that I saved those value into different numpy files, one the reasons for that the calibration scrpit took 2 hours to run and if we want to do some recontruction then we haveto rerun code again and again so rather doing that we just made our own file which we can later use anytime we want to.

For best 3D reconstruction we really need 3 parameters.

- The camera matrix
- The distorion coefficeents
- Focal length

As for focal length I decided to extract it from our exif data. Exif data is the data contain in the picture which can contains of focal length, number of pixel, size etc.

```
exif_img = PIL.Image.open(img_path[0])

exif_data = {
        PIL.ExifTags.TAGS[k]:v
        for k, v in exif_img._getexif().items()
        if k in PIL.ExifTags.TAGS}
focal_length_exif = exif_data['FocalLength']
focal_length = focal_length_exif[0]/focal_length_exif[1]
np.save("./camera_params/FocalLength", focal_length)
```

Figure 2.7

In figure 2.7 we have a snippet of code where using exif data we are extractnig focal length and again we will store that number into a NumPy file.
AT this point we have completed over first phase where we have over camera parameters by using libraries from opencv2 and setuping up over camera. I have also calculated over error rate which at this point is 0.17 which is good as its always better to have error near to zero.
Now at this point of system we need to start working on the reconstruction part. Now we will capture images for over input then use them to generate over output 3D reconstruction.

In most stereo applications out there you will find that each picture is taken from two individual cameras like in the image below

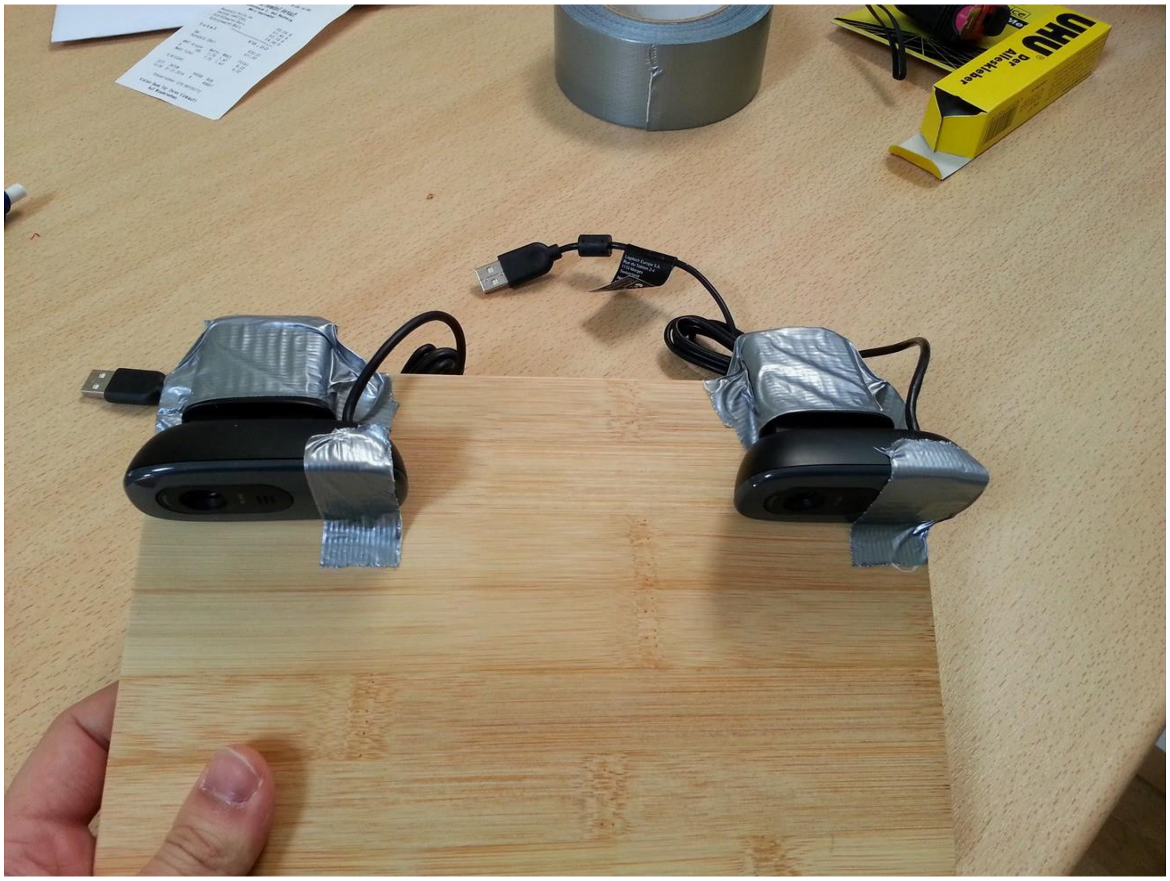


Figure2.8[4]

[4] https://erget.wordpress.com/2014/02/01/calibrating-a-stereo-camera-with-opencv/

As in figure 2.8 we can see that they are trying to repicate human eyes. So lot of people utlilize the above kind of setup but as we have a tripod and we wanted to create as hassle free 3D reconstruction I decide to use my phone camera.

To recreate a human stereo vision I took two picture using a tri pod while just shifty little right after taking one picture.

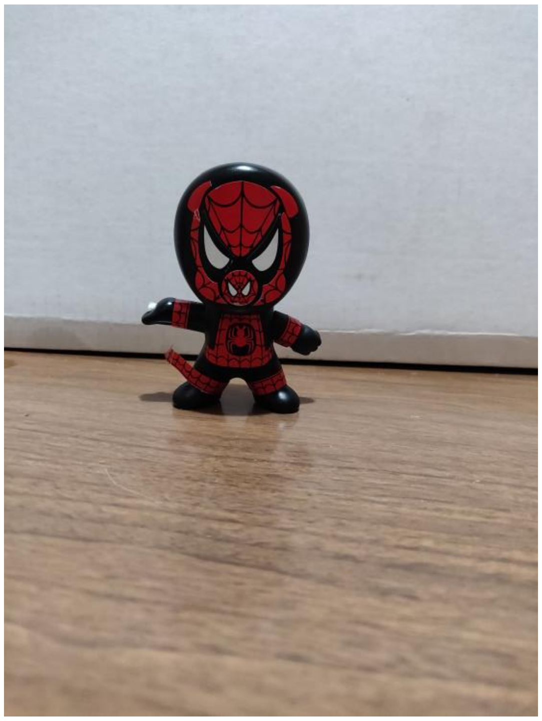


Figure 2.9(a)

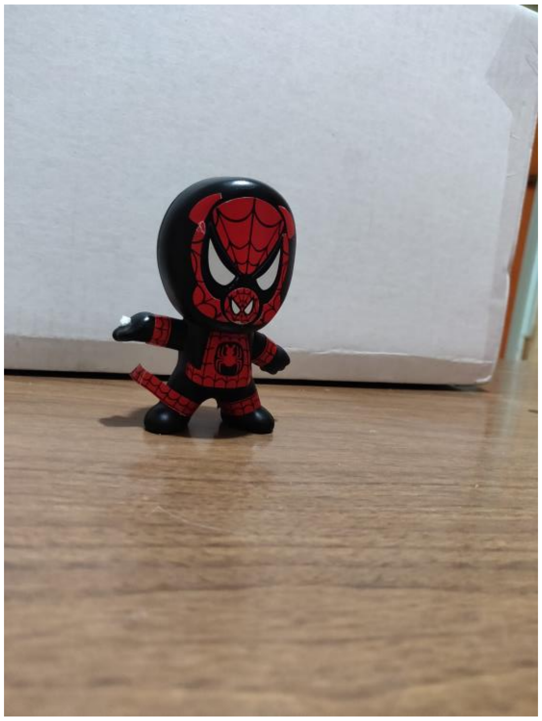
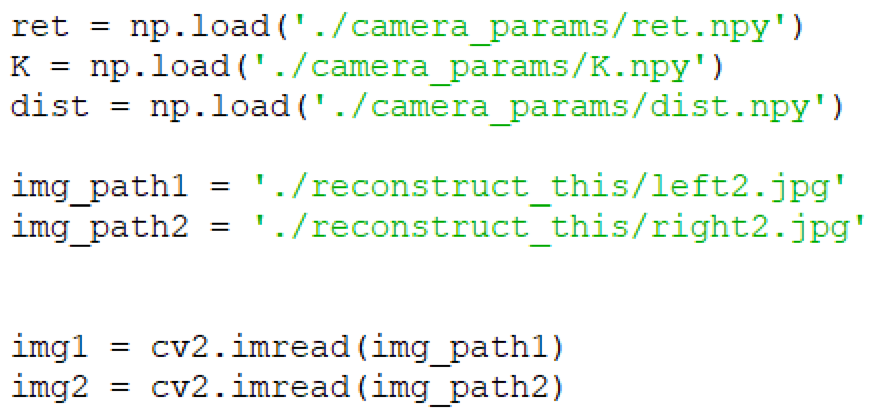


Figure 2.9(b)

The above figures 2.9 (a) suppose to replicate left veiw while (b) is working as right one. For our input this are our images .

Now we have our input, it time to work on those but first we need import over camera parameter which we will use

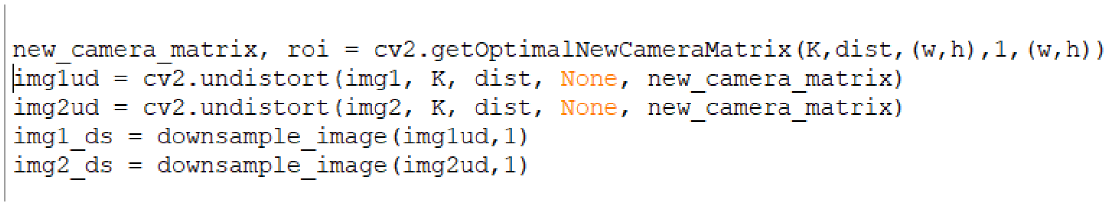
Once we have the pictures taken it is time to write some code. We will start by loading the camera matrix and the pictures shown above. As a friendly reminder, remember that the complete script can be found here.



```
ret = np.load('./camera_params/ret.npy')
K = np.load('./camera_params/K.npy')
dist = np.load('./camera_params/dist.npy')

img_path1 = './reconstruct_this/left2.jpg'
img_path2 = './reconstruct_this/right2.jpg'


img1 = cv2.imread(img_path1)
img2 = cv2.imread(img_path2)
```

Figure 2.1.0

We also imported our images into our system (refer to figure 2.1.0). So, now we calculate a new camera matrix with a free scaling parameter. Because we might change our image size that we use for inputs then over training ones, but we are not fully changing it but the new camera matrix we better for accuracy and works better on removal of distortion.



```
new_camera_matrix, roi = cv2.getOptimalNewCameraMatrix(K,dist,(w,h),1,(w,h))
img1ud = cv2.undistort(img1, K, dist, None, new_camera_matrix)
img2ud = cv2.undistort(img2, K, dist, None, new_camera_matrix)
img1_ds = downsample_image(img1ud,1)
img2_ds = downsample_image(img2ud,1)
```

Figure 2.1.1

So at this point we have completed our preprocessing and also downsample over image just one time as downsample image helps us to to get better processing power speed and it also provide us for better parameter tuning when it comes to building a dispartiy map.

We should mind over image size that we will use in feature matching algorithm, as the algorithm we are using has a specify search window size.

- If the window size is large it will take longer than need to compute our code
- If window size is small then we wont be able to find proper number of disparties and we will end haviong very noisy depth map.

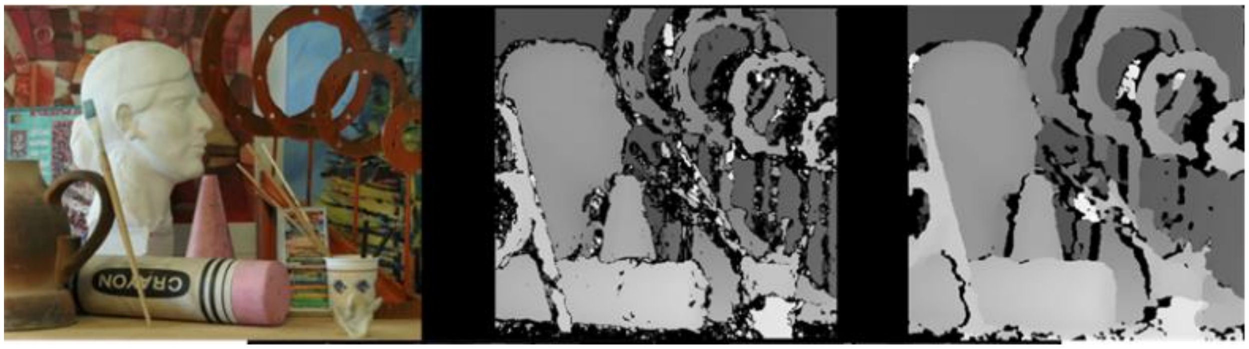
As we are downsampling our input images we are losing some information and beacuase of that over accuracy also suffers.

We have done over preprocessing now it time for matching algorithm to work on them. for our stereo computation we will be using a block matching algorithm.

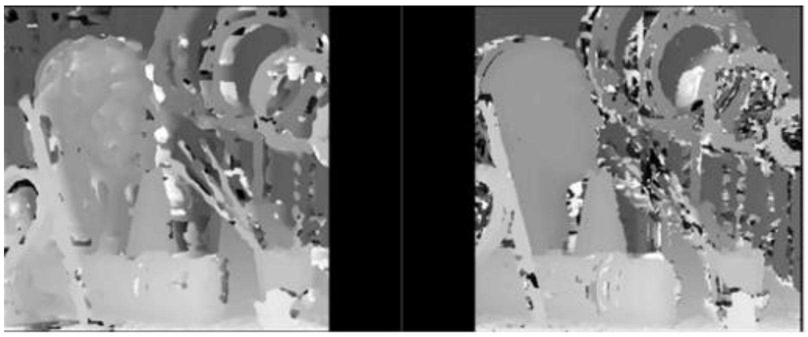
So, we are using a Block matching because we need to match points between two images with same object basically that means the system is will look for the same pixel value in those two pictures.

For getting over block matching done we are having 4 different choices of stereo vision algorithms

1. Block Matching (BM)
2. Semi Global Block Matching (SGBM)
3. Sum of Absolutely Differences (SAD)
4. Normalized Cross Correlation (NCC)



(a)original image (b)BM (c)SGBM



(d) SAD (e)NCC

Figure 3.0

On the above results we can see the difference in each algorithms, without doubt we have having better result from first two BM and SGBM those algorithms are giving us better results with smoother results. As we can also see that block matching focuses on high texture image an image with heavy background like lamp post outside or tree. While SGBM will focuses more on sub pixels level- matching and images which have smooths textures like indoor photos

So, as we are working inside, we will use SGBM as we also have a lot of smooth texture while working indoors.
This SGBM in the system will do the follow steps

- It will prefilter image for normalize brightness and better texture
- Initial a search on x-axis which is horizontal epipolar lines using a SAD window
- After those we will Postfilter those images to delete any bad correspondence matches

As talked beforehand stereo vision is a technique aimed at counting depth
from two or more cameras

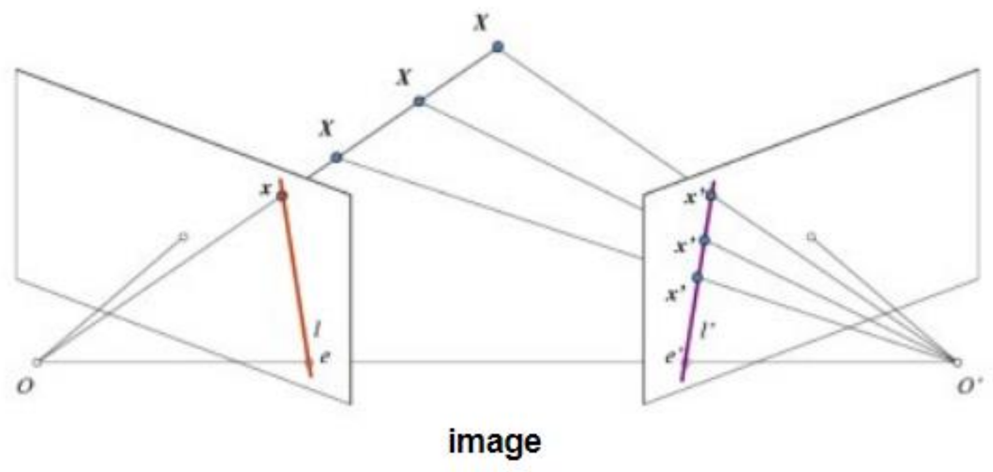
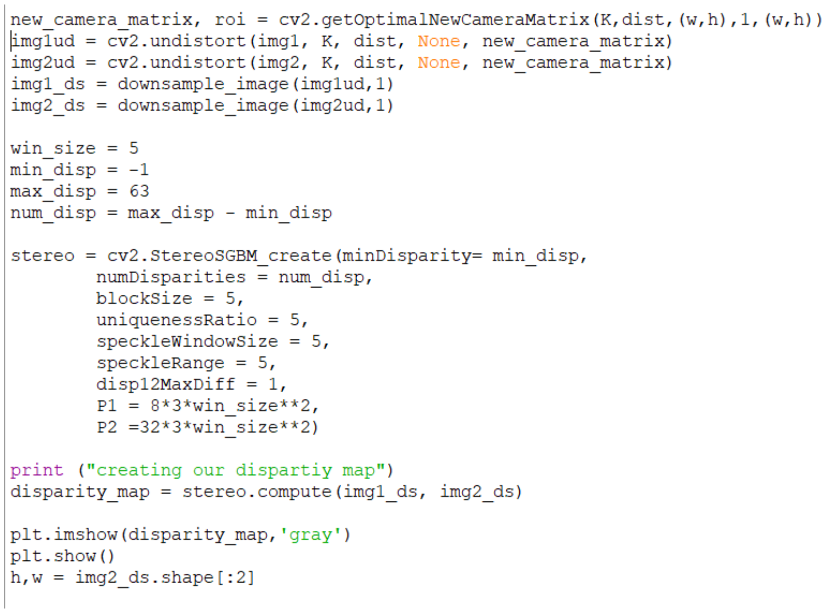


Figure 3.0 Using two or more cameras we can calculate a depth nby means of triangulation, if we are able to fine same pair of pixels in two iamges[5].

So an epipole is a the poitn of the intersection of the line joining the camera above those points are are "X" in figure 3.0. The plane which is created by this baseline are called epipolar plane which is one-parameter. And the line which intersection of an epipolar plane with the image plane is called an epipolar line while all epipolar lines intersect at the epipole.
Epipole are very important as after undistorting the input the epipolar line will be horizontal as we are using SGBM this algorithm will traverse them to find over match.

[5] https://docs.opencv.org/3.4.4/da/de9/tutorial_py_epipolar_geometry.html

Once we all the preprocessing out of the way we will work on the disparity. A disparity if an offset for a point. We will give our system the minimum and maximum disparities, at the end we must subtract them you calculate the number of disparities, which is a way to specify the acceptable range for which pixels can move in the picture.
The final step will doing some of the post processing, as the feature matching algorithm has good chance to have a false positive. To correct them we investigate OpenCV which has a uniqueness ratio which is a threshold for the match value.
Then as we might have some issues with block-matching near the boundaries of objects, which is why we get a region of lots of small disparities called “speckles”. To overcome they we must set a speckle window, which the area to which we accept these “speckles”.
In the SGBM algorithm, there is a parameter called disp12MaxDiff which specifies the maximum allowed difference between the disparities calculated from left to right and those calculated from right to left.



```
new_camera_matrix, roi = cv2.getOptimalNewCameraMatrix(K,dist,(w,h),1,(w,h))
img1ud = cv2.undistort(img1, K, dist, None, new_camera_matrix)
img2ud = cv2.undistort(img2, K, dist, None, new_camera_matrix)
img1_ds = downsample_image(img1ud,1)
img2_ds = downsample_image(img2ud,1)

win_size = 5
min_disp = -1
max_disp = 63
num_disp = max_disp - min_disp

stereo = cv2.StereoSGBM_create(minDisparity= min_disp,
        numDisparities = num_disp,
        blockSize = 5,
        uniquenessRatio = 5,
        speckleWindowSize = 5,
        speckleRange = 5,
        disp12MaxDiff = 1,
        P1 = 8*3*win_size**2,
        P2 =32*3*win_size**2)

print ("creating our dispartiy map")
disparity_map = stereo.compute(img1_ds, img2_ds)

plt.imshow(disparity_map,'gray')
plt.show()
h,w = img2_ds.shape[:2]
```

Figure 3.1

Mine disparity map looked like

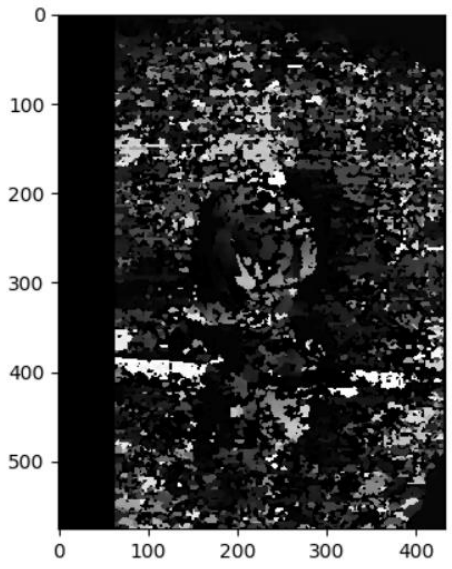


Figure 3.2

As we can see mine disparity map has lots of dead points and speckles in the area. This is because tuning the SBGM parameters very well. We have tune it at every possible time we run them.

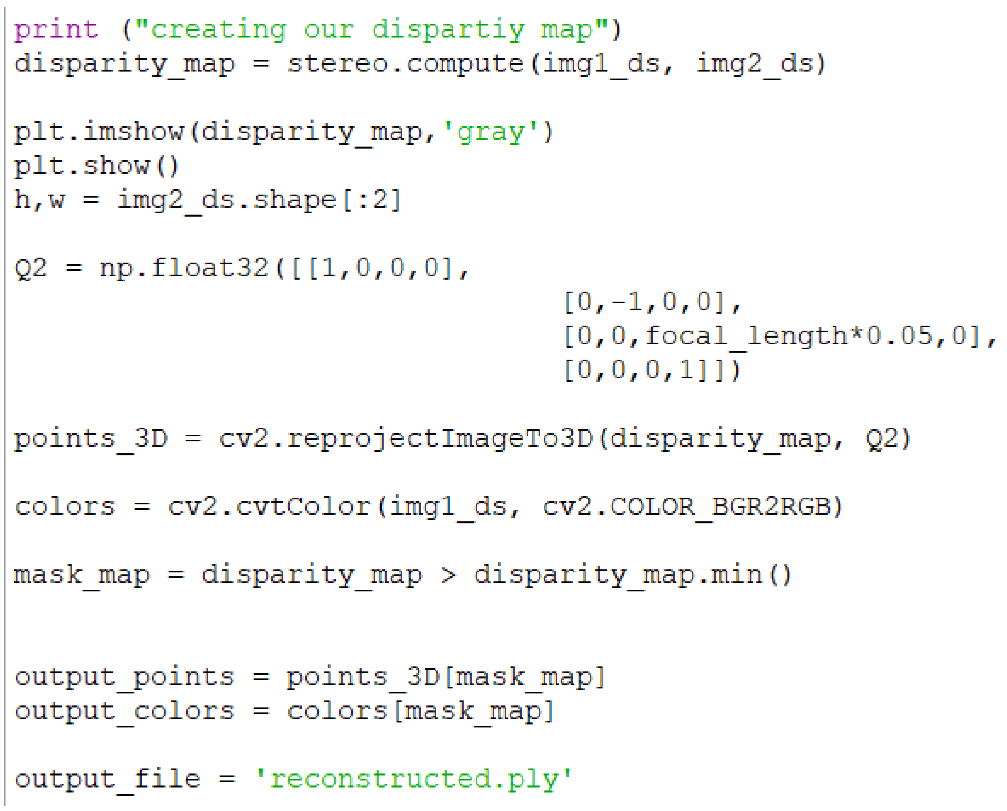
So, at this point we have a disparity map and now as we down sampled the image and turned into gray scale, we will get an array of colors used image and we will also extract height and width too. Now we will get a transformation matrix. The below is one those which I used in the system.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f' & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z/f' \end{bmatrix} \Rightarrow \left(f'\frac{x}{z}, f'\frac{y}{z}\right)$$

divide by the third coordinate

Figure 3.3[6]

[6] https://ags.cs.uni-kl.de/fileadmin/inf_ags/3dcv-ws14-15/3DCV_lec01_camera.pdf



```
print ("creating our dispartiy map")
disparity_map = stereo.compute(img1_ds, img2_ds)

plt.imshow(disparity_map,'gray')
plt.show()
h,w = img2_ds.shape[:2]

Q2 = np.float32([[1,0,0,0],
                                [0,-1,0,0],
                                [0,0,focal_length*0.05,0],
                                [0,0,0,1]])

points_3D = cv2.reprojectImageTo3D(disparity_map, Q2)

colors = cv2.cvtColor(img1_ds, cv2.COLOR_BGR2RGB)

mask_map = disparity_map > disparity_map.min()


output_points = points_3D[mask_map]
output_colors = colors[mask_map]

output_file = 'reconstructed.ply'
```

Figure 3.4

As an output we generate a point cloud which we will extract the into a text file with a specific header that is saved as a .ply file. And using mesh lab we will open it.

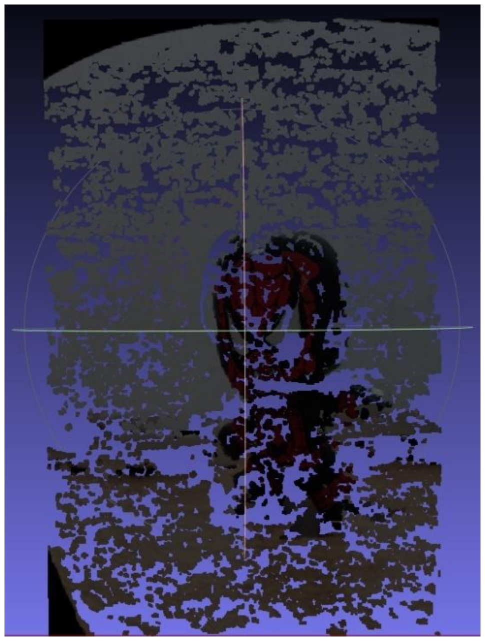


Figure 3.5

5. Conclusion and Discussion

3D reconstruction has not only endless methods but endless application too, yet it is one of those thing which still needs development. From working in phrases like initial images collection to calibrating, to extracting camera parameters into a NumPy file to create better functionality. Then the second phase where we undistort the image and load them into SGBM algorithm where traversing horizontally on epipolar lines we find our match disparity while getting rid of the speckles and then finally creating a disparity map. Using that disparity and a transformation matrix we create a point cloud. Finally, that point cloud will be imported into MeshLab where we will generate the 3D model. One the thing that motivated me was not enough number of tutorial on this topic so I wanted to take on this topic so I can explore and learn something new

Some of the future Work will be on accuracy of 3d model as right our model is highly inaccurate it's hard to put a now seven-point 3D accuracy which is highly used on 3D reconstruction. Implementation of GUI and calibrate and by fine tuning the parameters in the SGBM algorithm for better results. While working on this project I came across many system but they we were high standard either scientist or certain number of people created those system which are either very early stage or something we use daily like a MRI machine. 3D reconstruction has always been seen as a topic of research as we are walking towards better technology it is very unclear that what will discover in near future.

Refences

[1] Xiao, Jianxiong. “3 D reconstruction is not just a low-level task : retrospect and survey.” (2013).
[2] F. Simões et al., "Challenges in 3D Reconstruction from Images for Difficult Large-Scale Objects: A Study on the Modeling of Electrical Substations," 2012 14th Symposium on Virtual and Augmented Reality, Rio Janiero, 2012, pp. 74-83.
[3]Reiners, D.: OpenSG PLUS: Advances to Current Scenegraph Technology. In: Federal Ministry of Education and Research: Virtual and Augmented Reality Status Conference 2004. Proceedings CD-ROM. Leipzig, 2004.
[4] Sing, Keng & Xie, Wei. (2016). Garden: A Mixed Reality Experience Combining Virtual Reality and 3D Reconstruction. 180-183. 10.1145/2851581.2890370.
[5] Etienne Mouragnon, Maxime Lhuillier, Michel Dhome, Fabien Dekeyser, Patrick Sayd. Real-Time Localization and 3D Reconstruction. 2006, pp.0. hal-00091145
[6] Real-Time 3D Reconstruction and 6-DoF Tracking with an Event Camera”
Hanme Kim, Stefan Leutenegger, and Andrew J. Davison
Departmentof Computing, Imperial College London, UK
{hanme.kim,s.leutenegger,a.davison}@imperial.ac.uk
[8] C. Malerczyk, "3D-Reconstruction of Soccer Scenes," 2007 3DTV Conference, Kos Island, 2007, pp. 1-4.doi: 10.1109/3DTV.2007.4379486URL:
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4379486&isnumber=437 9375
[9]Liu-Shuang, Joan & Torfs, Katrien & Rossion, Bruno. (2015). An objective electrophysiological marker of face individualisation impairment in acquired prosopagnosia with fast periodic visual stimulation. Neuropsychologia. 83. 10.1016/j.neuropsychologia.2015.08.023
[10] A. O. Ulusoy, A. Geiger and M. J. Black, "Towards Probabilistic Volumetric Reconstruction Using Ray Potentials," 2015 International Conference on 3D Vision, Lyon, 2015, pp. 10-18.
[11] A. A. Soltani, H. Huang, J. Wu, T. D. Kulkarni and J. B. Tenenbaum, "Synthesizing 3D Shapes via Modeling Multi-view Depth Maps and Silhouettes with Deep Generative Networks," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 2511-2519
[12] Ham, Hanry & Wesley, Julian & Hendra, Hendra. (2019). Computer Vision Based 3D Reconstruction : A Review. International Journal of Electrical and Computer Engineering (IJECE). 9. 2394. 10.11591/ijece.v9i4.pp2394-2402.
[13] F. Santoso, M. Garratt, M. Pickering, and M. Asikuzzaman, “3D-Mapping for Visualisation of Rigid Structures: A Review and Comparative Study,” IEEE Sensors Journal, vol. PP, no. 99, pp. 1–1, 2015. [Online]. Available:
http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7322186
[14] A. Harjoko, R. M. Hujja, and L. Awaludin, “Low-cost 3D surface reconstruction using Stereo camera for small object,” 2017 International Conference on Signals and Systems (ICSigSys), pp. 285–289, 2017. [Online]. Available:
http://ieeexplore.ieee.org/document/7967057/
[15] G. D. Evangelidis, M. Hansard, and R. Horaud, “Fusion of Range and Stereo Data for High-Resolution Scene-Modeling,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 37, no. 11, pp. 2178–2192, 2015.
[16] G.-v. J. M and M.-v. J. C, “Simple and low cost scanner 3D system based on a Time-of-Flight ranging sensor,” pp. 3–7, 2017.

[17] R. Ravanelli, A. Nascetti, and M. Crespi, “Kinect V2 and Rgb Stereo Cameras Integration for Depth Map Enhancement,” ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial InformationSciences,vol.XLI-B5,no.July,pp.699–702,2016.[Online].Available: http://www.int-archphotogramm-remote-sens-spatial-inf-sci.net/XLI-B5/699/2016/isprs-archives-XLI-B5-699-2016.pdf

[i] https://www.tech-faq.com/stereoscopic-vision.html