Autonomous Drone Follow

Unmanned Aerial Vehicles

Task

- Accurate depth calculation
- Minimum Computation
- Maximum FPS

Literature Survey

Measuring Distance with Mobile Phones Using Single-Camera Stereo Vision

Abstract—Computer stereo vision is an important technique for robotic navigation and other mobile scenarios where depth perception is needed, but it usually requires two cameras with a known horizontal displacement. In this paper, we present a solution for mobile devices with just one camera, which is a first step towards making computer stereo vision available to a wide range of devices that are not equipped with stereo cameras. We have built a prototype using a state-of-the-art mobile phone, which has to be manually displaced in order to record images from different lines of sight. Since the displacement between the two images is not known in advance, it is measured using the phone's inertial sensors. We evaluated the accuracy of our single-camera approach by performing distance calculations to everyday objects in different indoor and outdoor scenarios, and compared the results with that of a stereo camera phone. As a main advantage of a single moving camera is the possibility to vary its relative position between taking the two pictures, we investigated the effect of different camera displacements on the accuracy of distance measurements.

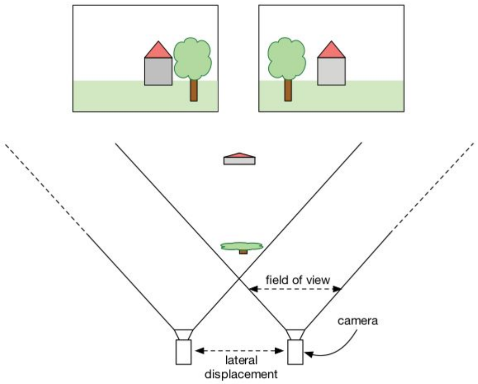
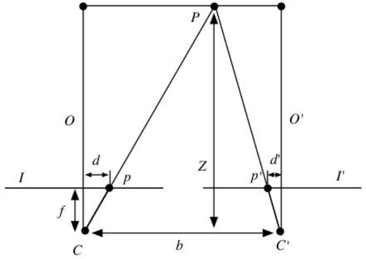
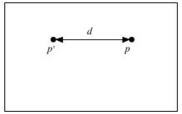


Figure 1. Two images recorded from a laterally displaced camera. The displacement of the objects provides relative depth information.



(a) Geometry of stereo triangulation.



(b) Overlay of the images shows the difference between the image locations p and p'.

Maths Used

Robert Collins
CSE486, Penn State

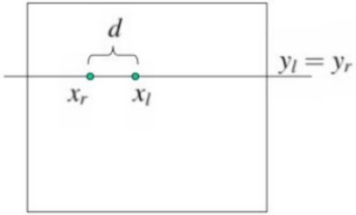
Recall: Stereo Disparity

Left camera

$$x_l = f\frac{X}{Z} \qquad y_l = f\frac{Y}{Z}$$

Right camera

$$x_r = f\frac{X - T_x}{Z} \qquad y_r = f\frac{Y}{Z}$$



Stereo Disparity

$$d = x_l - x_r = f\frac{X}{Z} - \left(f\frac{X}{Z} - f\frac{T_x}{Z}\right)$$

$$d = \frac{f\,T_x}{Z}$$

depth baseline

$$Z = \frac{f\,T_x}{d}$$

disparity

Important equation!

Triangle Similarity for Object/Marker to Camera Distance

In order to determine the distance from our camera to a known object or marker, we are going to utilize *triangle similarity*.

As I continue to move my camera both closer and farther away from the object/marker, I can apply the triangle similarity to determine the distance of the object to the camera:

D' = (W x F) / P

1.1 What is the range of the Depth measured by the camera?

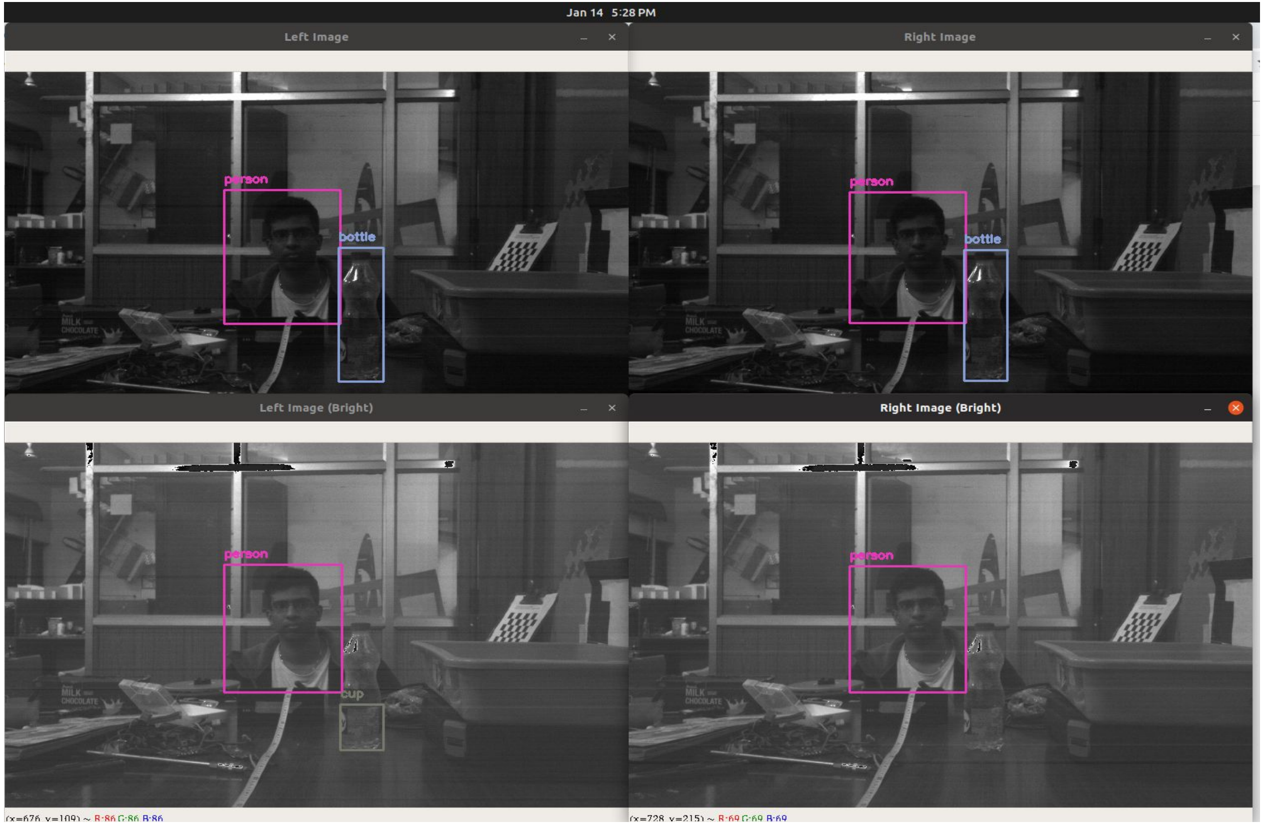
Depth Range is from **50 cm to 300cm**. The maximum working distance varies with the lens. With our default lens, the maximum working distance is about 300cm, beyond which the depth accuracy drops down

Approach

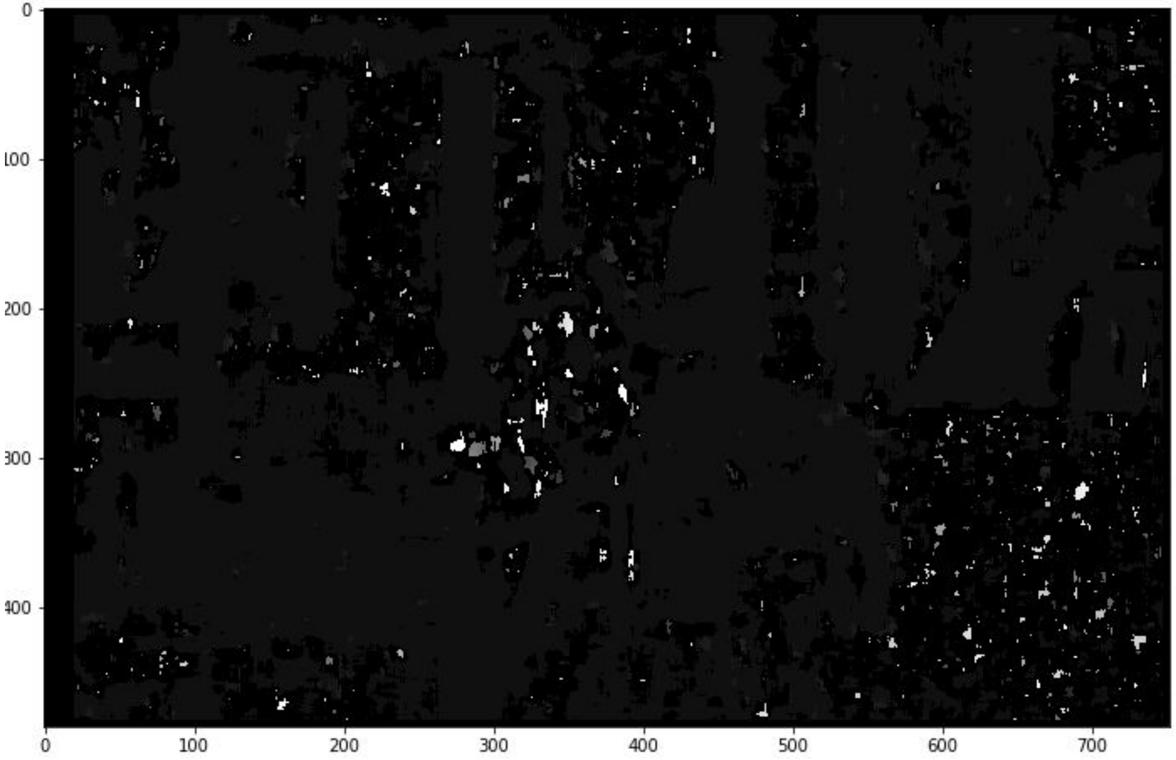
- Object detection with single camera
- Stereo vision camera for depth and width calculation
- Size based depth calculation (if object width known)

```python
def depth():
    for i in range(len(files)):
        imgR = cv2.imread(right + files[i])
        bboxR, labelR, confR = cv.detect_common_objects(imgR)
        if len(bboxR)>0 and labelR[0] == 'person' and flag==0:
            a = int(files[i][5:-4])
            a = a-1
            a = str(a)
            a = '{0}'.format(a.zfill(4))
            imgL = cv2.imread(left + files[i][0:5] + a + ".jpg")
            bboxL, labelL, confL = cv.detect_common_objects(imgL)

            if len(bboxL) == 0:
                print("Object only in Right Camera")
            else:
                x0 = int((bboxR[0][0]+bboxR[0][2])/2)
                x1 = int((bboxL[0][0]+bboxL[0][2])/2)
                d = abs(x1-x0)
                z = (f*baseline)/d
                print(z)

                p = bboxR[0][2] - bboxR[0][0]
                w = (z*p)/f
                flag=1

        elif len(bboxR)>0 and labelR[0] == 'person' and flag==1:
            p = bboxR[0][2] - bboxR[0][0]
            z = (w*f)/p
            print(z)

        else:
            print("No Object")
            flag=0
```

Experiments



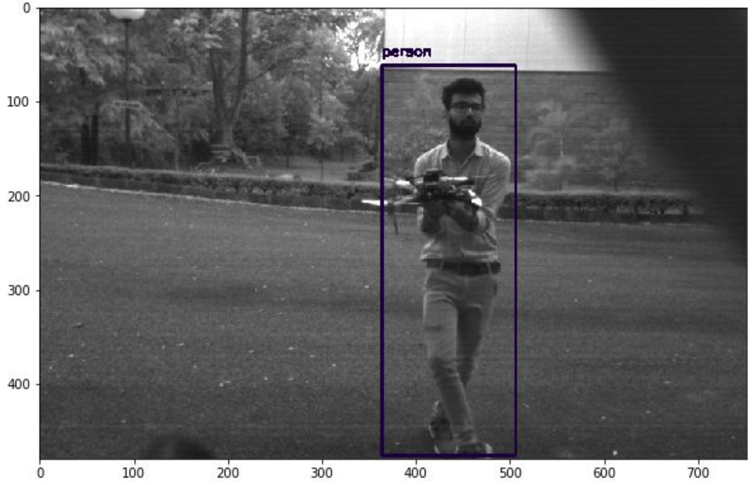
Experiments

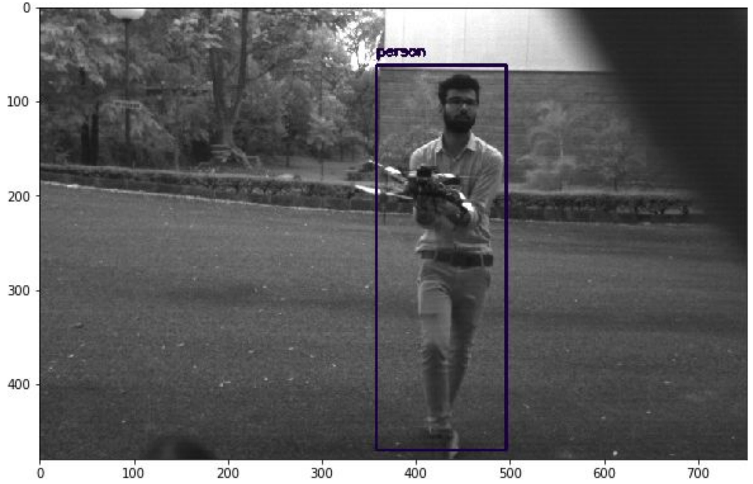



```
array([[-16, -16, -16, ..., -16, -16, -16],
       [-16, -16, -16, ..., -16, -16, -16],
       [-16, -16, -16, ..., -16, -16, -16],
       ...,
       [-16, -16, -16, ..., -16, -16, -16],
       [-16, -16, -16, ..., -16, -16, -16],
       [-16, -16, -16, ..., -16, -16, -16]], dtype=int16)
```

Experiments

```
No Object
No Object
No Object
Object only in Right Camera
No Object
2533.3333333333335
3065.193370165746
2859.7938144329896
3263.529411764706
No Object
No Object
No Object
No Object
No Object
No Object
No Object
No Object
No Object
No Object
No Object
No Object
No Object
No Object
No Object
No Object
No Object
No Object
No Object
Object only in Right Camera
No Object
4560.0
5461.077844311377
5922.077922077922
5629.62962962963
5211.428571428572
7238.0952380952385
5461.077844311377
6656.934306569343
6246.575342465753
6561.151079136691
6857.142857142857
6961.832061068702
```





Experiments

```
No Object
No Object
No Object
1543.1472081218274
No Object
No Object
1717.5141242937852
2303.030303030303
2082.1917808219177
2268.6567164179105
2320.610687022901
3070.7070707070707
2980.392156862745
2320.610687022901
2375.0
2303.030303030303
2235.294117647059
2980.392156862745
2788.9908256880735
2471.5447154471544
2338.4615384615386
2285.714285714286
2187.05035971223
2000.0
1974.025974025974
2356.5891472868216
2140.845070422535
1974.025974025974
2026.6666666666667
1727.2727272727273
1608.4656084656085
1551.0204081632653
1375.5656108597284
No Object
No Object
No Object
No Object
No Object
No Object
No Object
No Object
1256.198347107438
No Object
1388.1278538812785
1679.5580110497237
1567.0103092783504
1788.235294117647
```

Results

- Using CVLIB library for object detection
  - Object detection in single frame : 0.20 (FPS)
  - Depth calculation : 0.85 (FPS)
- Using Tiny Yolo weights
  - Object detection in single frame : 6.74 (FPS)
  - Depth calculation : 8.42 (FPS)

Result’s are of CPU and experimental based.

Problem

- Depth is not accurate (dependency on disparity)
    - Bounding boxes along the object is not perfect – due to which disparity is not accurate

- Solution (Future Work)
    - 1st method
        - Average Disparity
    - 2nd method
        - Instead of taking point's from bounding box need to take other method.
        - Trying to use concept of template matching, reading SURF and ORB techniques to get accurate results

Experiments

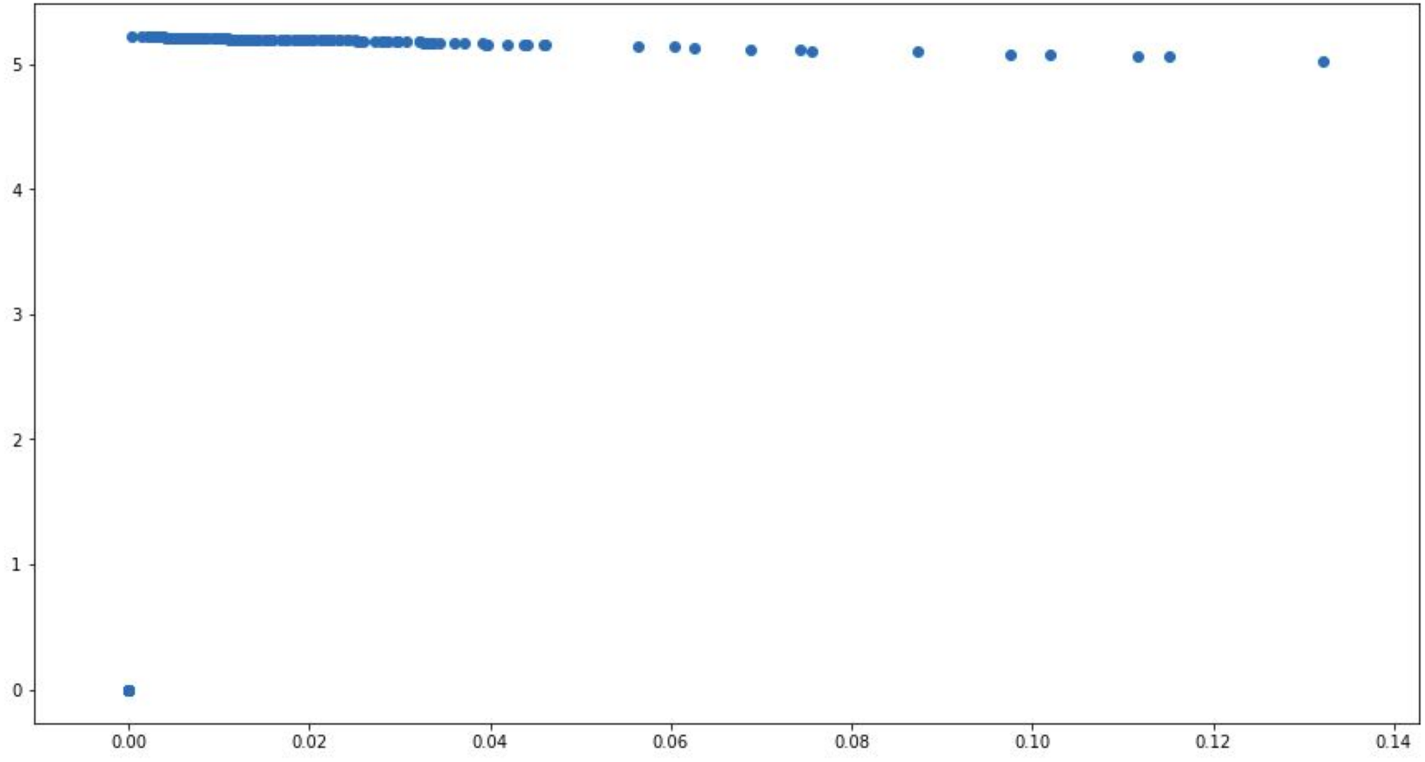
ORB Detector

- Disparity based on descriptor's shift

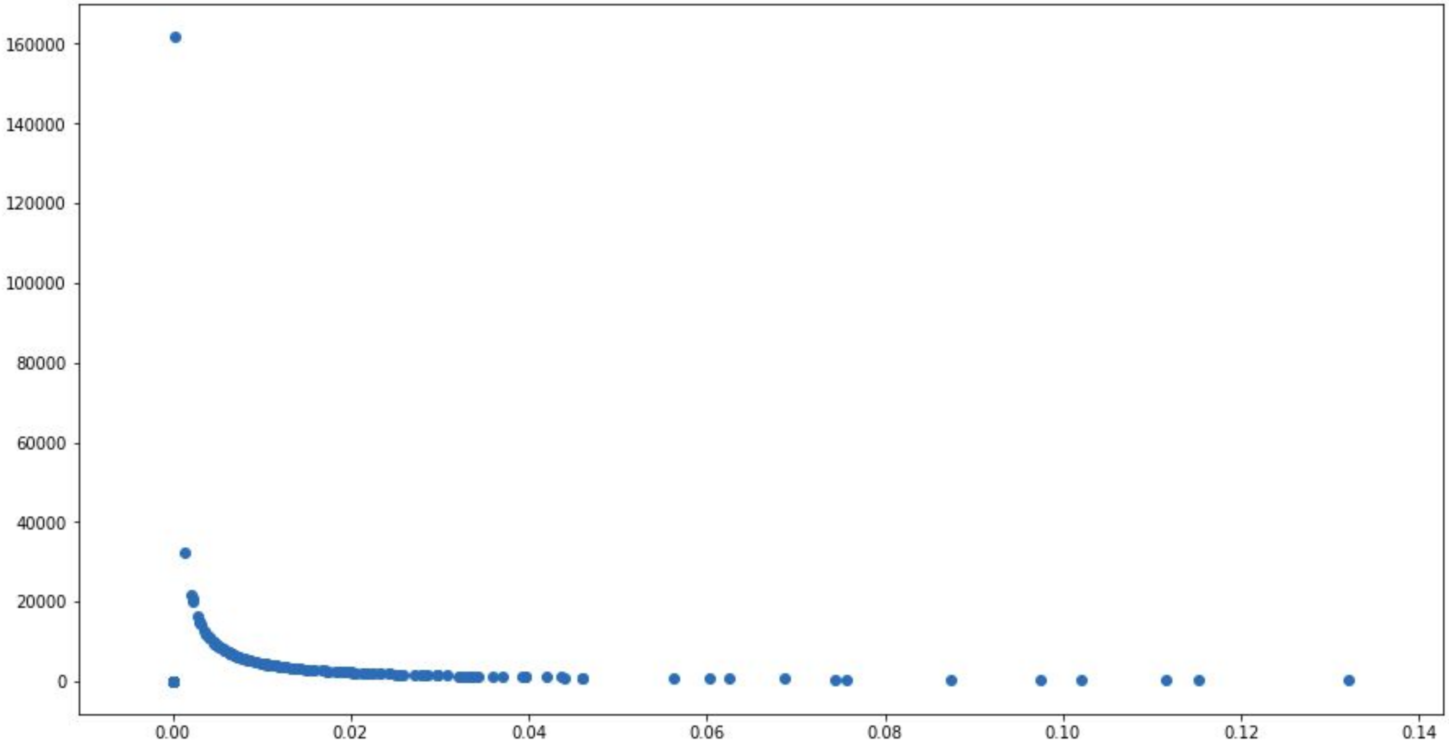
Average Disparity

- Average Disparity
  - Curve fitting
  - Using Formula

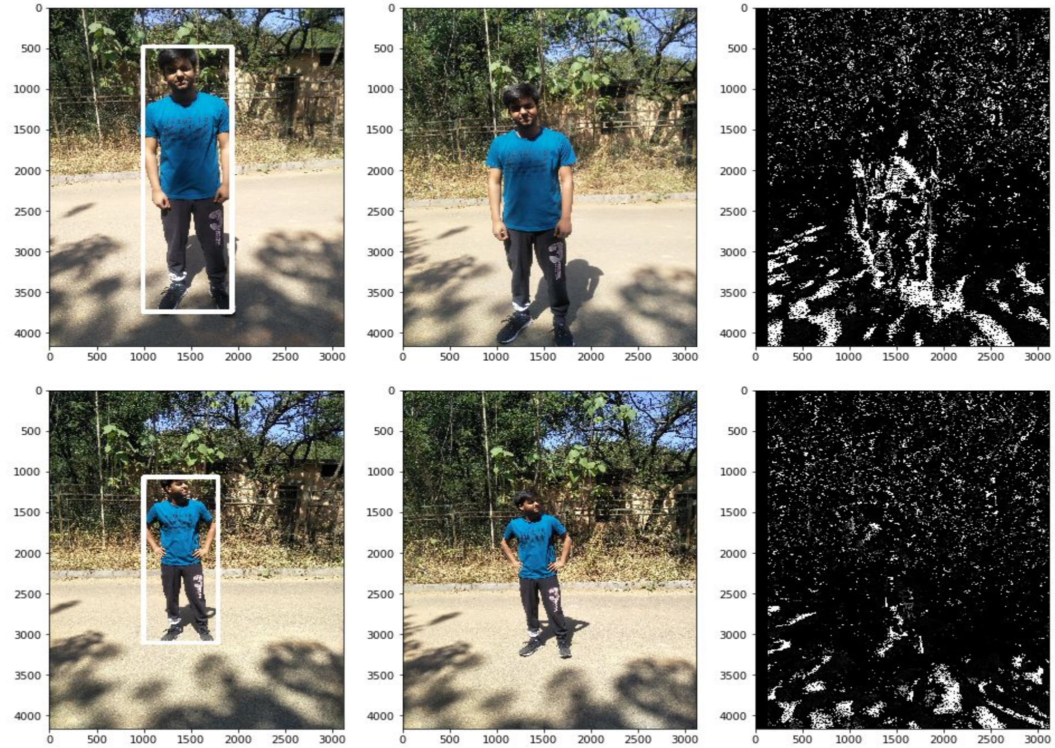
Experiments



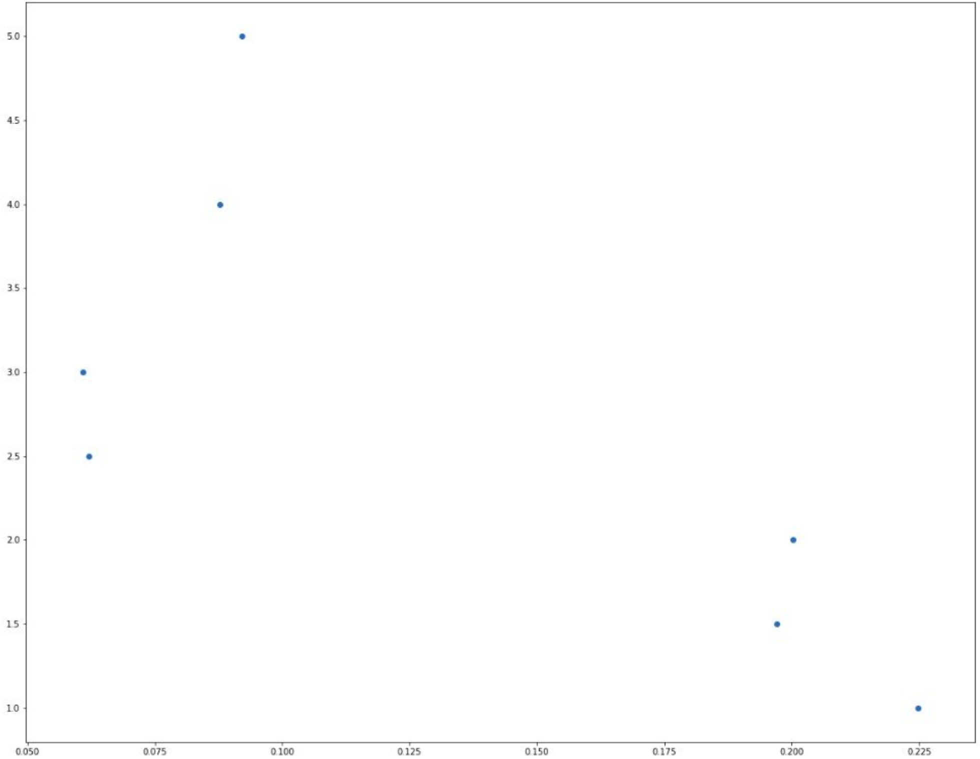
Experiments



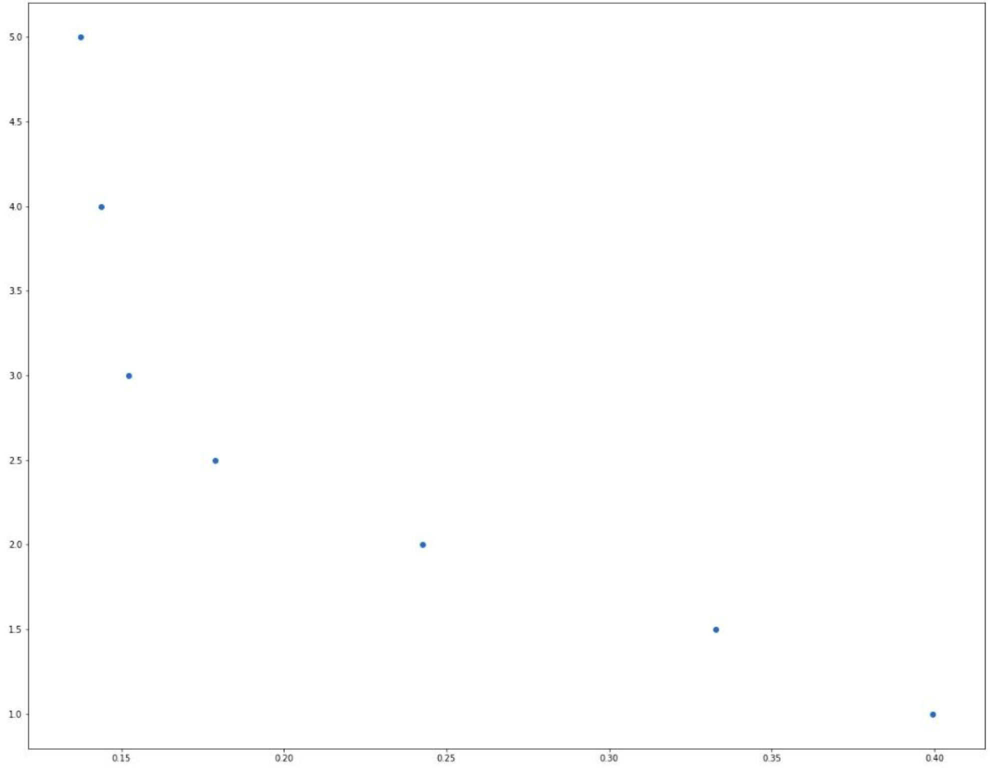
Experiments



Experiments



Experiments



Experiments