DEPTH COMPUTATION AND ITS ANALYSIS

Submitted as an assignment toward EC405 – Computer Vision

DEVVJIIT BHUYAN
ECB19050
B. TECH(ECE)
7th semester

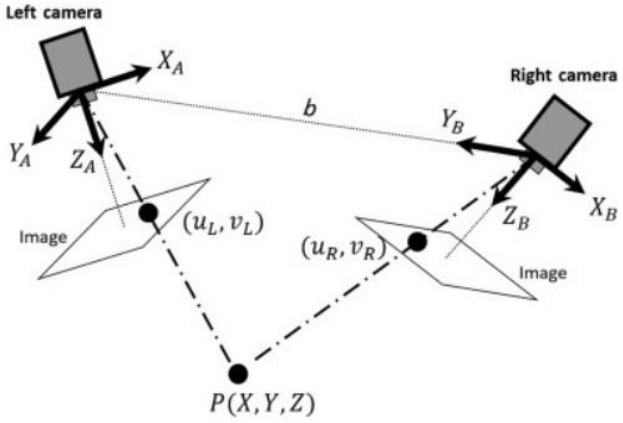
Introduction

This report contains the study and implementation of the Depth Computation and Verification in Stereo Vision. We have done an experimentation of the triangulation technique to find the depth map in a stereo pair of images, and found the results to be compliant, although with a few outliers.

We gather from this analysis that the show method can be a dependable technique for approximate depth computation.

This report is submitted toward EC405 – Computer Vision

Stereo Vision



Stereovision techniques use two cameras to see the same object. The two cameras are separated by a baseline, the distance for which is assumed to be known accurately. The two cameras simultaneously capture two images. The two images are analysed to note the differences between the images. Essentially, one needs to accurately identify the same pixel in both images, known as the problem of correspondence between the two cameras. Features like corners can be easily found in one image, and the same can be searched in the other image. Alternatively, the disparity between the images can be found to get the indicative regions in the other image, corresponding to the same regions in the first image, for which a small search can be used. The disparity helps to get the depth of the point which enables projecting it in a 3D world used for navigation.

Depthmap computation

Stereo vision can be used to perceive real world objects using a set of 2-D cameras. The two views from both cameras can be used to reconstruct a 3-D view of the actual object. It uses concepts of projection, parallax, and concepts such as triangulation to accurately locate an object in the real-world space. Stereo reconstruction is built upon this aspect of stereo vision.

A distance map can be used in many applications such as 3-D Reconstruction, finding the location of distant celestial bodies, etc.

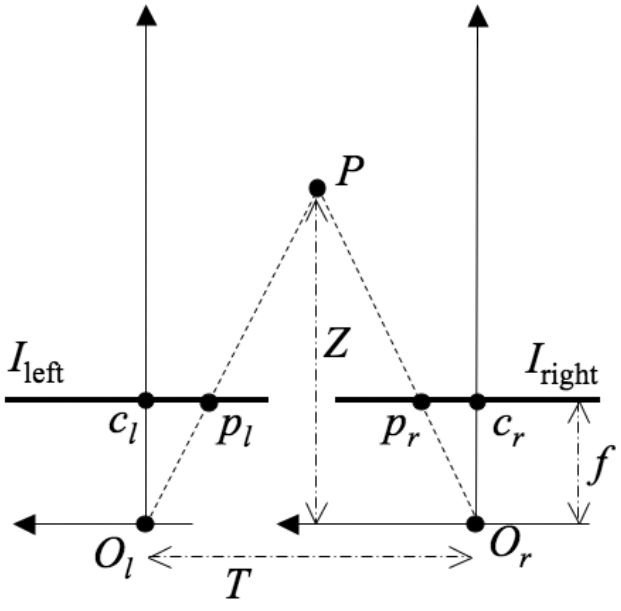
In simple terms it works on the concept of parallax – "A farther object will move lesser than a closer object when the viewing angle is changed". This parallax concept is extended to form the Triangulation technique, which is used to compute the Depthmap.

Triangulation

This technique is applied to pinpoint the depth of an object in real world space from the camera baseline. All we need is a disparity map (a map which shows how each keypoint varies between the two stereo images), knowledge about the camera parameters and stereo setup.

In the adjacent figure, considering T is the baseline, Z is the object depth, f is the focal length, and $x_l = c_l p_l$, and $x_r = c_r p_r$. Then we can derive:

$$Z = \frac{T * f}{x_l + (-x_r)}$$



Using this formula, for a given disparity map, we can create a depthmap for the scene.

Algorithm:

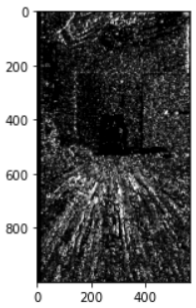
1. Draw correspondences between the two stereo images using methods such as SIFT (Scale-Invariant Feature Transform)
2. Use these correspondences to create a disparity map
3. Use the triangulation technique to find the depth corresponding to that pixel in the disparity map.
4. [Optional] Create Depthmap where intensity denotes the depth

Implementation in Python

In [58]:
```
import numpy as np
import cv2
import matplotlib.pyplot as plt

imgL = cv2.resize(cv2.imread("C:/Users/devbh/Desktop/IMG20221220170248.jpg", 0), (0, 0), fx = 0.25, fy = 0.25)
imgR = cv2.resize(cv2.imread("C:/Users/devbh/Desktop/IMG20221220170318.jpg", 0), (0, 0), fx = 0.25, fy = 0.25)

stereo = cv2.StereoBM_create(numDisparities=16, blockSize=5)
disparity = stereo.compute(imgL, imgR)
plt.imshow(disparity,'gray')
plt.show()
print(256, 5)
```

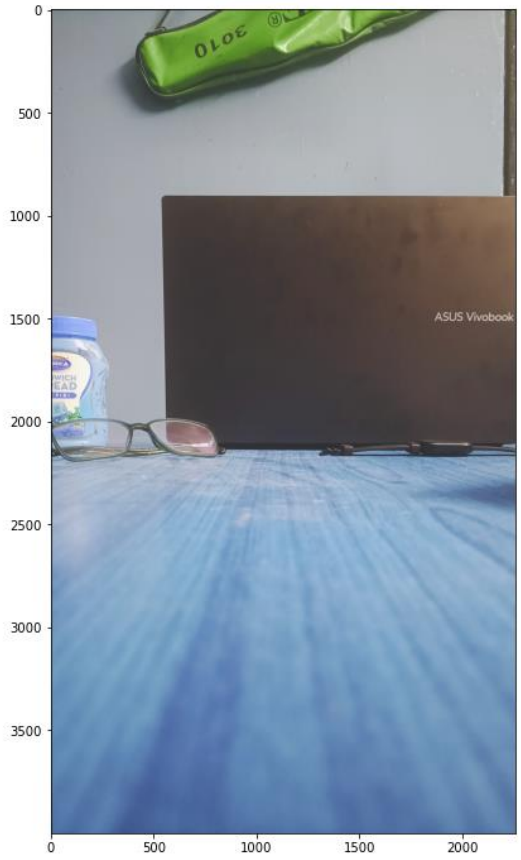


In [64]:
```
im = cv2.imread('C:/Users/devbh/Desktop/IMG20221220170248.jpg')
plt.figure(figsize = (60, 12))
plt.imshow(im)
```

Out[64]: <matplotlib.image.AxesImage at 0x1862d84afd0>

In [65]:
```
im = cv2.imread('C:/Users/devbh/Desktop/IMG20221220170318.jpg')
plt.figure(figsize = (60, 12))
plt.imshow(im)
```

Out[65]: <matplotlib.figure.AxesImage at 0x1862d89fac0>

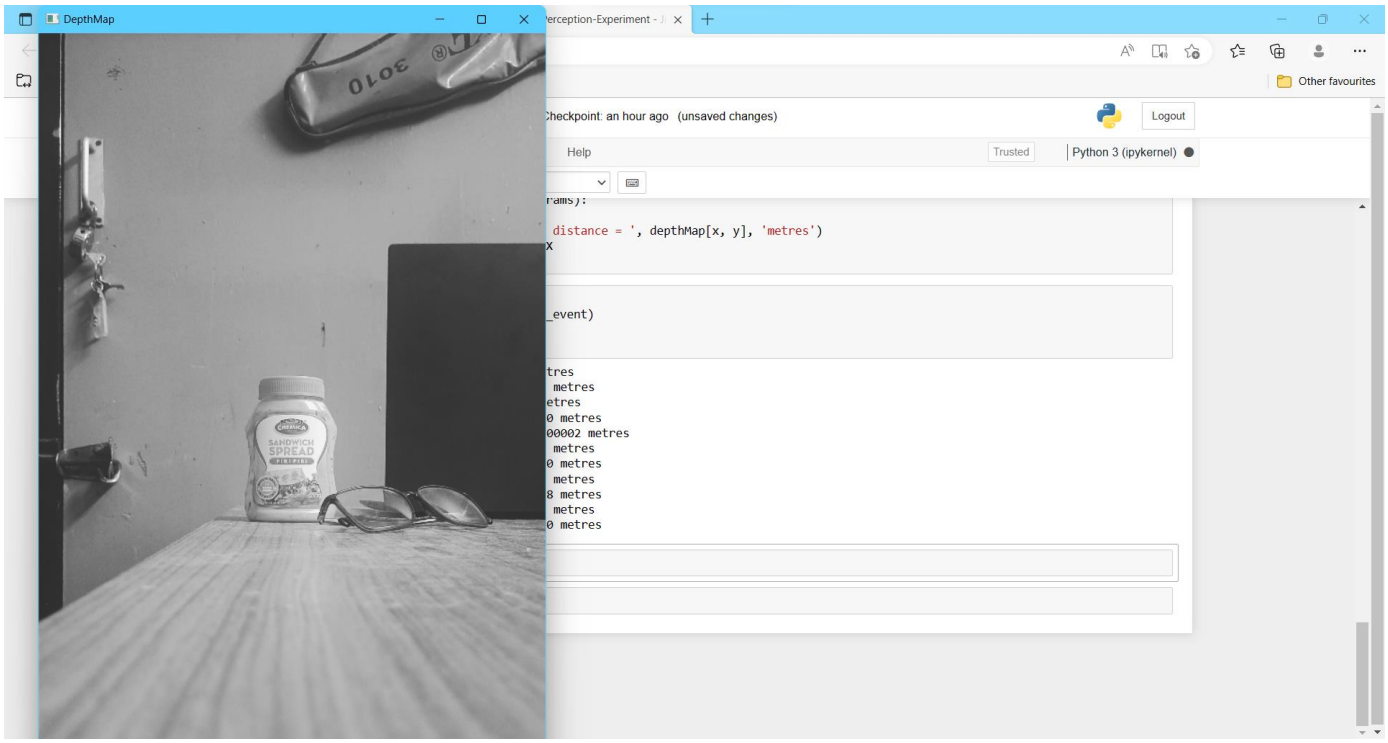


In [55]:
```
depthMap = np.zeros(disparity.shape)
Focal_length = 25
cameraBaseline = 0.1
h, w = disparity.shape
for i in range(h):
    for j in range(w):
        if disparity[i, j] > 0:
            depthMap[i, j] = (cameraBaseline * Focal_length) / disparity[i, j]
        else:
            depthMap[i, j] = 10000
```

In [56]:
```
def click_event(event, x, y, flags, params):
    if event == cv2.EVENT_LBUTTONDOWN:
        print('x = ', x, ' y = ', y, ' distance = ', depthMap[x, y], 'metres')
        font = cv2.FONT_HERSHEY_SIMPLEX
        cv2.imshow('DepthMap', imgL)
```

In [61]:
```
cv2.imshow('DepthMap', imgL)
cv2.setMouseCallback('DepthMap', click_event)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
x =  79  y =  479  distance =  1.09 metres
x =  364  y =  552  distance =  0.3518 metres
x =  283  y =  394  distance =  0.45 metres
x =  526  y =  521  distance =  10000.0 metres
x =  319  y =  327  distance =  0.579600002 metres
x =  498  y =  543  distance =  0.4912 metres
x =  215  y =  269  distance =  10000.0 metres
x =  355  y =  65  distance =  10000.0 metres
x =  315  y =  541  distance =  8.74568 metres
x =  280  y =  474  distance =  0.4877 metres
x =  513  y =  323  distance =  10000.0 metres
```



Reference Image for point selection

The points reported in the outputs are taken in the order of objects:

Object	Actual Distance measured	Distance shown in experiment
Door Lock	0.92 m	1.09 metres
Spectacles (nearer end)	0.33 m	0.3518 metres
Mayonnaise bottle's cap	0.55 m	0.45 metres
Lower portion on the laptop	0.50 m	10000.0 metres
Scratched point on the door	0.94 m	0.879600002 metres
Spectacles (farther end)	0.38 m	0.4912 metres
Point on door	0.94 m	10000.0 metres
Badminton Racquet (the number '0')	0.92 m perpendicular (diagonally > 1 m)	10000.0 metres
Spectacles (metal part)	0.32 m	8.74568 metres
Sandwich Spread text	0.54 m	0.4877 metres
Point on upper region of laptop	0.49 m	10000.0 metres

Images were captured using an Oppo mobile Phone with focal length 25 mm from the edge of the table, at an interval of 10 cm.

For certain points where the disparity isn't defined (or couldn't be computed), I have set the default depth value to be 10000 metres (to prevent division by zero). In most of the cases, the values are quite close to the actual values, with a few outliers. The results can further be improved if we use an actual stereo setup and a professional camera. However, from this analysis, we can gather that the triangulation technique alongwith OpenCV's disparity map computation is a reliable method for approximate depth computation.