Aerial Robotics Kharagpur Documentation Task Round

Sandeep Patel
21MF3IM17

***Abstract*— I have done the task 2.2 and 3.1 for the ARK Task Round 2022.I have worked on stereo Images and Aruco marker dictionaries and fuctions related to it for creating the code for the task.**

I. INTRODUCTION

This task is all about finding the distances of object form a stereo camera setup.It roams around finding disparity map and then dealing with the focal length and baseline to find the actual depth.

II. PROBLEM STATEMENT TASK 2.2

In this task we were given 2 images from 2 cameras one left and one right using which we have to find out the distance of a specific blue colored bike. I use the standard approach for this problem by initially calculating the disparity and then using it to find out the exact distance.

III. RELATED WORK

I learned more about the open cv functions and application like creating disparity map using opencv on stereo images and calculating 3d map from 2d images.

IV. INITIAL ATTEMPTS

My intial attempts were to use template matching to find the coordinates of the centre of bike in both the images and then use trignometry to find the distance.But soon I realised that it would work but it won't be generalised.

V. FINAL APPROACH

My final approch was to by calculating disparity map of both the 2 images.

A. *Disparity*

Disparity is the apparent pixel difference or motion in the 2 stereo images.A disparity map shows the intensity of motion or change in pixels of two images and by calculating taking the value at a certain point one can easily get the disparity value in pixels.It is inversely proportional to depth, and it is possible to define a mapping from an (x,y,d) triple to a three-dimensional position.

Then I used the cv2.imread function to read both the images along with the bike in gray.

```
imgL=cv.imread("left.png",0)

imgR=cv.imread("right.png",0)

template = cv.imread('bike.png',0)
```

Then I used template matching to find the location of bike center in one of the images.

```
matchTemplate(imgL,template,cv.TM_CCOEFF_NORMED)
```

After getting the coordintes I created a disparity map out of the images using the function:

```
StereoBM_create(numDisparities=48,blockSize=21)
```

numDisparities – Maximum disparity minus minimum disparity. This parameter must be divisible by 16.

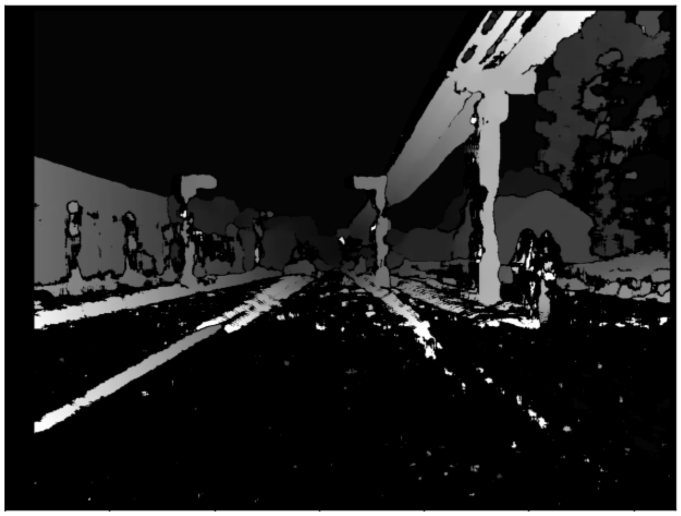
I adjusted the numDisparities and blockSize values accordingly to get values of non negative disparity value. After getting the disparity map I found the values of focal length and distance between the cameras from the given camera matrix. Using the formula

$$d = \frac{flength * baseline}{disparity} \tag{1}$$

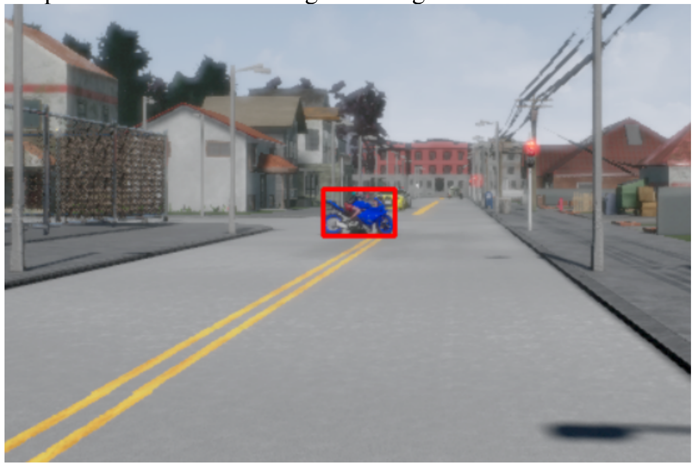
we can calculate the depth of the object if we know the disparity at that point and other parameters.

VI. Results and Observation

This is the disparity map created from the 2 given images



The darker the shade of gray shows the farther the image is ,as the value of disparity is inversely propotional to the distance from the object.This image shows the matched template of the bike in the given image.



I used the formula and found the distance to be near around 299.41. It has its unit according to the units of the focal length provided.

VII. Future Work

For the future work the same algorithm can be applied to a stereo camera that can be attached to the drone and using which we can calculate presence of any realtime obstacles in the drone's view by creating a simple disparity map of the video frames captured.Through which we can easily avoid any obstacle in the predefined path.

Conclusion

The problem was all about calculating disparity and then using it to calculate the distance.It would be very useful in judging the relative location of objects around the Vehicle.

VIII. Introduction

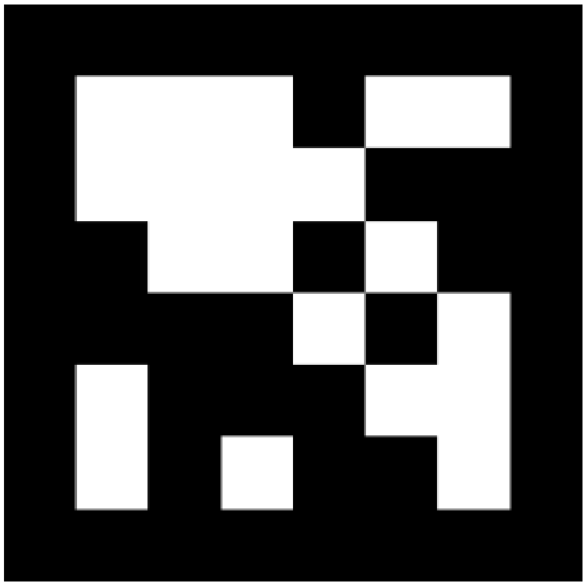
In this task we need to identify Aruco markers there location and then calculating the pose in realtime and along with creating an axis that shows the position of the marker.

IX. Problem Statement 3.1

This task requires use to know how to use aruco markers for estimation of pose and relative location of camera using the same.Using webcam and opencv-python one can create functions to estimate pose and location of aruco markers.

A. Aruco Markers

ArUco markers are binary square fiducial markers that can be used for camera pose estimation. Their main benefit is that their detection is robust, fast and simple. The aruco module includes the detection of these types of markers and the tools to employ them for pose estimation and camera calibration.They all have a specific number attached to each of the marker resolution type.They look something like this



X. Realted work

In this part I explored a lot about the aruco library in open-cv and using its function for Detecting Aruco Markers.

XI. Inital Attempts

In intial attempts I just was able to find the markers but was unable to estimate the pose after a lot of searching I came to find out about the camera calibration and its use in estimating the pose and then I applied to reach the final result.

XII. Final Approach

A. Camera Calibration

This is the first step in which we need to calibrate our camera's and get the camera and distortion matrix which when we feed in the pose estimation fuction we can find the pose of Marker without any distortion.

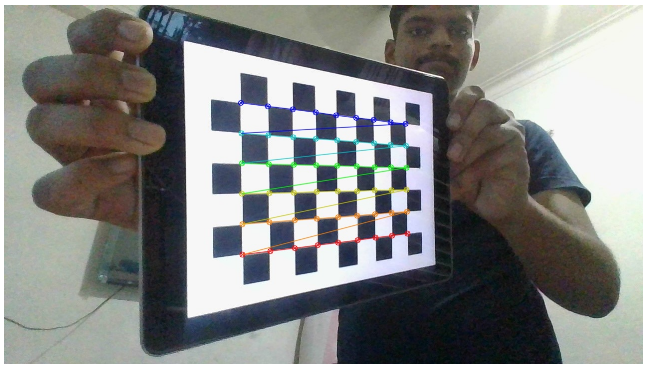
Camera Matrix:- It is a 3X4 matrix which describes the mapping of a pinhole camera from 3D points in the world to 2D points in an image.

Distortion Matrix:-The camera matrix does not account for lens distortion because an ideal pinhole camera does not have a lens. To accurately represent a real camera, the camera model includes the radial and tangential lens distortion.So we use a distortion matrix to keep a note of these kind of distortion

```
images = glob.glob("../images/*.jpg")
i = 0
for fname in images:
    img = cv2.imread(fname)
    gray = cv2.cvtColor(img, cv2.COLOR
    _BGR2GRAY)
    size = gray.shape[::-1]

    ret, corners = cv2.findChessboardC
    orners(gray,
    (9, 6), None)
```

Now we use the glob function from glob library to access all the files with a similar extension(.jpg)in a certain folder.In that folder one need to take photos of chessboards in different orientation so that one the next steps the we can find and mark those accordingly to create a camera matrix. Then we use the cv2.findChessboard function that finds chessboards in the given image and note there Corners and whether it finds it or not.I then stored it in a list and then we feed them to the cv2.calibrate function that returns the camera and distortion matrix.The picture shows me holding the chessboard picture in my tablet



. After getting the camera Matrix and distortion parameters we can start the finding the Aruco marker pose so we start by describing a key for our Marker.

```
getattr(cv2.aruco,f'DICT_{markerSize}X
{markerSize}_{totalMarkers}')
```

this store a dictionary of markers that are stored previously in the opencv library and then are used to match from the images given to find the markers. Then we directly use the find aruco marker function to detect and get the corners of the aruco marker in teh given Image.

```
cv2.aruco.detectMarkers(gray,aruco_dict,
parameters=parameters)
```

Now that we have the corners of each of the aruco markers we simply use drawDetectedMarkers which simply draws lines and border by the means of the corner coordinates we provided it earlier.
Now we declare our camera Matrix and Distortion Coefficients in our code and then use the

```
cv2.aruco.estimatePoseSingleMarkers(corners[i],0.0
```

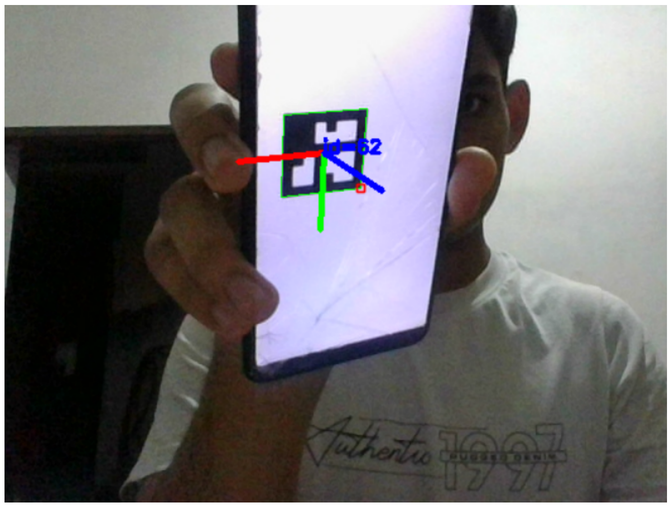
function to estimate the pose from the given frame of the video.And then

```
 cv2.aruco.drawAxis(frame,mtx,dist,rvec,tvec,0.02)
```

to draw Axis around the marker so that the pose estimation is visible.Then the cv2.imshow is used to show all these things directly in the webcam stream.

XIII. Results and Observations

Using this we can create a rough estimate of the surrounding with just a simple marker which can be printed or projected very easily.One can quite easily determine the relative pose from the marker and then if we know the absolute pose of the marker the absolute orientation of the object can also be determined quite easily. Here you can see me holding a aruco marker which is being detected by my code



XIV. Future Work

For future Work this can be helpful for many ways such as estimating orientation and calibrating a drones camera for knowing its location in the real world. And there are even many more possibilities realted to the same.

XV. Conclusion

In this part of the task it was all about knowing how to use the marker for various purposes and also gave a complete brushup for camera calibration and terms related to it like Distortion Matrix which is an essential part.It made me explore more about the open-cv library and its relatability to the real world.