Task 2.2: Stereo image based collision avoidance

Aryan A Kumar

***Abstract*— While looking out of the side window of a moving car, the distant scenery seems to move slowly while the lamp posts flash by at a high speed. This effect is called parallax, and it can be exploited to extract geometrical information from a scene. From multiple captures of the same scene from different viewpoints, it is possible to estimate the distance of the objects, i.e. the depth of the scene.**
This task requires the generation of a depth map using stereo images(images from two cameras placed side by side). Then, using the depth map the distance to an obstacle is obtained, which will be an essential parameter for collision avoidance.

I. INTRODUCTION

In order to find distances to obstacles we need depth information. For this we use two cameras in specific orientation relative to each other to generate slightly different images called **stereo** images. For this task we need to generate a **depth map** using stereo images, find the obstacle and calculate the **distance**. For depth map generation, first the disparity value was found using template matching and then depth was calculated using similarity of triangles.

II. PROBLEM STATEMENT

In this task we are provided with a pair of stereo images and projection matrices for the cameras. We need to:

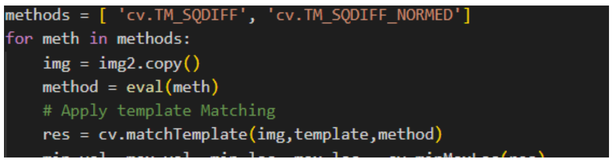
1) **Generate** a depth map using the stereo images and projection matrices.
2) **Find** the obstacle - a blue bike.
3) **Calculate** the distance to the obstacle (to be used for collision avoidance).

III. RELATED WORK

The method involving template matching was chosen because of its **efficiency** and **ease** of implementation.

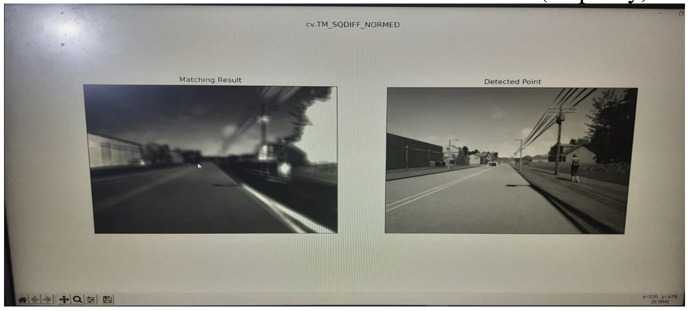
IV. APPROACH

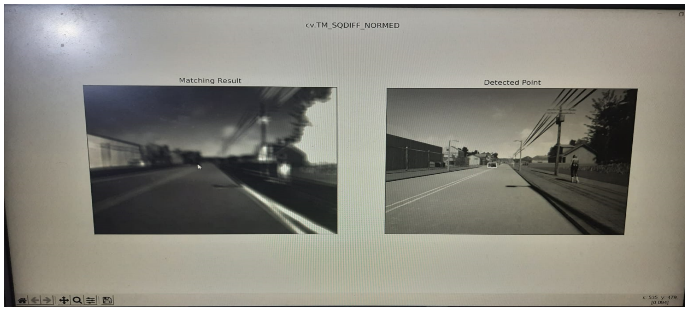
For calculating disparity, **template matching** in OpenCV was done using *cv.matchTemplate* function. The *cv.TM_SQDIFF* and the normalised *cv.TM_SQDIFF_NORMED* methods to estimate the extent of matching. A lower value returned by *cv.TM_SQDIFF* would imply a better matching.



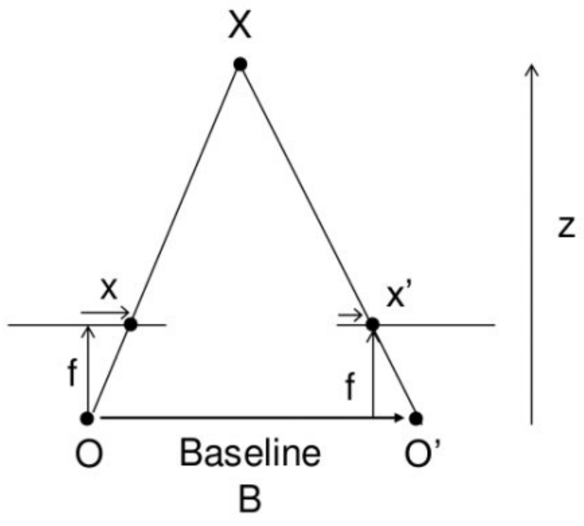
```
methods = [ 'cv.TM_SQDIFF', 'cv.TM_SQDIFF_NORMED']
for meth in methods:
    img = img2.copy()
    method = eval(meth)
    # Apply template Matching
    res = cv.matchTemplate(img,template,method)
```

The basic idea is to calculate the difference between x coordinates (disparity):





Disparity = 545-535 = 10 pixels. Next we calculate the distance:



$$disparity = x - x' = \frac{Bf}{Z}$$

Here x and x' are the distance between points corresponding to the scene point in 3D and the camera center in the

image plane . B is the **distance** between two cameras and f is the focal length of camera. **Z** is the **depth** required.
So in short, the above equation says that the depth of a point in a scene is inversely proportional to the difference in distance of corresponding image points and their camera centers. So with this information, we can derive the depth of all pixels in an image.

Next, for calculating intrinsic parameters like focal length (f) and distance between cameras (B), we need to calculate the **intrinsic matrix** (**K**):

$\underset{\text{(intrinsics)}}{\mathbf{K}}$ (converts from 3D rays in camera coordinate system to pixel coordinates)

$$\text{in general, } \mathbf{K} = \begin{bmatrix} -f & s & c_x \\ 0 & -\alpha f & c_y \\ 0 & 0 & 1 \end{bmatrix}$$ f is the focal length of the camera

α : **aspect ratio** (1 unless pixels are not square)

s : **skew** (0 unless pixels are shaped like rhombi/parallelograms)

(c_x, c_y) : **principal point** ((0,0) unless optical axis doesn't intersect projection plane at origin)

To do that, we use the following matrix equation:

Projection matrix

$$\mathbf{\Pi} = \underset{\text{intrinsics}}{\mathbf{K}} \underbrace{\underset{\text{projection}}{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}} \underset{\text{rotation}}{\begin{bmatrix} \mathbf{R} & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ 0\;0\;0 & 1 \end{bmatrix}} \underset{\text{translation}}{\begin{bmatrix} \mathbf{I}_{3\times3} & -\mathbf{c} \\ 0\;0\;0 & 1 \end{bmatrix}}}$$

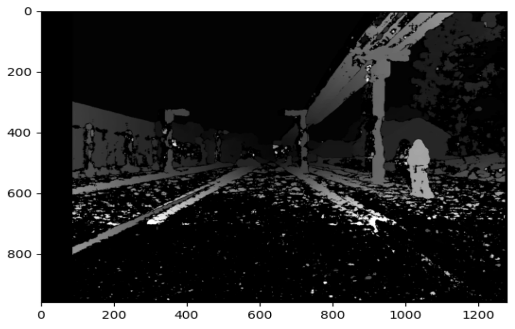
Here the symbol pi stands for the projection matrix. For simplicity we assume R = 3x3 identity matrix.
The given projection matrices:

```
p_left = [[640.0,   0.0, 640.0, 2176.0],
         [0.0, 480.0, 480.0,  552.0],
         [0.0,   0.0,   1.0,    1.4]]
p_right = [[640.0,   0.0, 640.0, 2176.0],
          [0.0, 480.0, 480.0,  792.0],
          [0.0,   0.0,   1.0,    1.4]]
```

The focal length f= -640 pix. The distance between cameras = (792-552)/480 =0.5 meters. Using these and the value of disparity in the above said equation, we get depth (Z) = f*B/disparity = 640*0.5/10 = **32 meters**.

V. Results and Observation

The **distance** to the obstacle was found to be about **32** metres, which seems to be of the true order. For representation, the **disparity** map can be created and displayed using *stereoBM.create*:



VI. Future Work

After calculating the **distance** to the obstacle, the next step would be to determine the **direction** of the obstacle from the ego vehicle relative to our camera with the help of projection matrices.
Enabling obstacle avoidance: The basic concept of obstacle avoidance is determining if the distance of any object from the sensor is closer than the minimum threshold distance. In our case, the sensor is a stereo camera.

After that, we would need to exclude all paths that are blocked by the obstacle from our list of possible paths. However, if the obstacle is moving, we would need real time data to navigate successfully.(without crashing into an obstacle).

Conclusion

While navigating autonomously(or otherwise) in a 3D environment, we need data about the surrounding. This can be done for example, using LiDAR.
"anyone relying on lidar is doomed." - Elon Musk.
LiDAR and similar sensing methods, although accurate, are computationally very expensive and require expensive sensors in general. As a cheaper and more practical substitute, we use cameras for sensing the environment and extract all necessary data from the input images itself. Obtaining depth information of objects is one such case.

This task required utilizing stereo images to obtain a depth map of the 3D world. Similar to the way our eyes work, the two slightly different images can give us information of depth which is crucial for collision avoidance.
The depth was calculated by first generating a disparity map, followed by calculation of depth using focal length, **baseline**(both obtained from the projection matrices), and **similar triangles**.

References

[1] https://docs.opencv.org/4.x/dd/d53/tutorial_py_depthmap.html
[2] https://learnopencv.com/depth-perception-using-stereo-camera-python-c/
[3] https://engineering.purdue.edu/kak/computervision/ECE661Folder/Lecture17.pdf
[4] https://courses.cs.washington.edu/courses/cse576/17sp/notes/CamerasStereo17.pdf