CSL7360:Computer Vision
Assignment 1

Anushkaa Ambuj (B21ES006)

Objective:

This assignment aims to implement the Harris Corner Detection algorithm from scratch. Implement the stereo 3D reconstruction algorithm to find the disparity map, depth map (depth map at each pixel), and 3D point cloud representation of the underlying scene and find the corresponding pixels on the epipolar line in the first image.

Input Images:

https://drive.google.com/drive/folders/1la4hwF_n4g7T25d2gyCF1ob3HJOir3Th

Task 1: Harris Corner Detection Algorithm

❖ **Key formulas used in the Harris Corner detection algorithm:**

1. **Gradient Calculation**
   The algorithm starts by calculating the gradients of the image in the x and y directions. This is typically done using derivative filters such as Sobel filters. The gradients are denoted as I_x and I_y.

$$I_x = \frac{\partial I}{\partial x}, I_y = \frac{\partial I}{\partial y}$$

2. **Structure Tensor Elements**
   Using the gradients, the elements of the structure tensor M are computed. These elements are used to characterize the local image structure.

$$I_{xx} = I_x^2, I_{yy} = I_y^2, I_{xy} = I_x I_y$$

3. **Summation over window**
   The next step involves summing up these structure tensor elements over a local neighbourhood or window. This is typically done using a Gaussian filter to give more weight to central pixels.

$$S_{xx} = \sum_{x,y} w(x,y) \cdot I_{xx}, S_{yy} = \sum_{x,y} w(x,y) \cdot I_{yy}, S_{xy} = \sum_{x,y} w(x,y) \cdot I_{xy}$$

4. **Harris Corner Response (R)**

$$R = det(M) - k \cdot trace(M)^2$$

   Where, parameter k is a constant typically chosen in the range of 0.04 to 0.06

$$det(M) = S_{xx} \cdot S_{yy} - S_{xy}^2, trace(M) = S_{xx} + S_{yy}$$

5. **Corner Thresholding**

$$\text{Region} = \begin{cases} \text{Corner}, & \text{if } R > \text{Threshold} \\ \text{Not Corner}, & \text{otherwise} \end{cases}$$

   Note:
   1. When |R| is small, the region is flat; that is when λ1 and λ2 are small.
   2. When R<0, which happens when λ1>>λ2 or vice versa, the region is an edge.
   3. When R is large, which happens when λ1 and λ2 are large and λ1~λ2, the region is a corner.

❖ **Comparing Harris Corner Detection from scratch with OpenCV library.**

Image 1

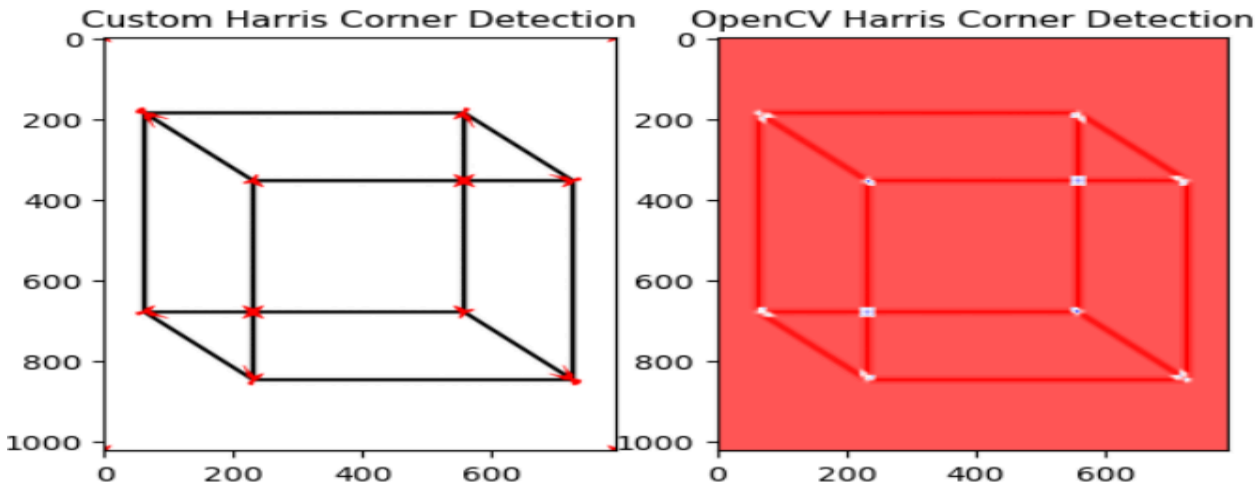


Image 2

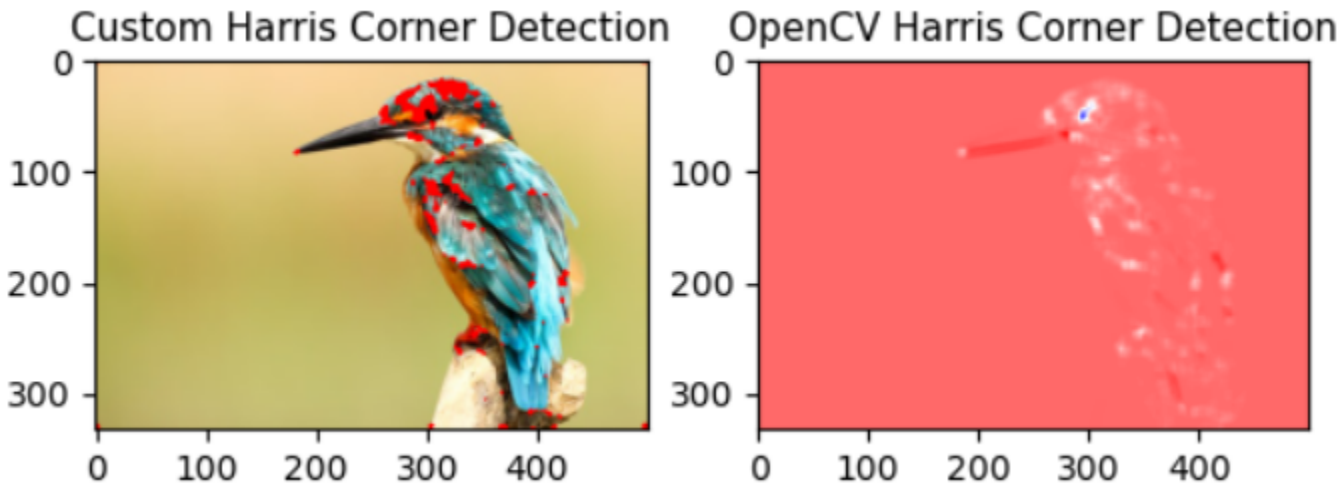


Image 5

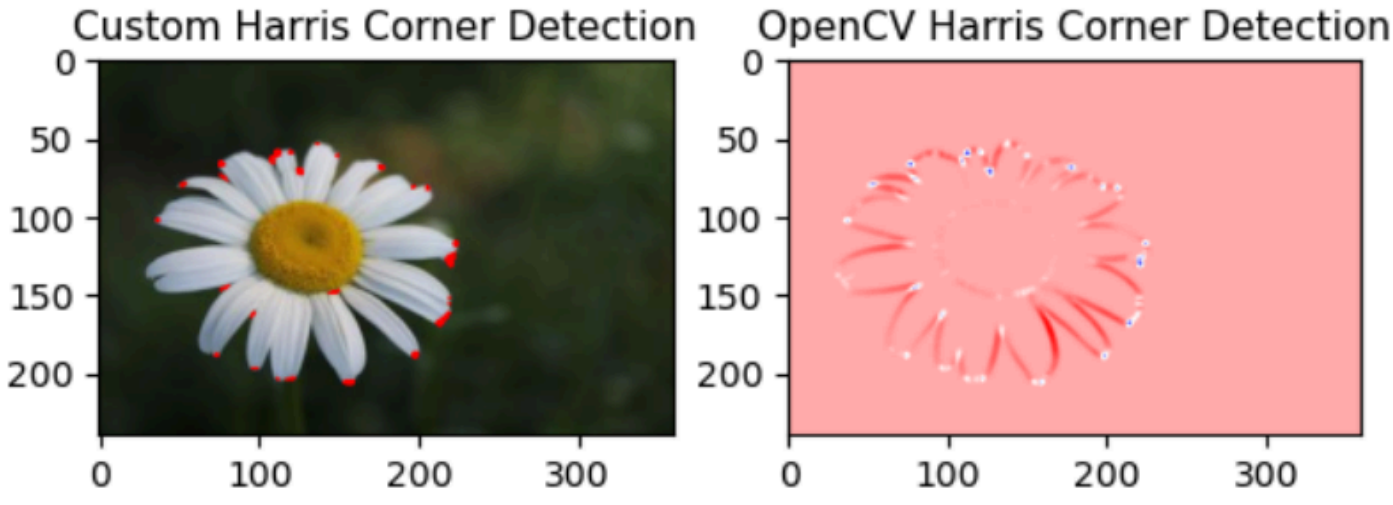


Image 6

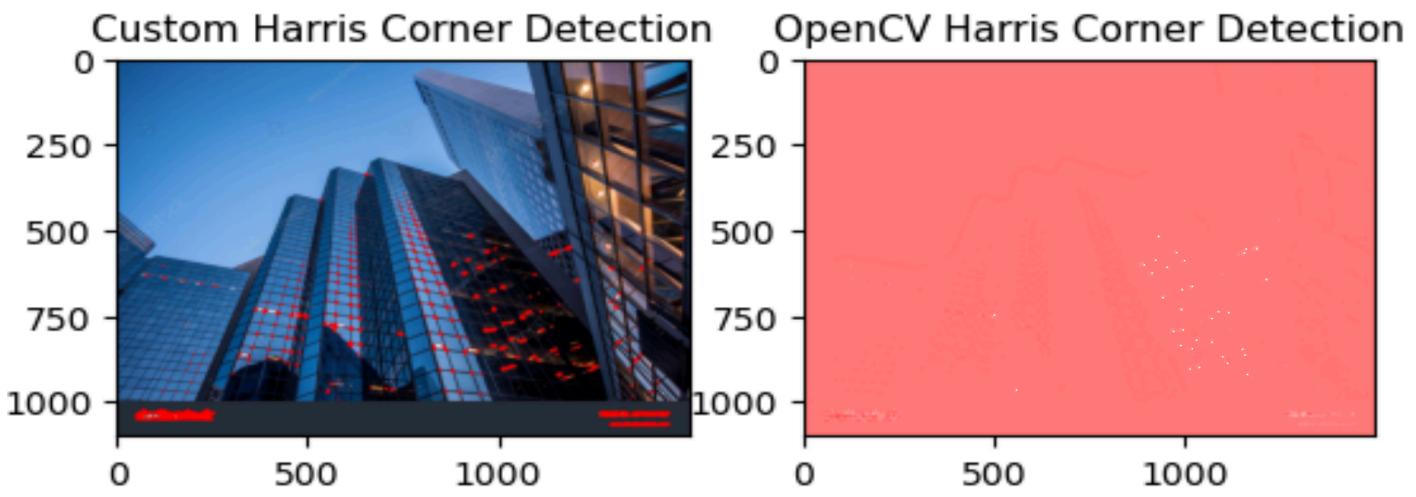


Image 7

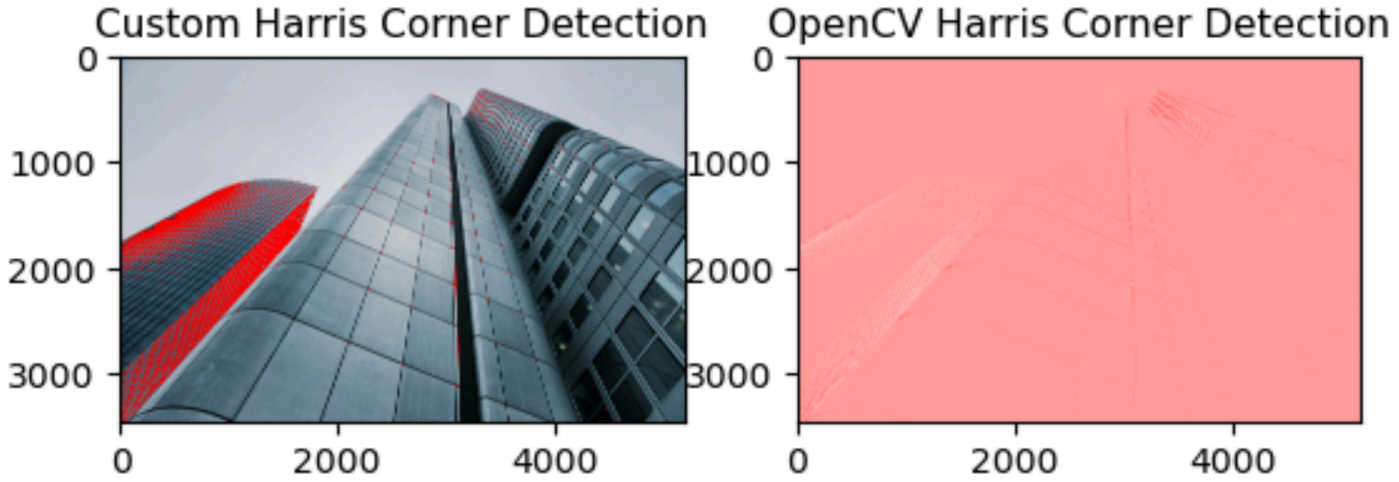


Image 8

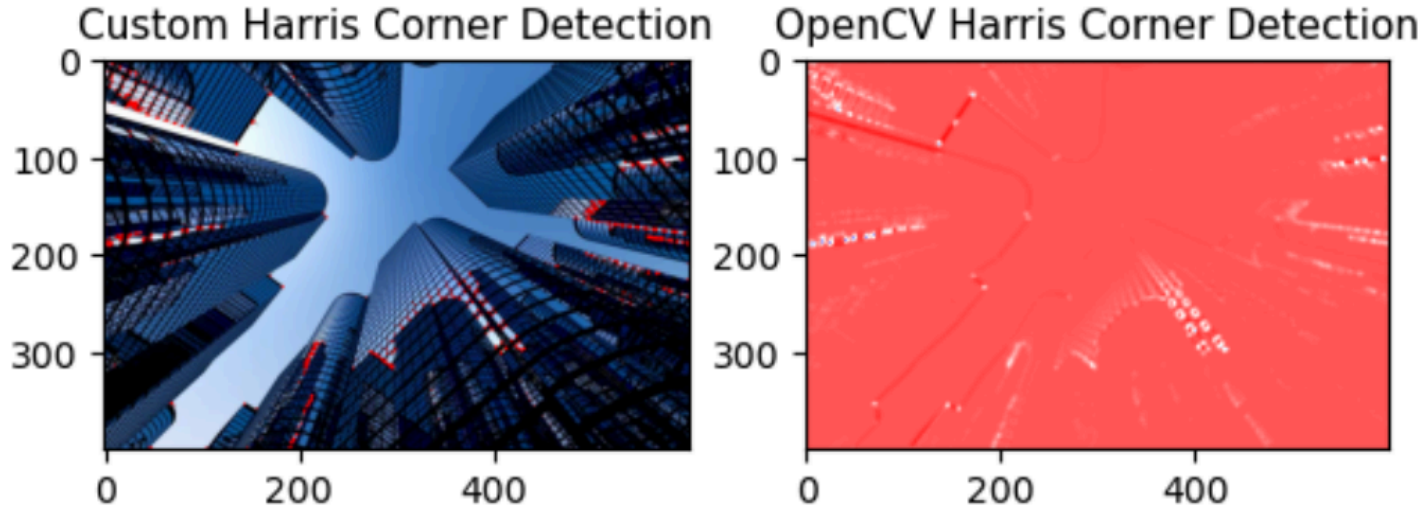


Image 11

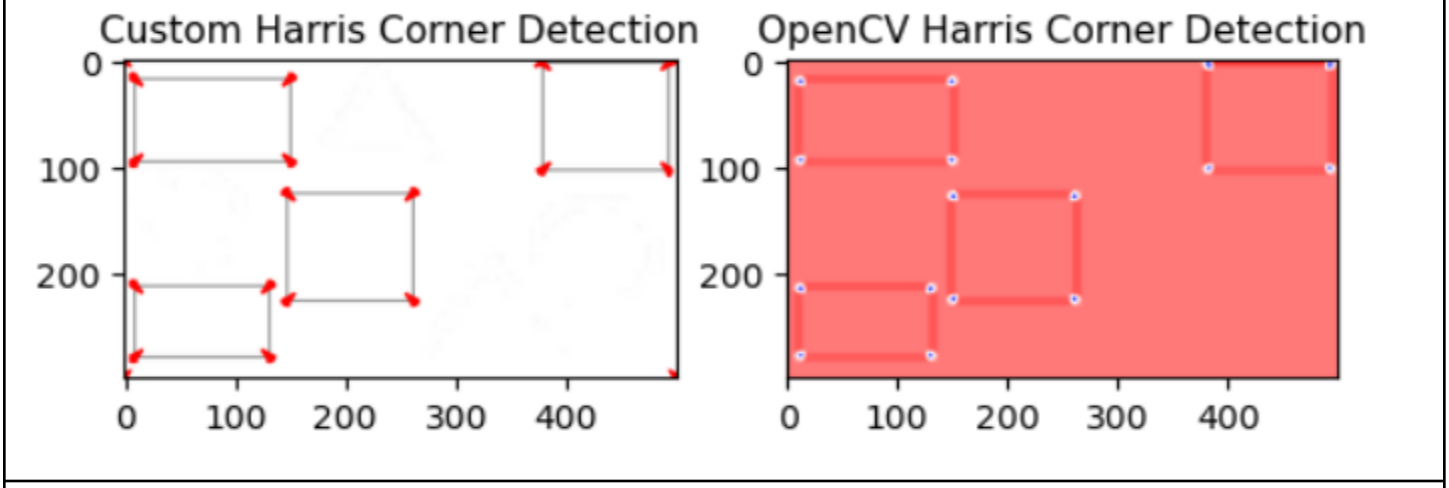


Image 10

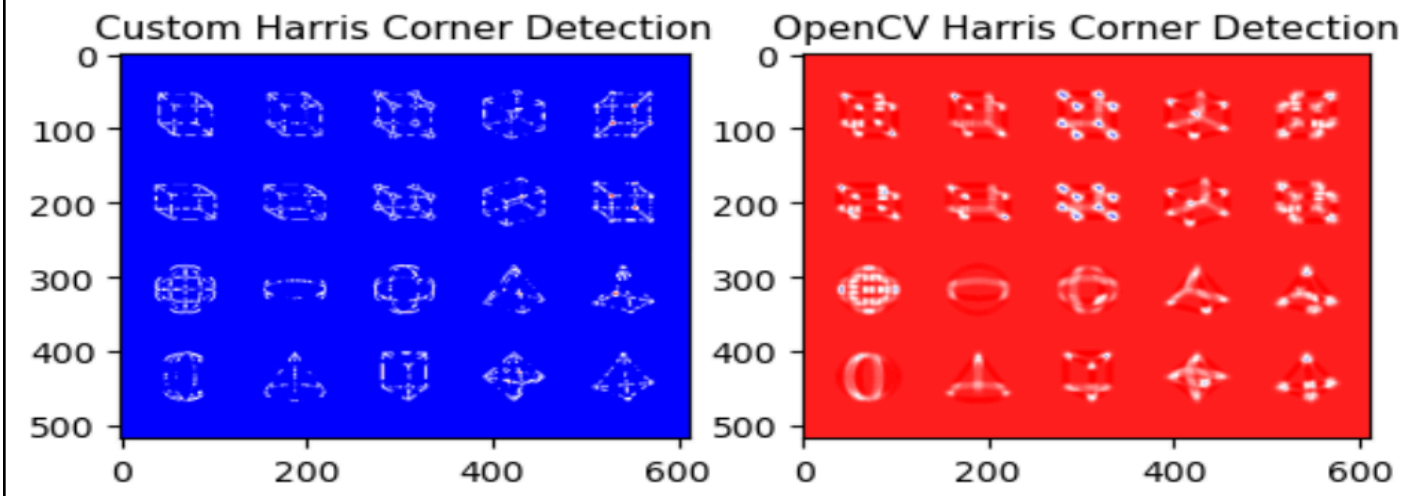
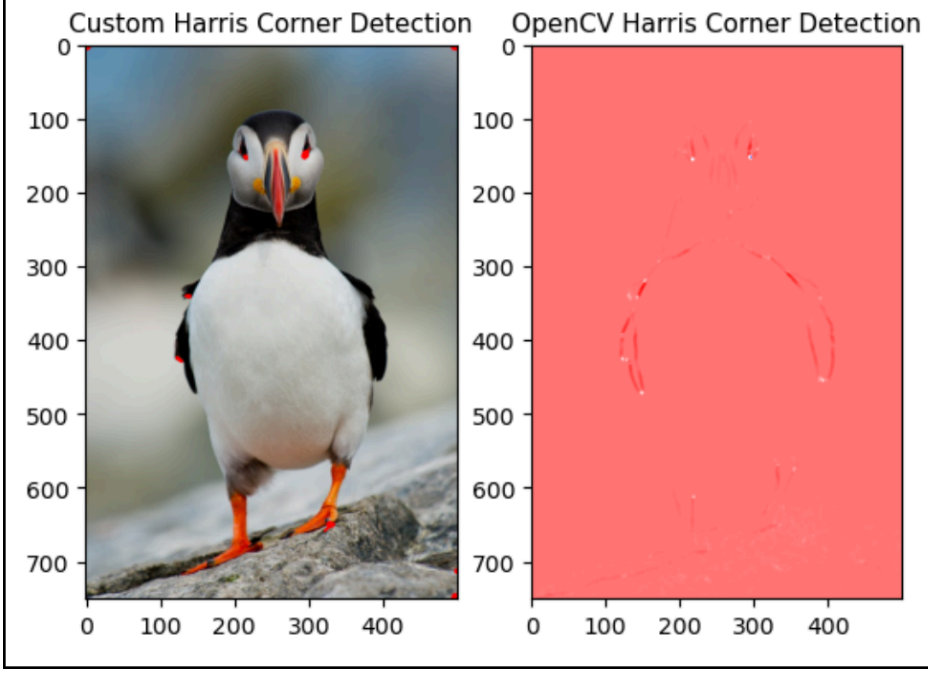


Image 9



Observations:

The implemented Harris Corner detection algorithm exhibited robust performance in detecting corners across various images. Adjusting parameters significantly impacted the corner detection results. Comparing the results with OpenCV's corner detection library showcased comparable performance, validating the correctness of the implementation.

- Smaller window sizes and lower thresholds resulted in more corners being detected, including finer details in the images.
- Larger window sizes and higher thresholds led to fewer corners detected, emphasizing only prominent features.

Task 2: 3D Reconstruction Algorithm

❖ Key formulas used:

1. **Disparity Calculation:**
   The disparity map D represents the pixel-wise differences in the corresponding locations between a pair of stereo images. It can be computed using algorithms like Stereo Block Matching (SBM).

$$D(x, y) = SBM(I_1(x, y), I_2(x, y))$$

   Here, I_1 and I_2 are the left and right stereo images, and (x,y) are the pixel coordinates.

2. **Depth Calculation:**
   The depth map Z provides the distance of each pixel from the camera plane. It can be calculated using the disparity map and camera parameters such as baseline distance between stereo cameras (B) and focal length of the camera (f).

$$Z(x, y) = \frac{B.f}{D(x,y)}$$

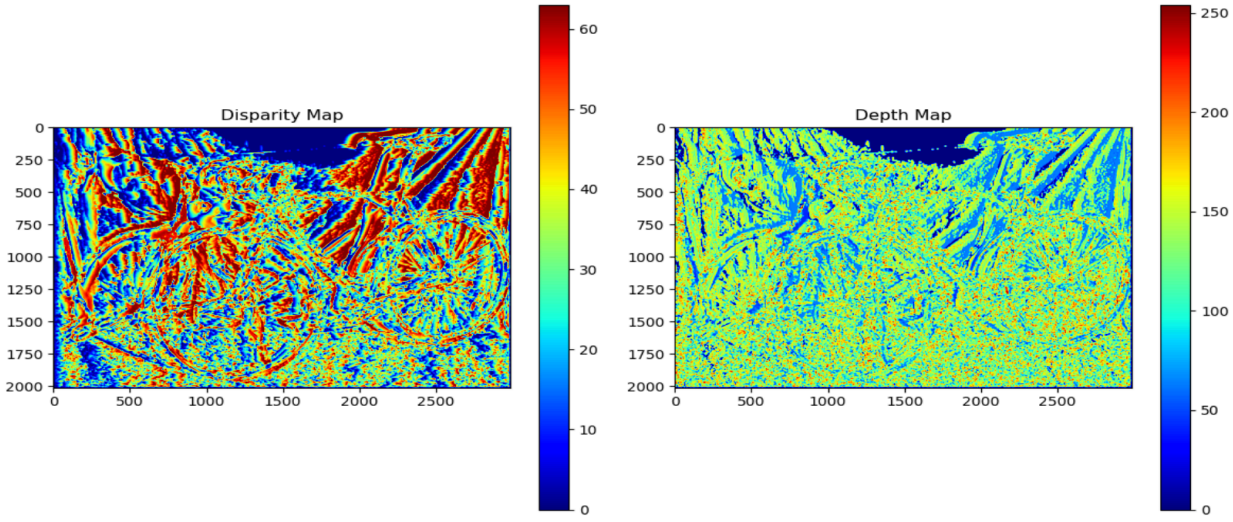
   Here, f = K[0,0], where, K = intrinsic matrix of the camera

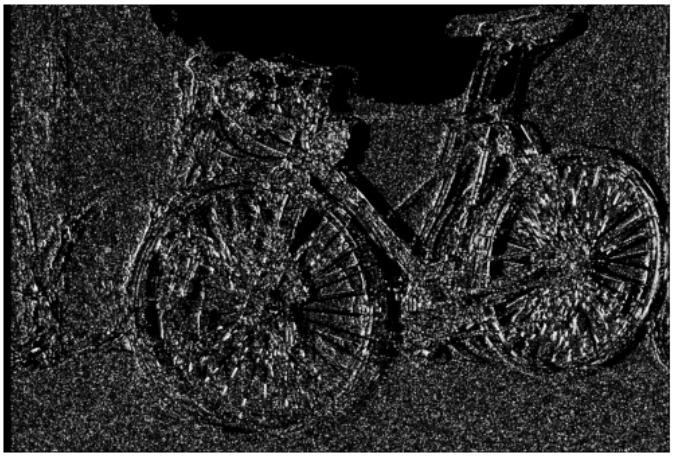
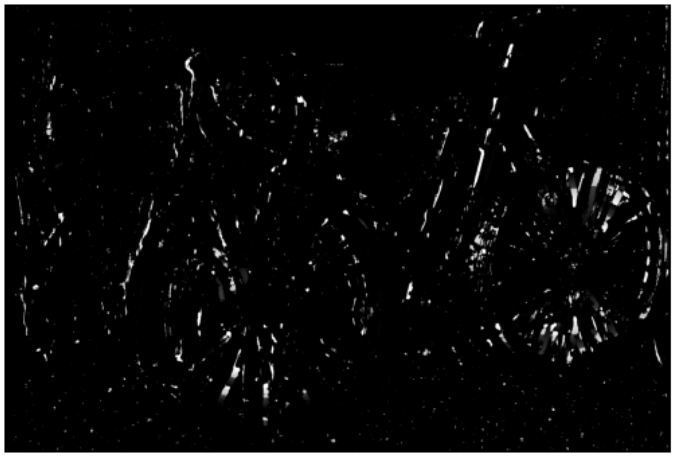
3. **3D Point Cloud Generation:**
   Once the depth map is obtained, the 3D coordinates of each pixel can be calculated using the camera's intrinsic matrix (K) and the depth value (Z).

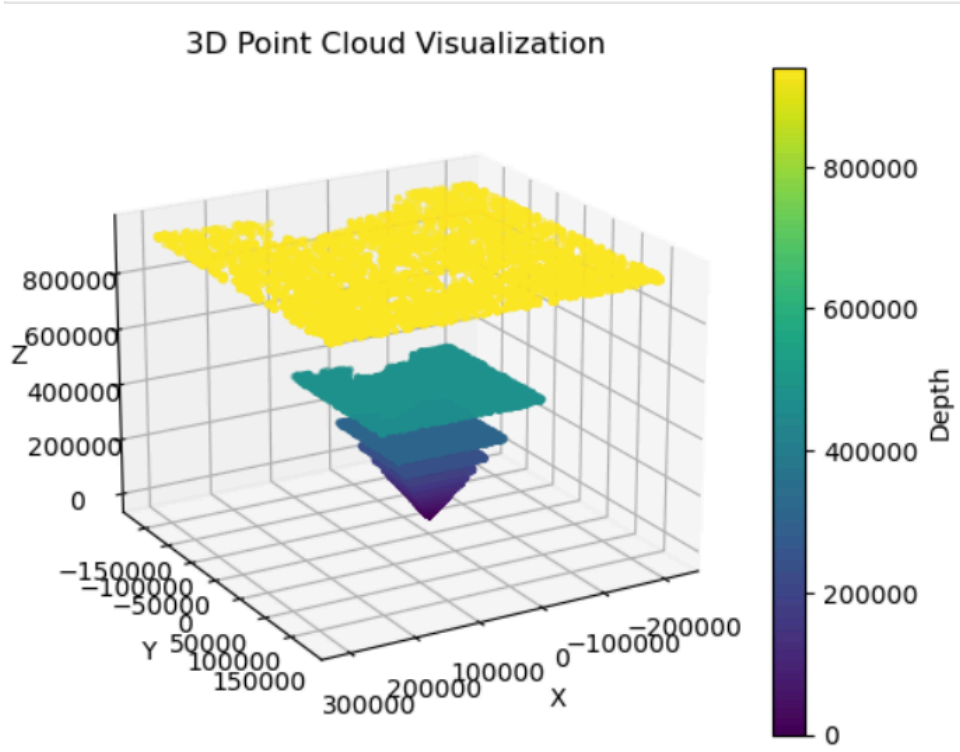
$$X(x, y) = \frac{(x - c_x) \cdot Z(x,y)}{f_x}, \quad Y(x, y) = \frac{(y - c_y) \cdot Z(x,y)}{f_y}$$

$$Z(x, y) = Z(x, y)$$

   Where, (c_x, c_y) = (K[0,2] , K[1,2]) are the principal point coordinates, and (f_x, f_y) = (K[0,0] , K[1,1])are the focal lengths along the x and y axes, respectively.



Disparity Map	Depth Map
	



Observations:

- Depth values inversely correlated with disparity values
- The depth map provided a clear visualization of scene depth, with brighter regions representing closer objects and darker regions indicating farther objects.

Task 3: Epipolar Lines

❖ Key formulas used:

1. **Simple Stereo**

   WKT, $F \,.\, point1 \;=\; 0$ and $F^{T} \,.\, point2 \;=\; 0$

   Where, F = Fundamental Matrix and point1 & point2 are corresponding points in the left and right images.

2. **Epipolar lines**

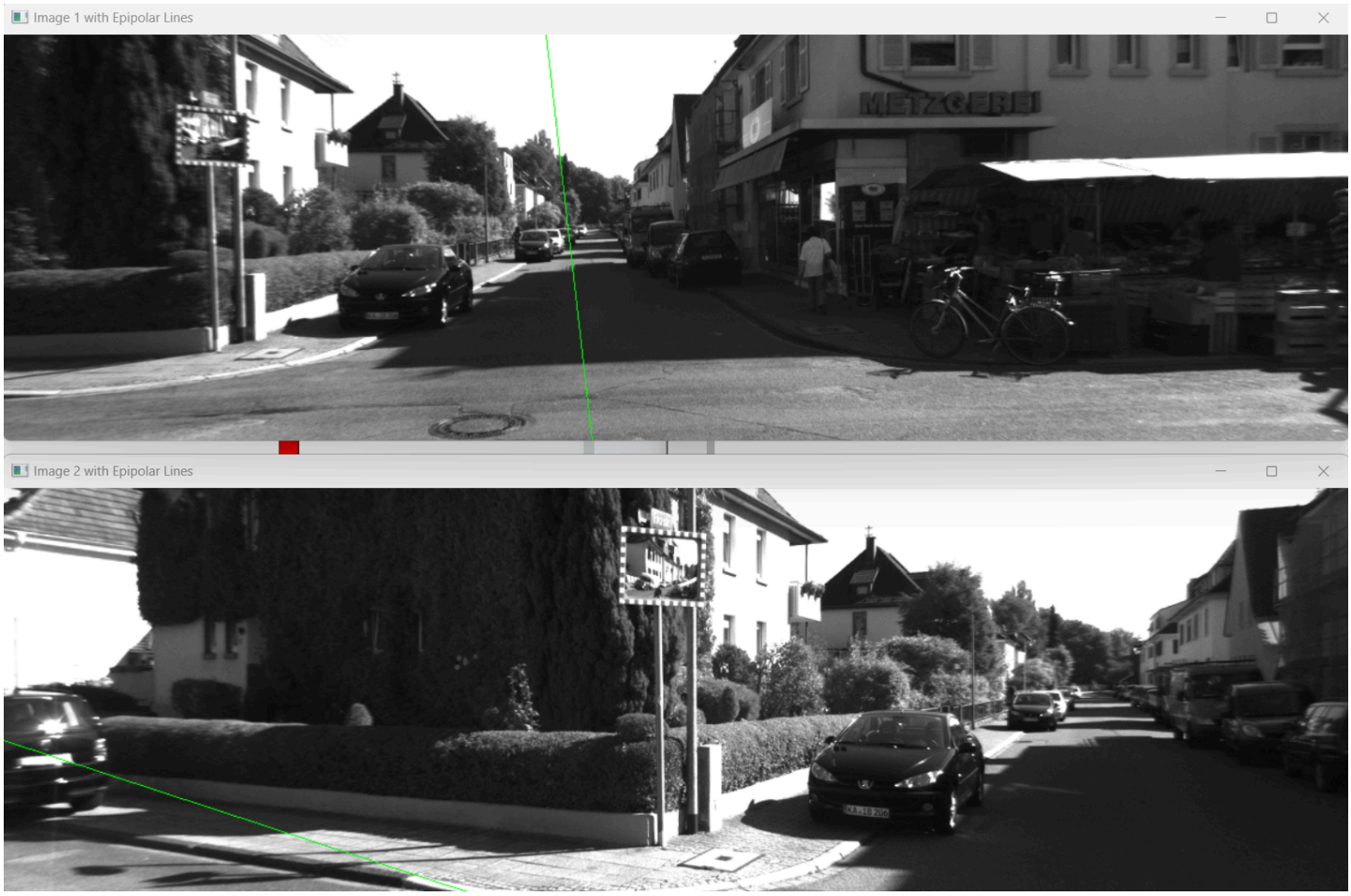
Epipolar lines represent the projection of points from one image onto the other image's plane. Given a point (x,y) in one image, the corresponding epipolar line $l^{'}$ in the other image can be calculated using the fundamental matrix F.

$$l^{'} = F.\ (x, y, 1)$$

The epipolar line l in the first image corresponding to a point in the second image can be obtained similarly using the transpose of F.

$$l = F^{T}.\ (x^{'}, y^{'}, 1)$$

Where, $(x^{'}, y^{'})$ are the coordinates of the corresponding point in the second image.



Observations:

- Selecting uniformly spaced pixels on epipolar lines allowed for efficient pixel correspondence between stereo images.
- The process of finding corresponding pixels on epipolar lines aided in establishing point matches without exhaustive search.

References:

- https://docs.opencv.org/4.x/dc/d0d/tutorial_py_features_harris.html
- https://github.com/muthuspark/ml_research/blob/master/Process%20of%20Harris%20Corner%20Detection%20Algorithm.ipynb
- https://medium.com/analytics-vidhya/distance-estimation-cf2f2fd709d8