Coursework 3 Report

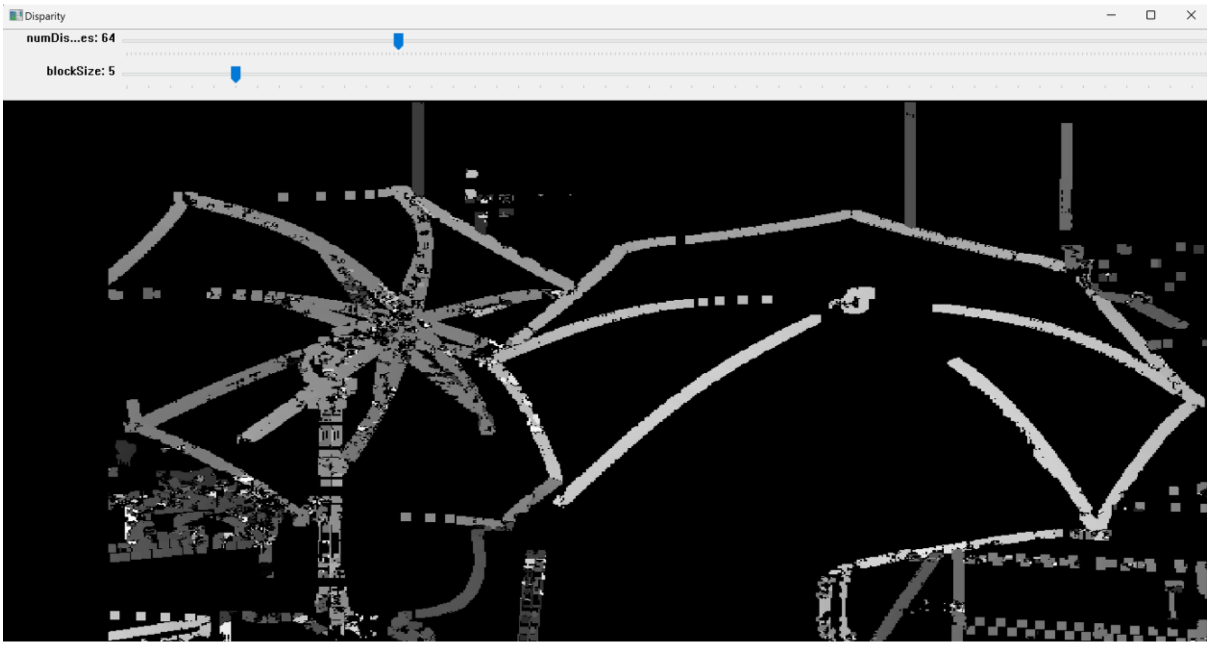
Usman Ali

1.1 Calculate Focal Length

The focal length was calculated using the width of the camera sensor size and dividing this by width of the resolution. The resulting number was then multiplied by the focal length in pixels.

$$F_{mm} = F_{px} \text{ x } (Sw / Iw)$$
$$F_{mm} = 5806.559 \text{ x } (22.2 / 3088)$$
$$F_{mm} = 41.74 \text{ mm (2dp)}$$

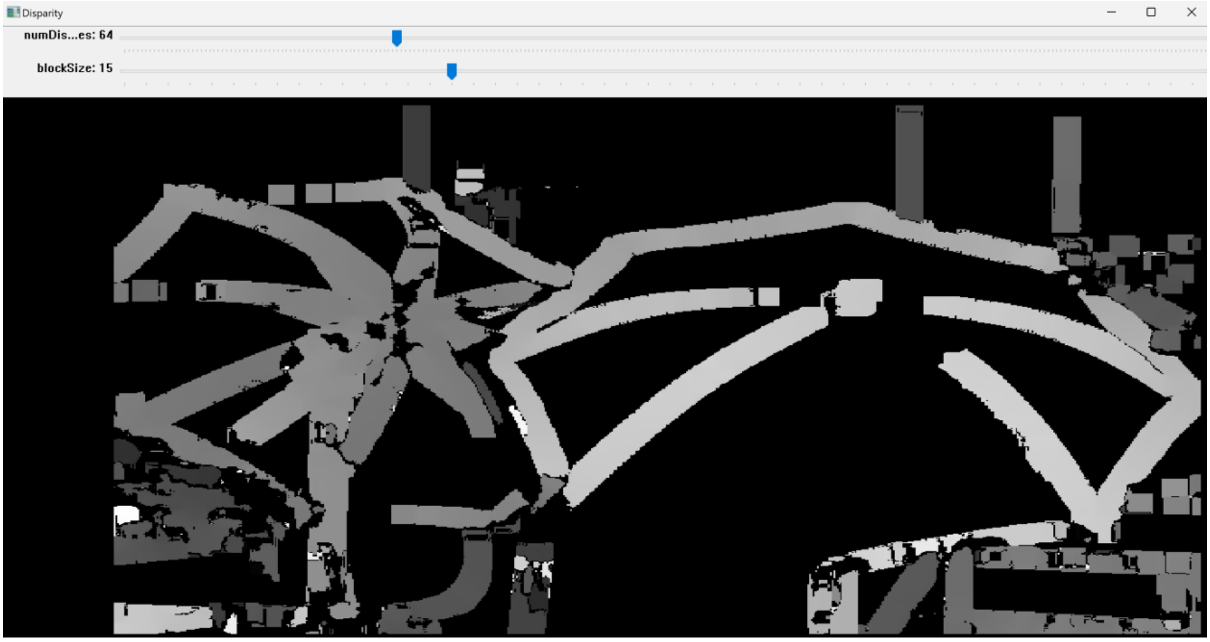
1.2 Disparity Map



numDis...es: 64

blockSize: 5

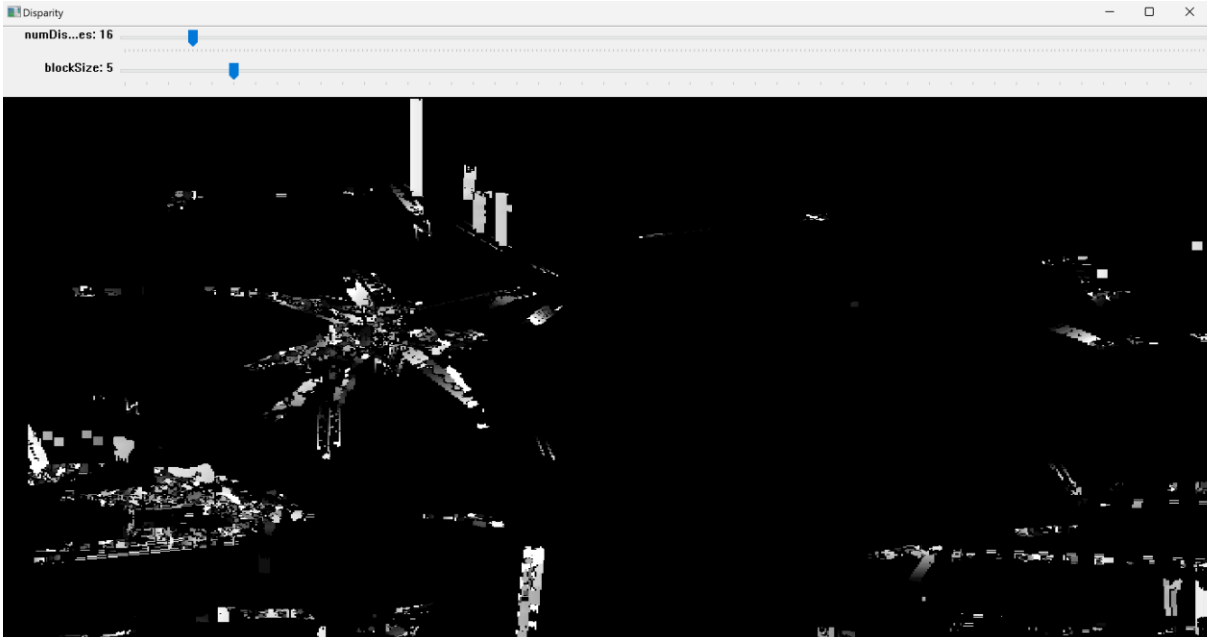
Figure 1 Disparity Image with blocksize of 5 and numOfDisparities at 64



numDis...es: 64

blockSize: 15

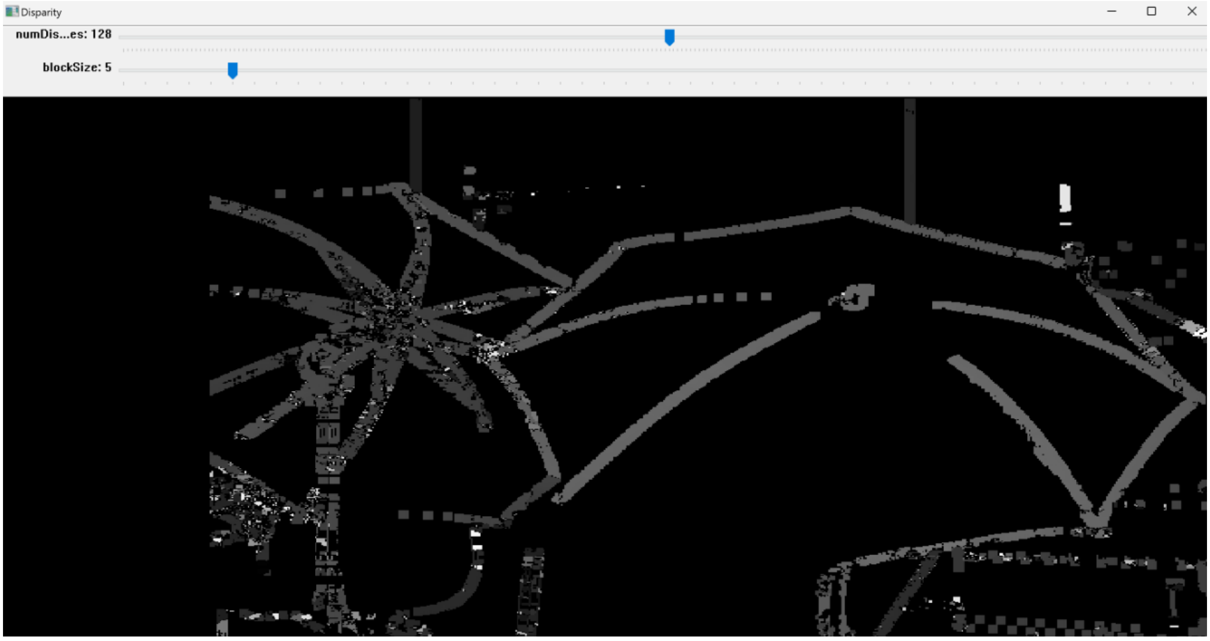
Figure 2 Disparity Image with blocksize of 15 and numOfDisparities at 64



numDis...es: 16

blockSize: 5

Figure 3 Disparity Image with blocksize of 5 and numOfDisparities at 16



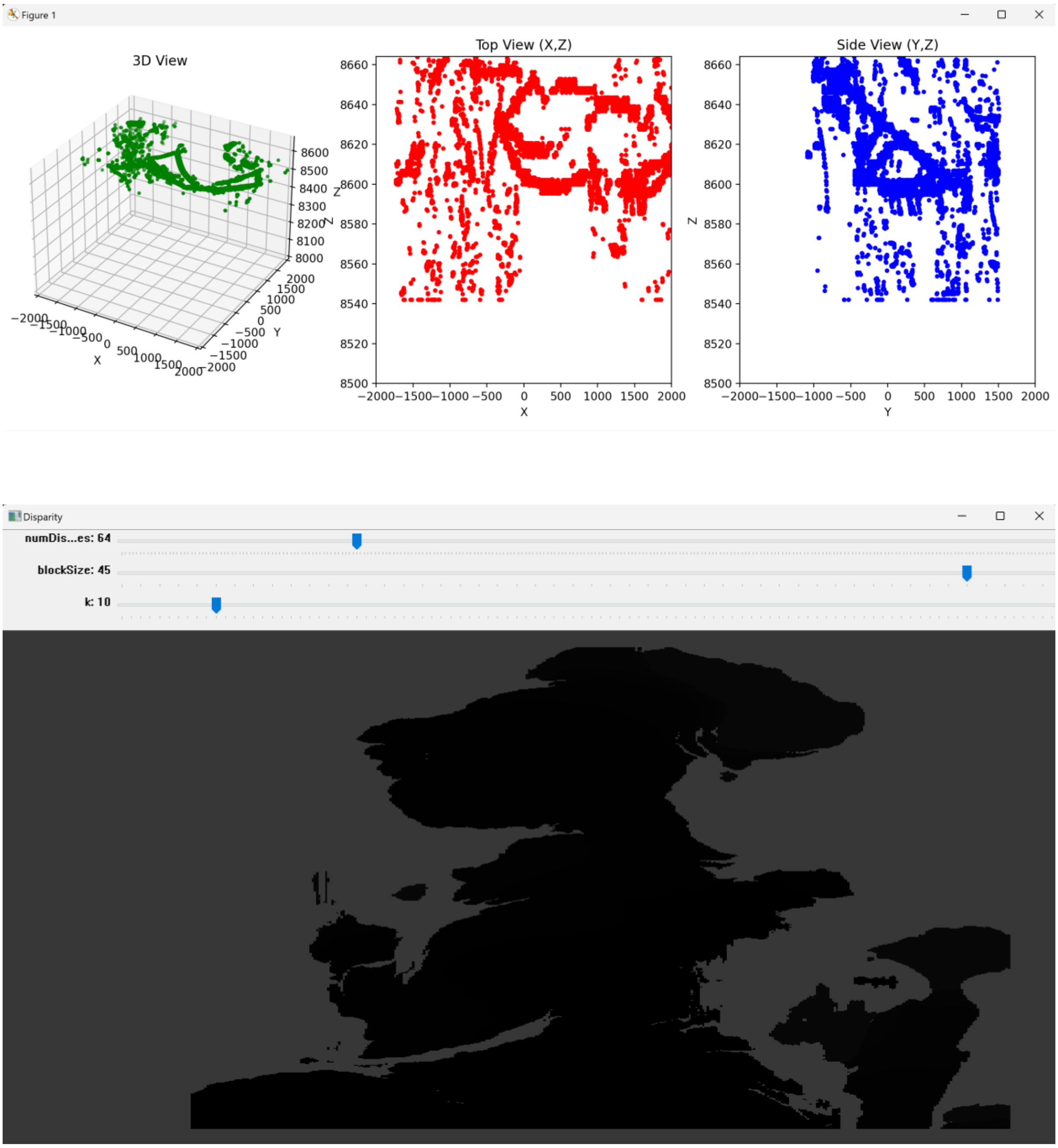
numDis...es: 128

blockSize: 5

Figure 4 Disparity Image with blocksize of 5 and numOfDisparities at 128

The **blockSize** defines the size of the comparison window used to find the corresponding points between the left and right images. The robustness of disparity calculations can be enhanced by increasing the **blockSize** because more pixels are needed to establish correspondence. The result of this change can be seen when comparing Figure 2 with Figure 1. Figure 2 has a larger **blockSize**, and as a result the details in the image, represented by the thicker lines, are a lot smoother and there is less noise present. Although a smaller **blockSize** can retain more detail and identify edges and small objects more accurately, it may struggle in low-textured areas and be more vulnerable to noise.

The number of disparities represents the number of horizontal shifts that the algorithm will test to find correspondences between the left and right images. The depth range that the system is able to detect is directly affected by this parameter. A comparison between Figure 3 (disparity parameter set at 16) and Figure 4 (disparity parameter set at 128) allows us to see the effect of increasing the number of disparities. This increase results in a disparity map that is more detailed and captures a wider range of depth. In contrast, the disparity map seems less detailed and only the most significant shifts are depicted when the number of disparities is set to a smaller value, such as 16. These larger or closer objects typically have more pronounced depth variations from the background.

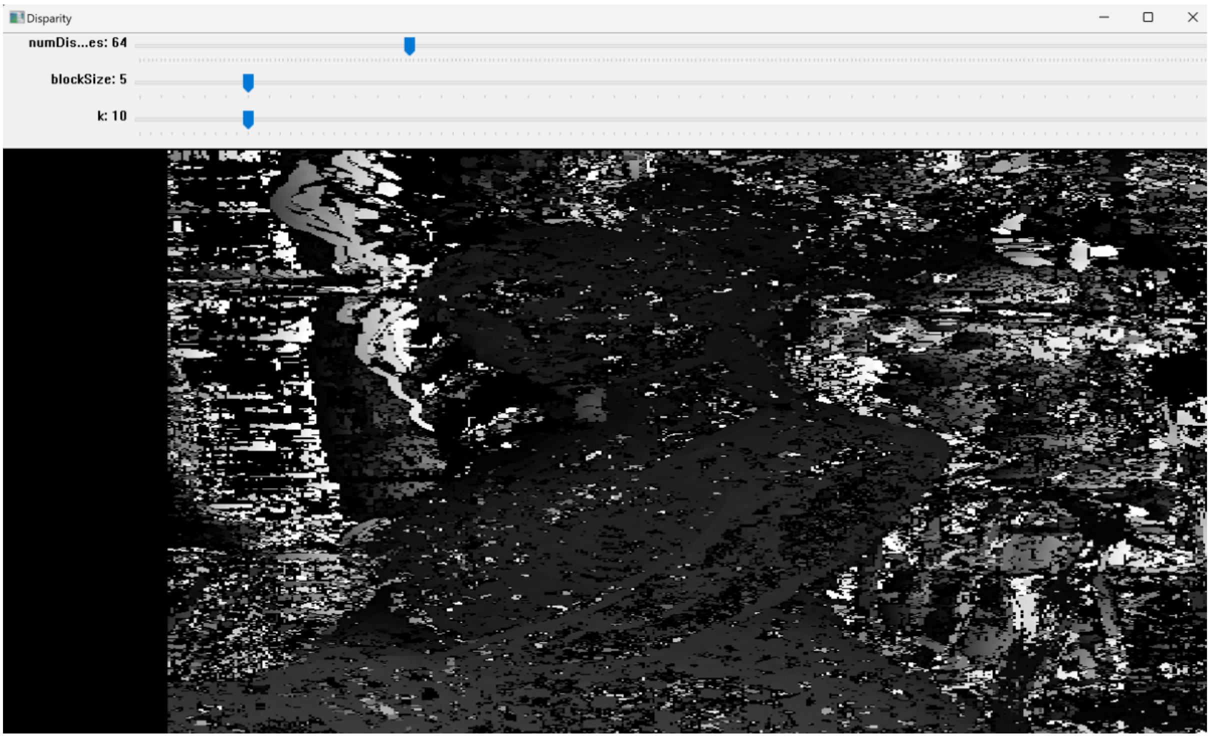


numDis...es: 64

blockSize: 45

k: 10

Figure 5

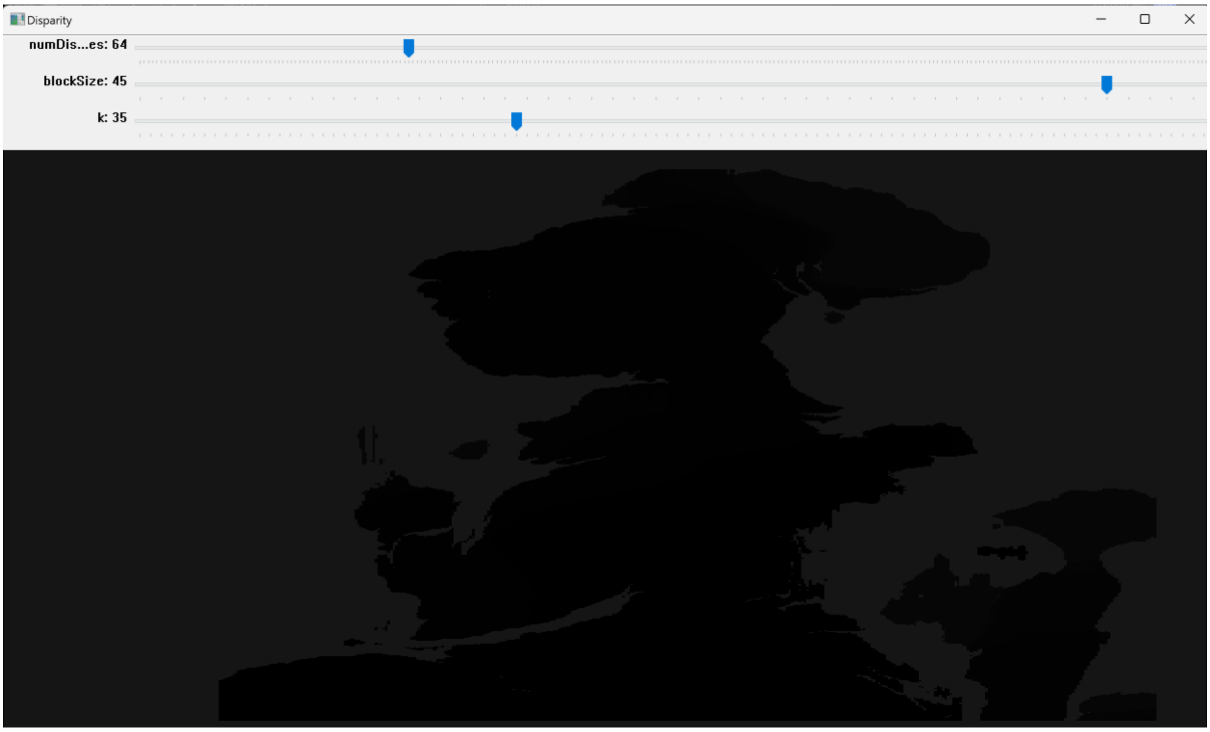


numDis...es: 64

blockSize: 5

k: 10

Figure 6



numDis...es: 64

blockSize: 45

k: 35

Figure 7

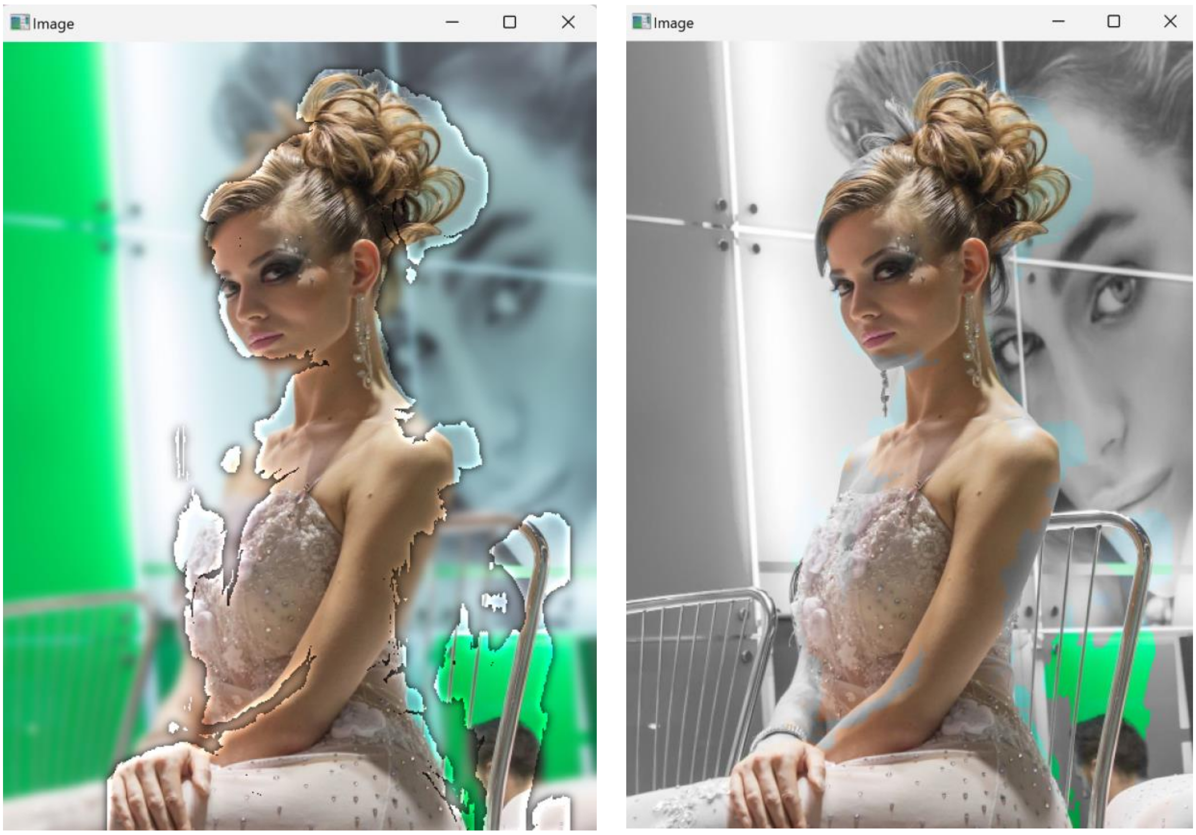


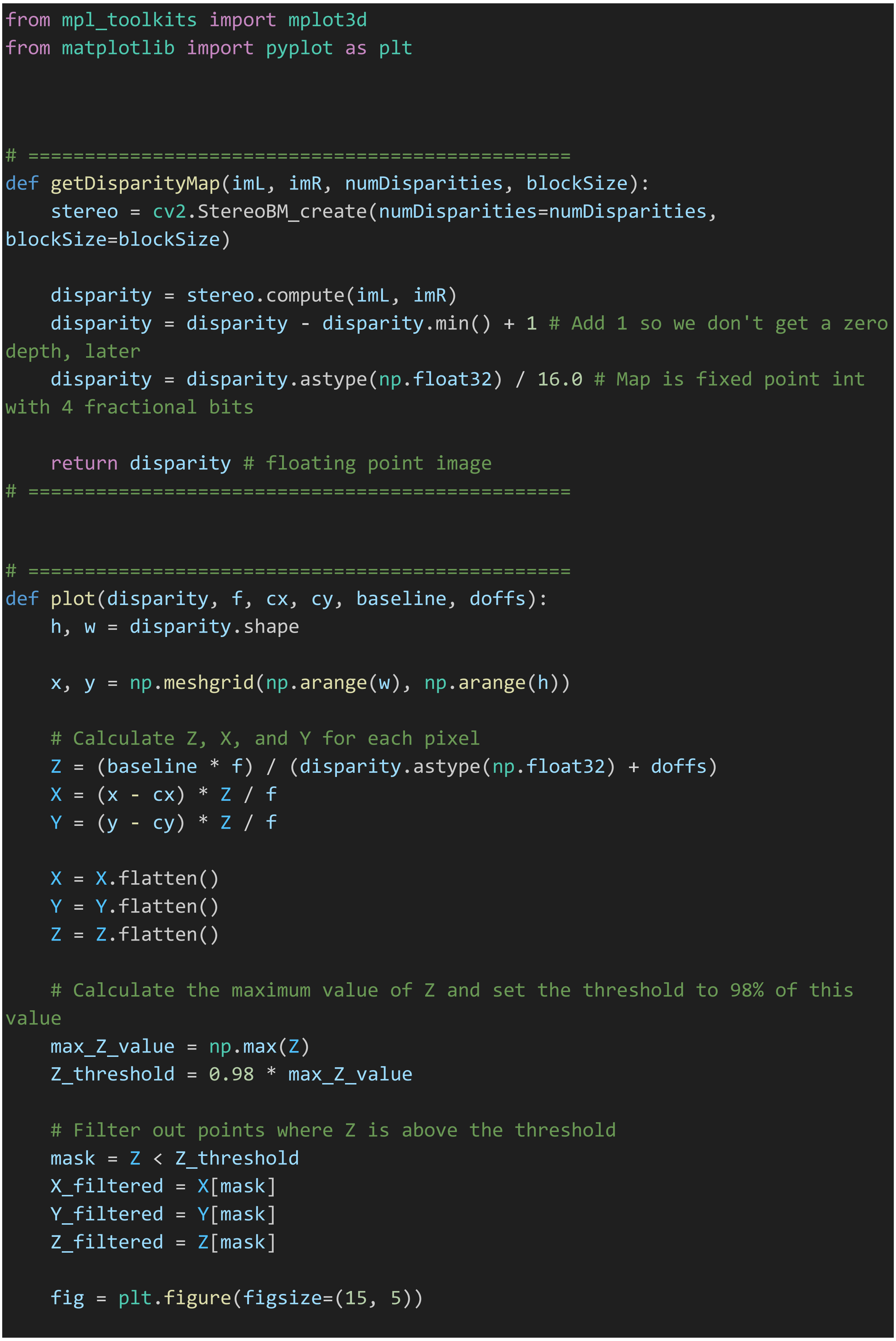
Figure 8

depth = 1/ (disparity + k)

In the depth calculation formula, the variable “k" serves as a control to avoid division by zero and to adjust how disparity changes affect the depth values that are obtained. A smaller k value increases the depth estimation's responsiveness to disparity alterations, a greater k value ensures a smoother depth map and mitigates noise.

Appendix

```
import numpy as np
import cv2
import sys
```



```
from mpl_toolkits import mplot3d
from matplotlib import pyplot as plt



# ================================================
def getDisparityMap(imL, imR, numDisparities, blockSize):
    stereo = cv2.StereoBM_create(numDisparities=numDisparities,
blockSize=blockSize)

    disparity = stereo.compute(imL, imR)
    disparity = disparity - disparity.min() + 1 # Add 1 so we don't get a zero
depth, later
    disparity = disparity.astype(np.float32) / 16.0 # Map is fixed point int
with 4 fractional bits

    return disparity # floating point image
# ================================================


# ================================================
def plot(disparity, f, cx, cy, baseline, doffs):
    h, w = disparity.shape

    x, y = np.meshgrid(np.arange(w), np.arange(h))

    # Calculate Z, X, and Y for each pixel
    Z = (baseline * f) / (disparity.astype(np.float32) + doffs)
    X = (x - cx) * Z / f
    Y = (y - cy) * Z / f

    X = X.flatten()
    Y = Y.flatten()
    Z = Z.flatten()

    # Calculate the maximum value of Z and set the threshold to 98% of this
value
    max_Z_value = np.max(Z)
    Z_threshold = 0.98 * max_Z_value

    # Filter out points where Z is above the threshold
    mask = Z < Z_threshold
    X_filtered = X[mask]
    Y_filtered = Y[mask]
    Z_filtered = Z[mask]

    fig = plt.figure(figsize=(15, 5))
    # 3D plot
    ax1 = fig.add_subplot(131, projection='3d')
    ax1.scatter(X_filtered, Y_filtered, Z_filtered, c='green', marker='.')
    ax1.set_xlabel('X')
    ax1.set_ylabel('Y')
    ax1.set_zlabel('Z')
    ax1.set_xlim([-2000, 2000])
    ax1.set_ylim([-2000, 2000])
    ax1.set_zlim([8000, Z_threshold])
    ax1.title.set_text('3D View')

    # Top view (x,z)
    ax2 = fig.add_subplot(132)
    ax2.scatter(X_filtered, Z_filtered, c='red', marker='.')
    ax2.set_xlabel('X')
    ax2.set_ylabel('Z')
    ax2.set_xlim([-2000, 2000])
    ax2.set_ylim([8500, Z_threshold])
    ax2.title.set_text('Top View (X,Z)')

    # Side view (y,z)
    ax3 = fig.add_subplot(133)
    ax3.scatter(Y_filtered, Z_filtered, c='blue', marker='.')
    ax3.set_xlabel('Y')
    ax3.set_ylabel('Z')
    ax3.set_xlim([-2000, 2000])
    ax3.set_ylim([8500, Z_threshold])
    ax3.title.set_text('Side View (Y,Z)')

    plt.tight_layout()
    plt.show()

def calc_focal_length(f_px, sw ,iw):
    f_mm = f_px * (sw / iw)
    return f_mm

focal_length = calc_focal_length(5806.559, 22.2, 3088)
print(focal_length)

# Global variables to store the trackbar values
numDisparities = 64
blockSize = 5

def on_trackbar_disp(val):
    # global numDisparities
    if val % 16 == 0 and val != 0:
        numDisparities = val
    disparity = getDisparityMap(edgesL, edgesR, numDisparities, blockSize)
    # disparity = getDisparityMap(imgL, imgR, numDisparities, blockSize)
    disparityImg = np.interp(disparity, (disparity.min(), disparity.max()),
(0.0, 1.0))
    cv2.imshow('Disparity', disparityImg)


def on_trackbar_block(val):
    #global blockSize
    blockSize = val
    blockSize = max(5, blockSize + (blockSize % 2 == 0))
    disparity = getDisparityMap(edgesL, edgesR, numDisparities, blockSize)
    # disparity = getDisparityMap(imgL, imgR, numDisparities, blockSize)
    disparityImg = np.interp(disparity, (disparity.min(), disparity.max()),
(0.0, 1.0))
    cv2.imshow('Disparity', disparityImg)


f_original = 5806.559  # Focal length in pixels for the original image size
cx_original = 1429.219  # The x-coordinate of the principal point for the
original image size
cy_original = 993.403  # The y-coordinate of the principal point for the
original image size
doffs_original = 114.291  # The x-difference of the principal points
baseline = 174.019  # The camera baseline in millimeters

# Original and resized image dimensions
width_original, height_original = 2960, 2016
width_resized, height_resized = 740, 505

scale_factor = width_resized / width_original

f = scale_factor * f_original
cx0 = scale_factor * cx_original
cy = scale_factor * cy_original
doffs = scale_factor * doffs_original

if __name__ == '__main__':

    # Load left image
    filename = 'umbrellaL.png'
    imgL = cv2.imread(filename, cv2.IMREAD_GRAYSCALE)
    if imgL is None:
        print('\nError: failed to open {}.\n'.format(filename))
        sys.exit()
    edgesL = cv2.Canny(imgL, 50, 150)

    # Load right image
    filename = 'umbrellaR.png'
    imgR = cv2.imread(filename, cv2.IMREAD_GRAYSCALE)
    if imgR is None:
        print('\nError: failed to open {}.\n'.format(filename))
        sys.exit()
    edgesR = cv2.Canny(imgR, 50, 150)


    # Initialize the disparity settings
    numDisparities = 64
    blockSize = 5

    cv2.namedWindow('Disparity', cv2.WINDOW_NORMAL)
    cv2.createTrackbar('numDisparities', 'Disparity', numDisparities, 256,
on_trackbar_disp)
    cv2.createTrackbar('blockSize', 'Disparity', blockSize, 50,
on_trackbar_block)

    # Recalculate the disparity map with the new parameters
    disparity = getDisparityMap(edgesL, edgesR, numDisparities, blockSize)
    disparityImg = np.interp(disparity, (disparity.min(), disparity.max()),
(0.0, 1.0))
    cv2.imshow('Disparity', disparityImg)
    plot(disparityImg, f, cx0, cy, baseline, doffs)

    while True:
        key = cv2.waitKey(1)
        if key == ord(' ') or key == 27:
            break

    cv2.destroyAllWindows()
```

File 2:

```
def on_trackbar_k(val):
    # global k

    k = val/100

    # Recalculate the disparity map with the new parameters
    disparity = getDisparityMap(imgL, imgR, numDisparities, blockSize)
    depth_map = cv2.divide(1.0, cv2.add(disparity, k))
    disparityImg = np.interp(depth_map, (disparity.min(), disparity.max()),
(0.0, 1.0))
```

```
    cv2.imshow('Disparity', disparityImg)

def process_background(image, depth_map, k, blur=False, grayscale=False):
    # Threshold the depth map to create a mask for the foreground
    _, foreground_mask = cv2.threshold(depth_map, k, 255, cv2.THRESH_BINARY)
    foreground_mask = foreground_mask.astype(np.uint8)
    foreground_mask_3c = cv2.cvtColor(foreground_mask, cv2.COLOR_GRAY2BGR)
    foreground = cv2.bitwise_and(image, foreground_mask_3c)

    background_mask = cv2.bitwise_not(foreground_mask)
    background_mask_3c = cv2.cvtColor(background_mask, cv2.COLOR_GRAY2BGR)
    background = cv2.bitwise_and(image, background_mask_3c)

    if blur:
        foreground = cv2.GaussianBlur(foreground, (21, 21), 0)
    if grayscale:
        foreground = cv2.cvtColor(foreground, cv2.COLOR_BGR2GRAY)
        foreground = cv2.cvtColor(foreground, cv2.COLOR_GRAY2BGR)

    combined_image = cv2.add(foreground, background)

    return combined_image


if __name__ == '__main__':

    # Load left image
    filename = 'girlL.png'
    imgL = cv2.imread(filename, cv2.IMREAD_GRAYSCALE)
    if imgL is None:
        print('\nError: failed to open {}.\n'.format(filename))
        sys.exit()

    # Load right image
    filename = 'girlR.png'
    image = cv2.imread(filename, cv2.IMREAD_COLOR)
    imgR = cv2.imread(filename, cv2.IMREAD_GRAYSCALE)
    if imgR is None:
        print('\nError: failed to open {}.\n'.format(filename))
        sys.exit()

    numDisparities = 64
    blockSize = 45
    k2 = 10
    k = 0.99

    # Create a window to display the image in
    cv2.namedWindow('Disparity', cv2.WINDOW_NORMAL)
    cv2.createTrackbar('numDisparities', 'Disparity', numDisparities, 256,
on_trackbar_disp)
    cv2.createTrackbar('blockSize', 'Disparity', blockSize, 50,
on_trackbar_block)
    cv2.createTrackbar('k', 'Disparity', k2, 100, on_trackbar_k)

    disparity = getDisparityMap(imgL, imgR, numDisparities=numDisparities,
blockSize=blockSize)
    disparityImg = np.interp(disparity, (disparity.min(), disparity.max()),
(0.0, 1.0))
    depth_map = cv2.divide(1.0, cv2.add(disparityImg, k))

    new_image = process_background(image, depth_map, k, blur=False,
grayscale=True)
    cv2.imshow('Image', new_image)

    while True:
        key = cv2.waitKey(1)
        if key == ord(' ') or key == 27:
            break
```