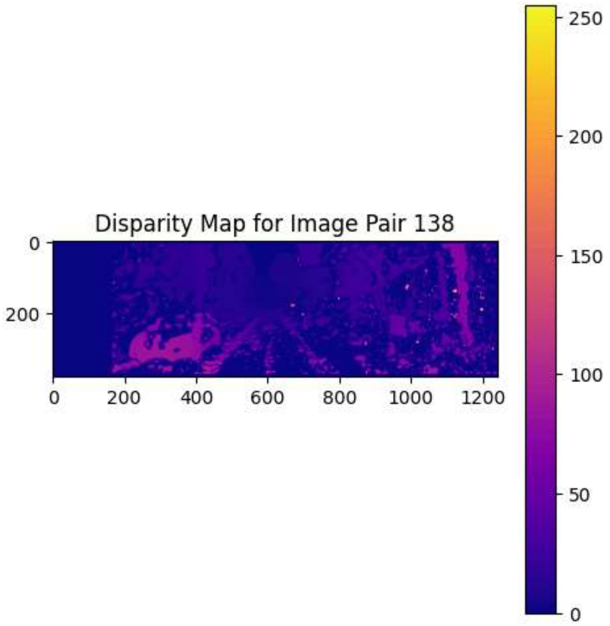
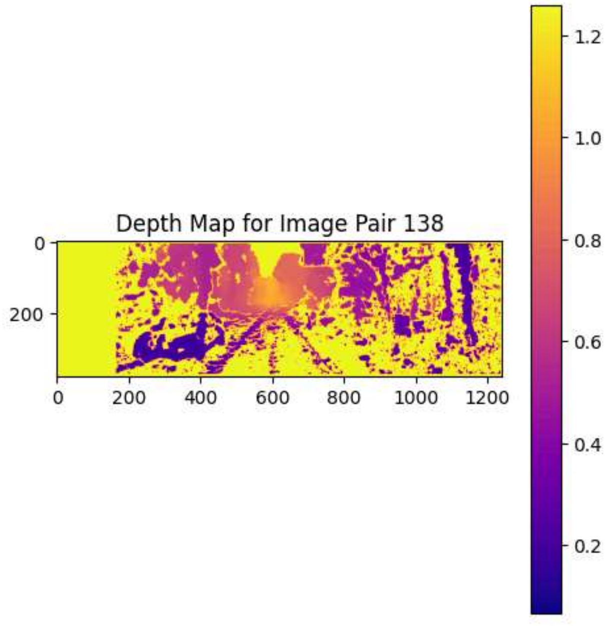
```
import cv2
import numpy as np
import os
import matplotlib.pyplot as plt
import random

# Paths to your stereo image directories
image_2_path = '/content/Image 2'  # Left images
image_3_path = '/content/Image 3'  # Right images

# Function to load images from a folder
def load_images_from_folder(folder):
    images = []
    for filename in sorted(os.listdir(folder)):
        img_path = os.path.join(folder, filename)
        img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
        if img is not None:
            images.append(img)
    return images

# Load left and right images
left_images = load_images_from_folder(image_2_path)
right_images = load_images_from_folder(image_3_path)

# Stereo block matching parameters
block_size = 15
min_disp = 0
num_disp = 160  # Number of disparities (increase based on your data)

# Focal length and baseline (for depth calculation)
focal_length = 21  # Example value (in mm, adjust based on your camera setup)
baseline = 0.54  # Example value (in meters, adjust based on your camera setup)

# Create stereo block matcher
stereo = cv2.StereoBM_create(numDisparities=num_disp, blockSize=block_size)

# Randomly select 10 pairs of images
num_images_to_display = 10
indices = random.sample(range(len(left_images)), num_images_to_display)

# For each randomly selected pair, compute disparity map, depth map and 3D reconstruction
for idx in indices:
    left_img = left_images[idx]
    right_img = right_images[idx]

    # Compute disparity map
    disparity = (stereo.compute(left_img, right_img).astype(np.float32) / 16.0) + 10

    # Normalize disparity for better visualization
    disparity_normalized = cv2.normalize(disparity, None, 0, 255, cv2.NORM_MINMAX)
    disparity_normalized = np.uint8(disparity_normalized)

    # Calculate depth map (D = f * B / d)
    # Convert disparity to depth using the formula
    depth_map = (focal_length * baseline) / (disparity + 1e-5)  # Adding small epsilon to avoid division by 

    # Create a 3D point cloud from the depth map
    height, width = depth_map.shape
    points_3D = []
    for y in range(height):
        for x in range(width):
            # Get depth value at this pixel
            Z = depth_map[y, x]
            if Z > 0:  # Only consider valid depth values
```

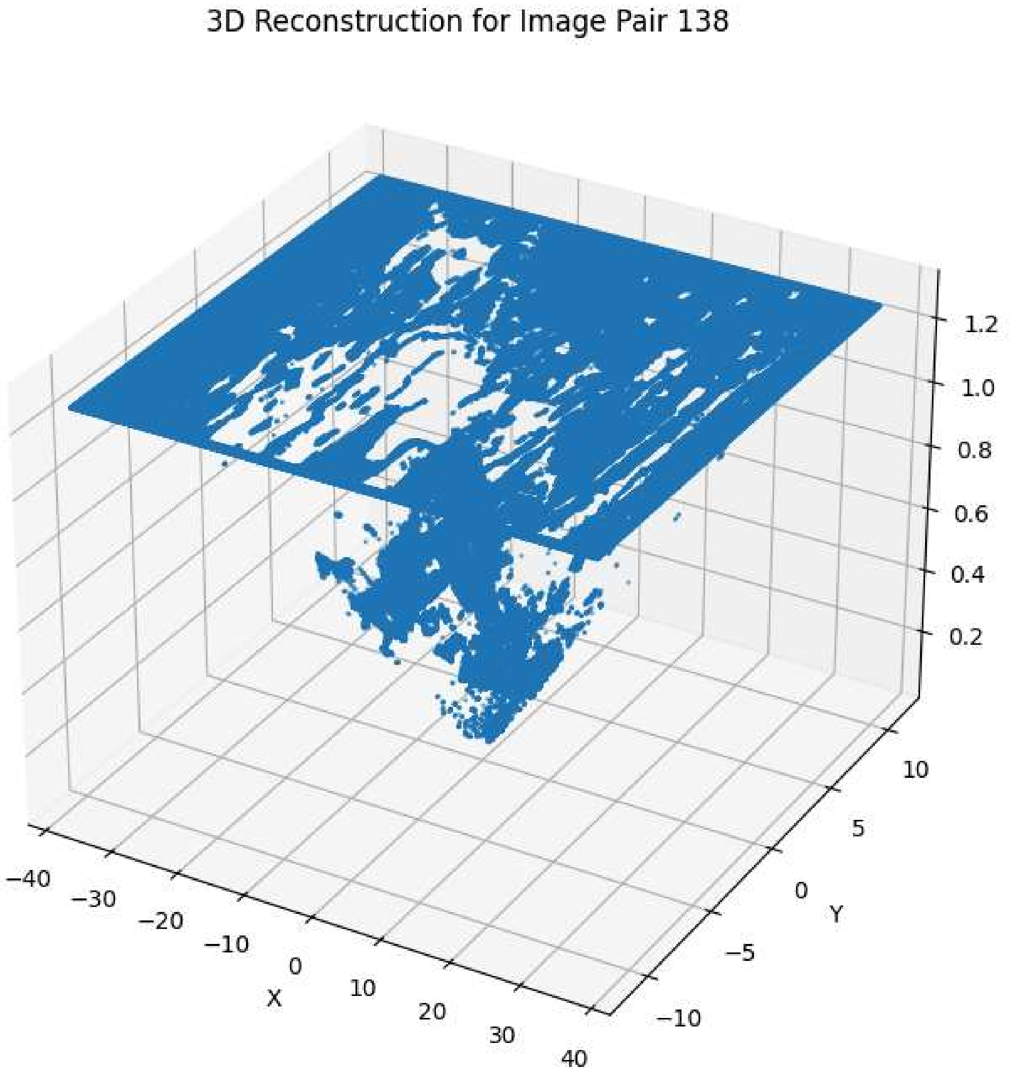
```
            # Reconstruct the 3D coordinates using the camera intrinsic parameters
            X = (x - width / 2) * Z / focal_length
            Y = (y - height / 2) * Z / focal_length
            points_3D.append([X, Y, Z])

    # Convert points to numpy array for visualization or further processing
    points_3D = np.array(points_3D)

    # Display the disparity and depth maps
    plt.figure(figsize=(12, 6))

    # Disparity Map Visualization
    plt.subplot(1, 2, 1)
    plt.imshow(disparity_normalized, cmap='plasma')
    plt.title(f'Disparity Map for Image Pair {idx + 1}')
    plt.colorbar()

    # Depth Map Visualization
    plt.subplot(1, 2, 2)
    plt.imshow(depth_map, cmap='plasma')
    plt.title(f'Depth Map for Image Pair {idx + 1}')
    plt.colorbar()

    plt.show()

    # Plot 3D point cloud using matplotlib
    fig = plt.figure(figsize=(10, 8))
    ax = fig.add_subplot(111, projection='3d')

    # Plot points in 3D space
    ax.scatter(points_3D[:, 0], points_3D[:, 1], points_3D[:, 2], s=1)
    ax.set_title(f'3D Reconstruction for Image Pair {idx + 1}')
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('Z')

    plt.show()

    # Optional: You can save the 3D point cloud to a file, like a .txt or .pcd for later use
    # For now, the point cloud is stored in `points_3D`
    # np.savetxt(f"point_cloud_{idx+1}.txt", points_3D)
```



Disparity Map for Image Pair 138
0
200
0 200 400 600 800 1000 1200
250
200
150
100
50
0

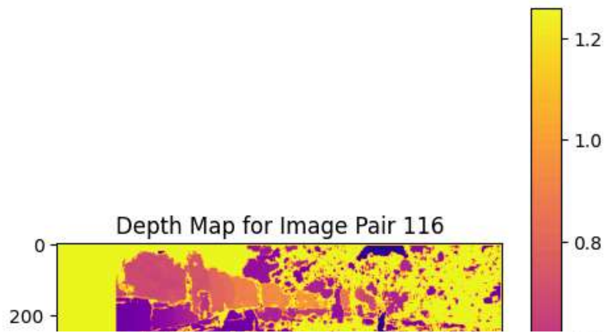


Depth Map for Image Pair 138
0
200
0 200 400 600 800 1000 1200
1.2
1.0
0.8
0.6
0.4
0.2

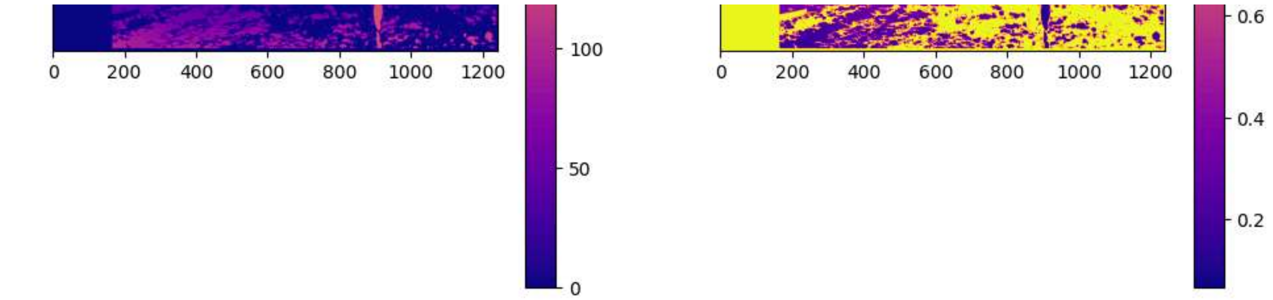


3D Reconstruction for Image Pair 138
1.2
1.0
0.8
0.6
0.4
0.2
−40 −30 −20 −10 0 10 20 30 40
X
10
5
0
−5
−10
Y

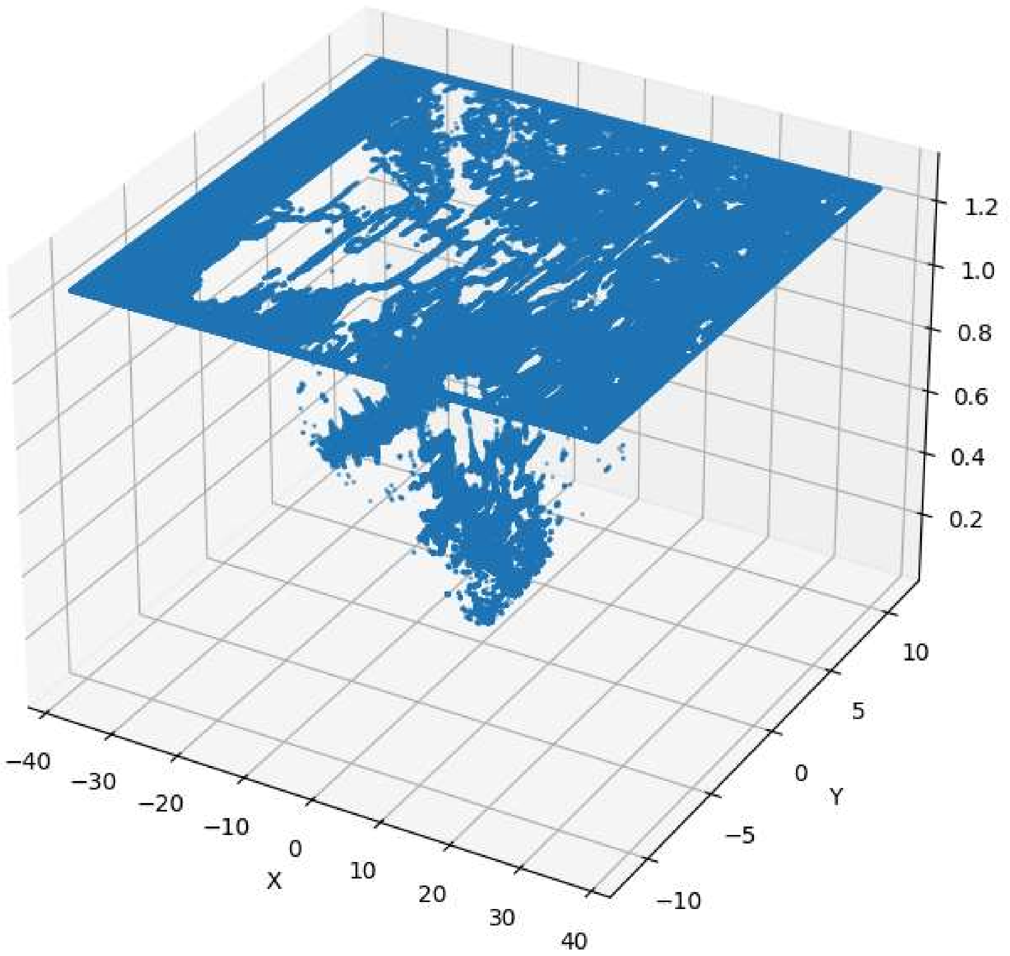
Disparity Map for Image Pair 116
0
200
250
200
150

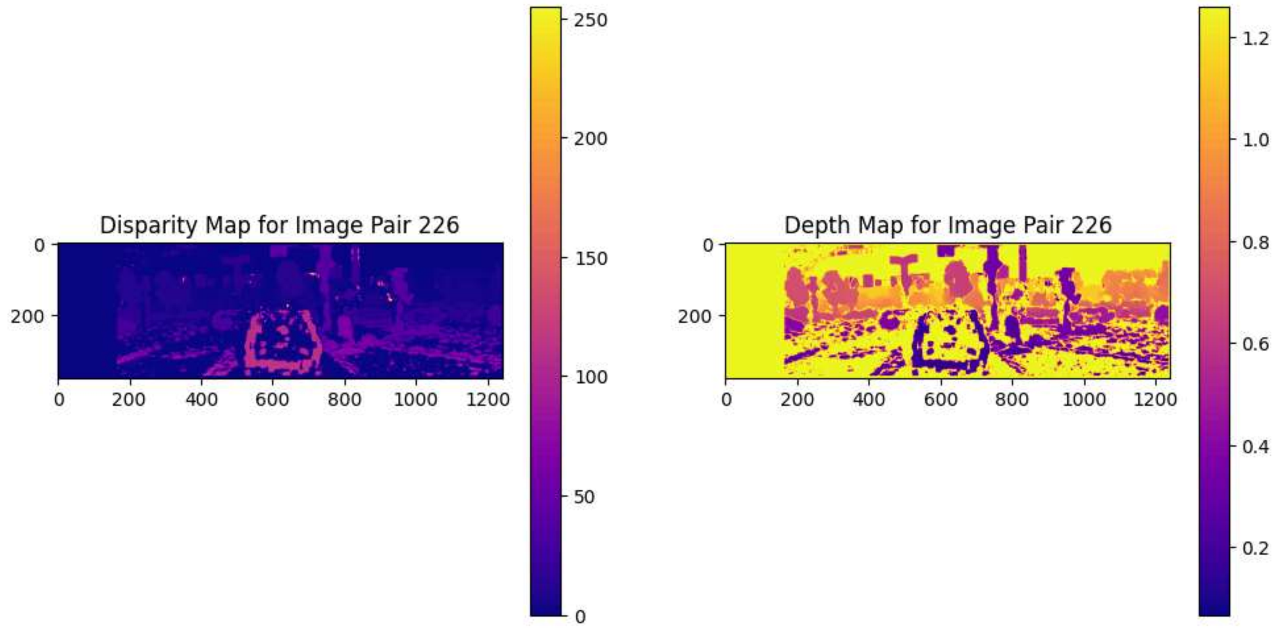


Depth Map for Image Pair 116
0
200
1.2
1.0
0.8

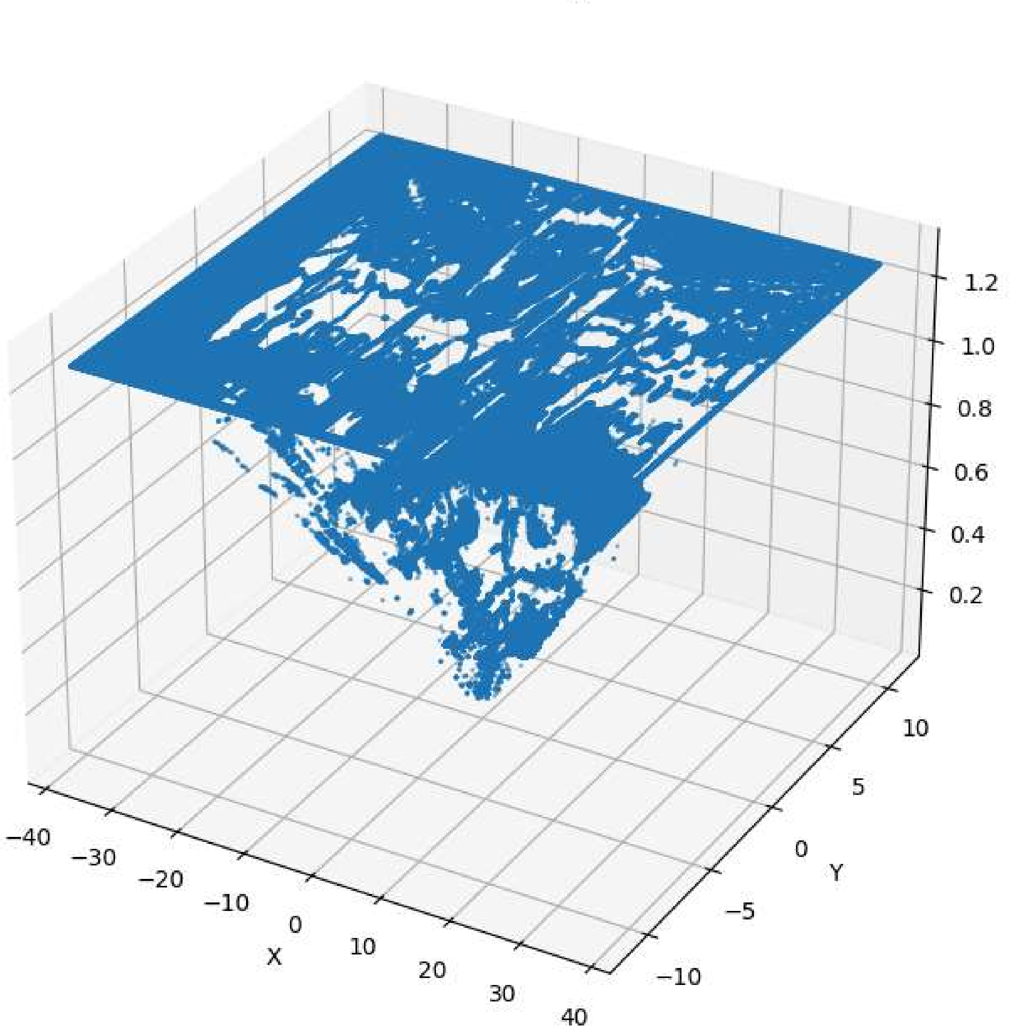


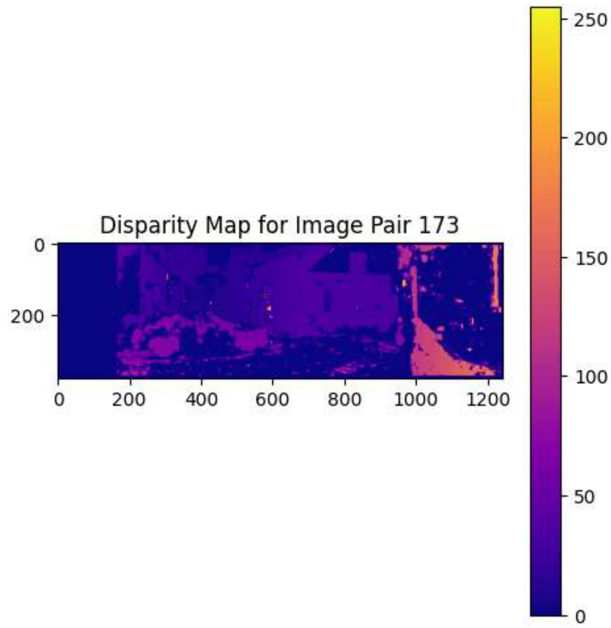
3D Reconstruction for Image Pair 116

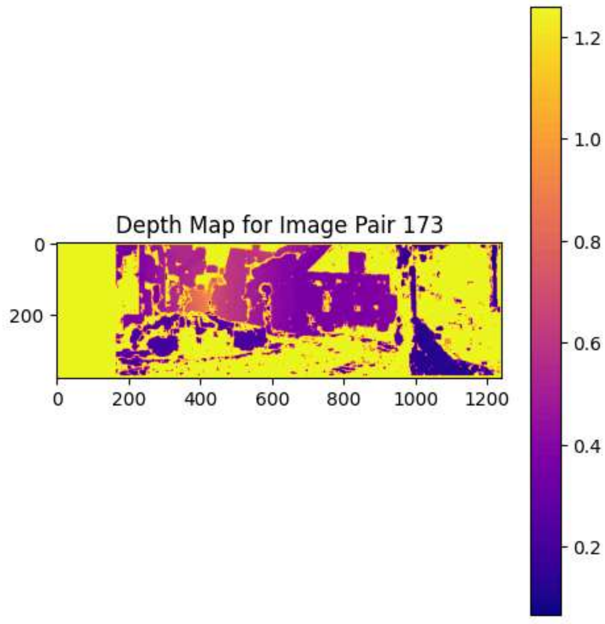




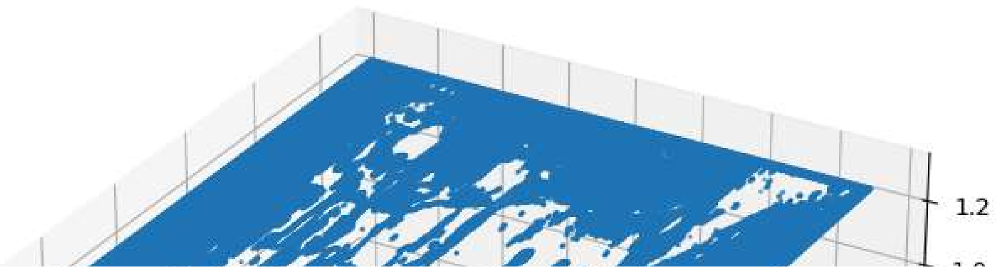
3D Reconstruction for Image Pair 226

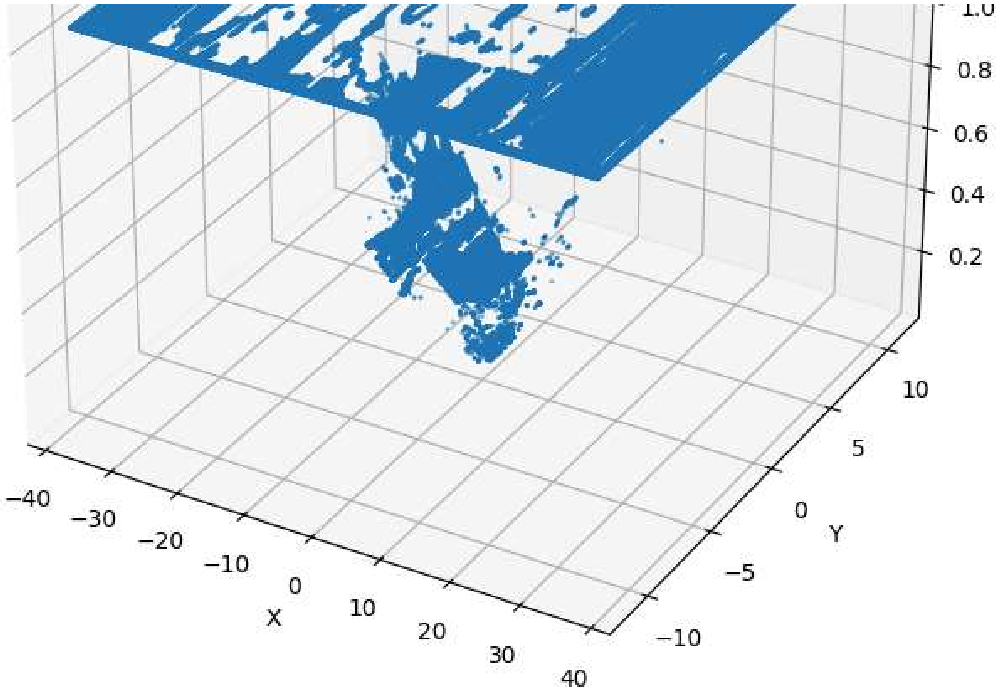


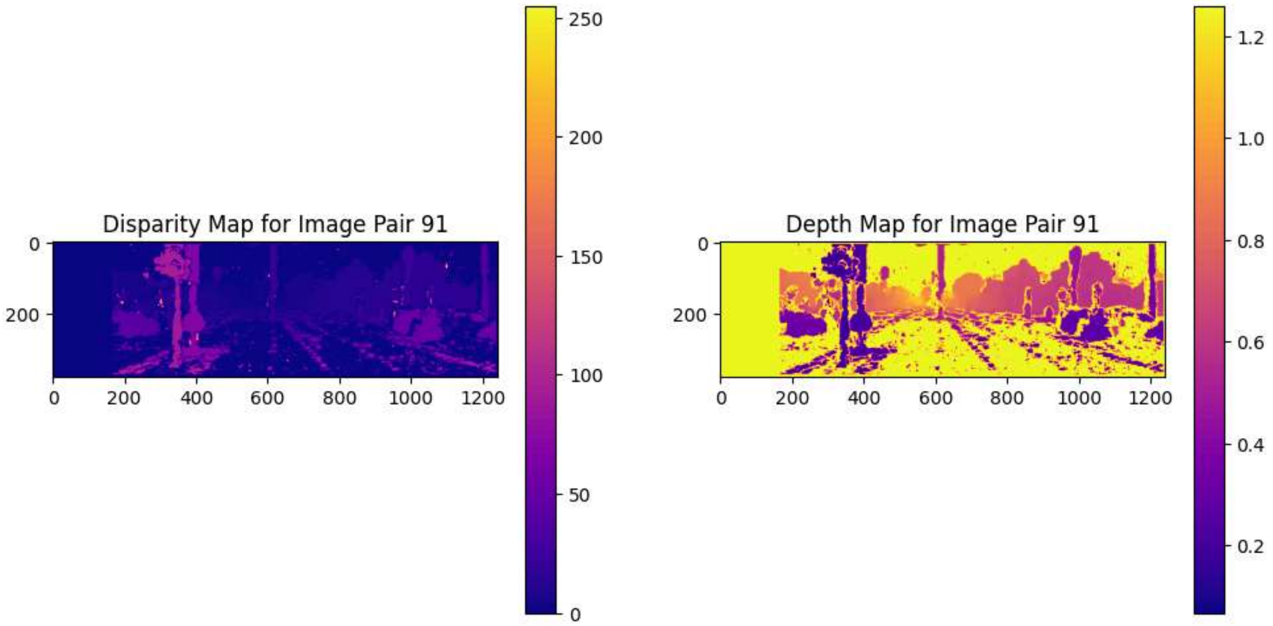




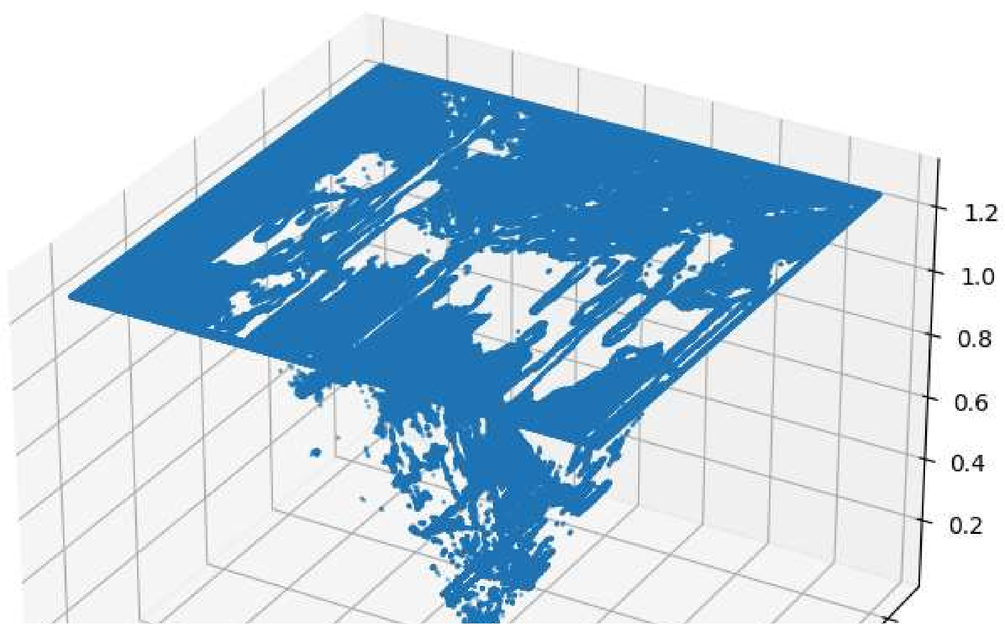
3D Reconstruction for Image Pair 173

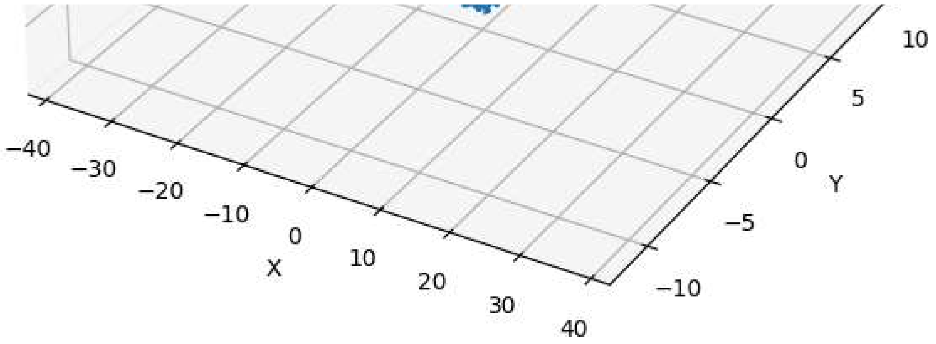


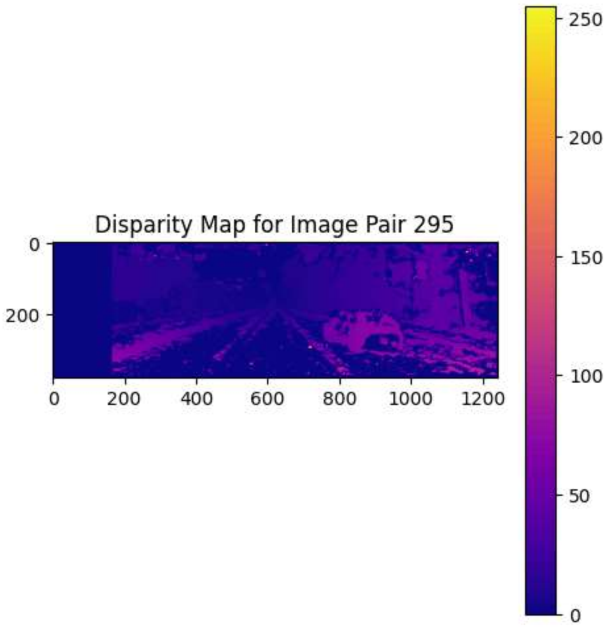


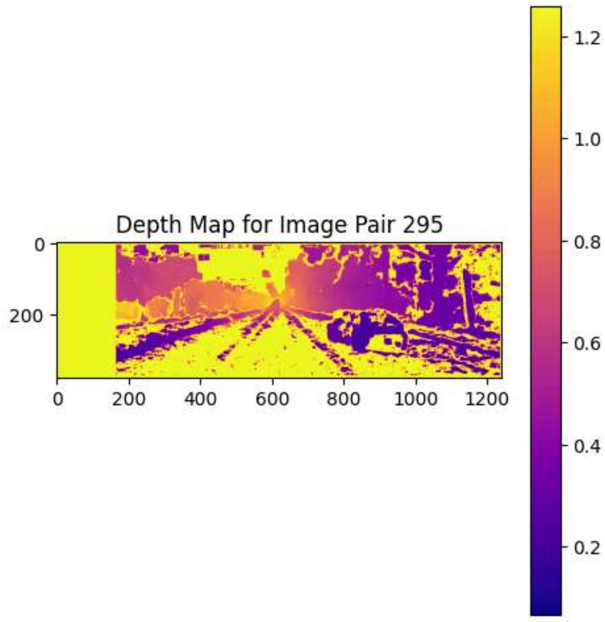


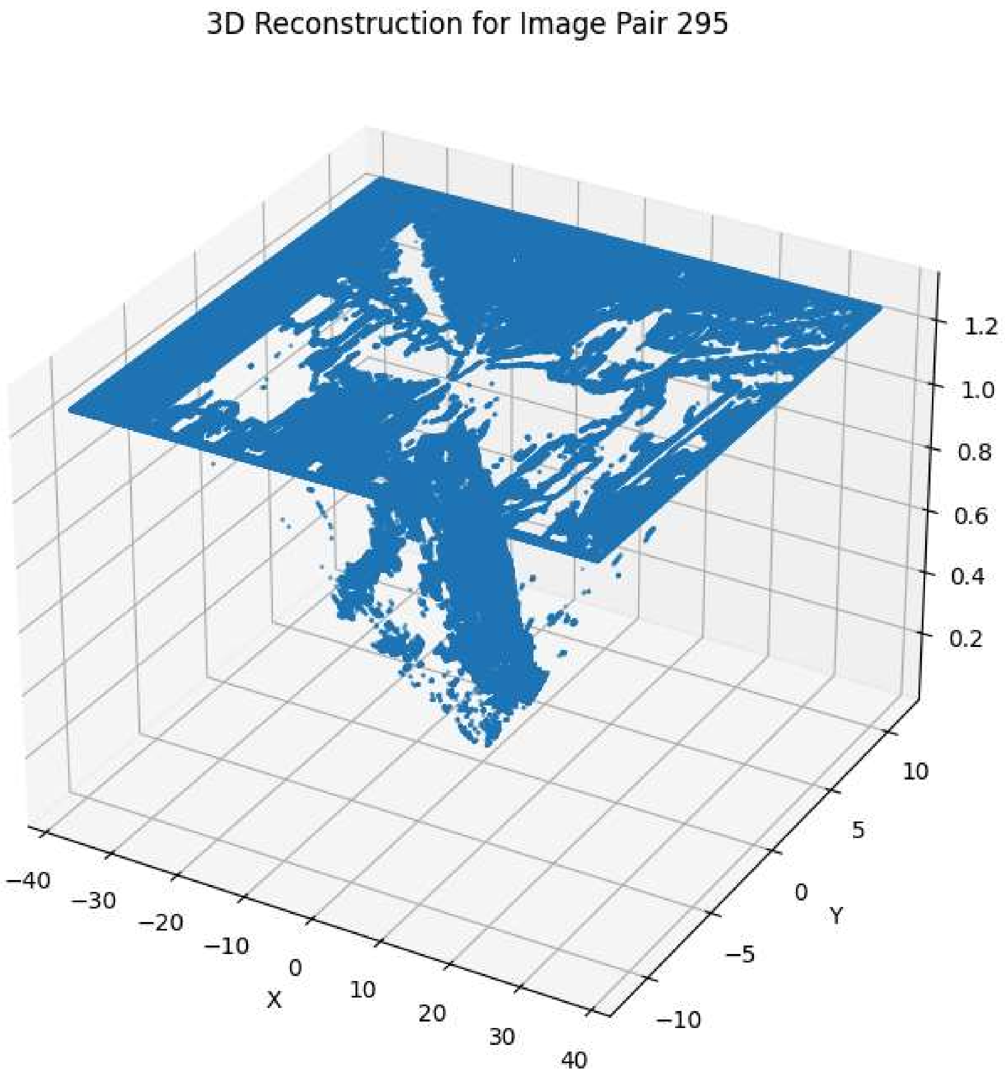
3D Reconstruction for Image Pair 91

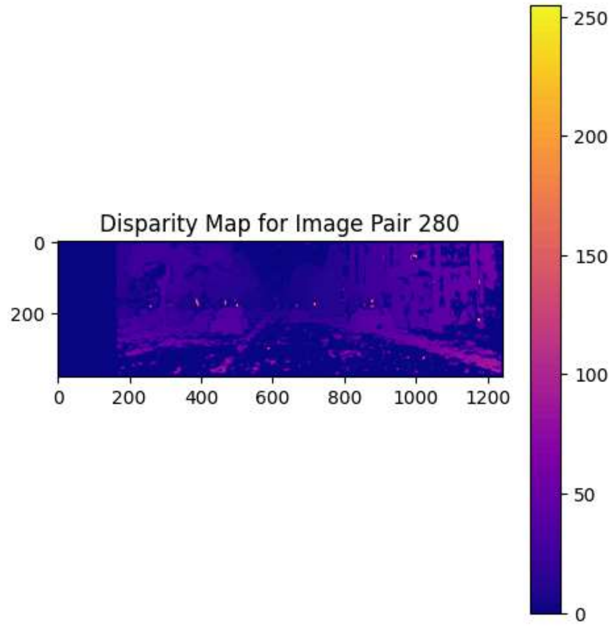


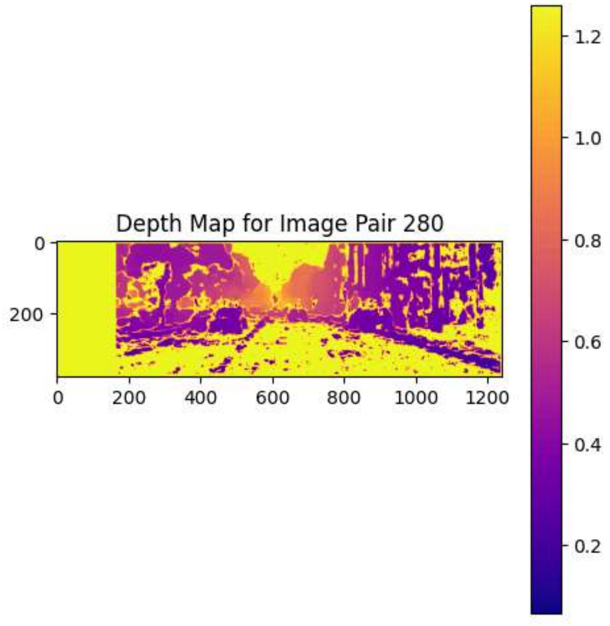

−40
−30
−20
−10
0
10
20
30
40
X
10
5
0
−5
−10
Y

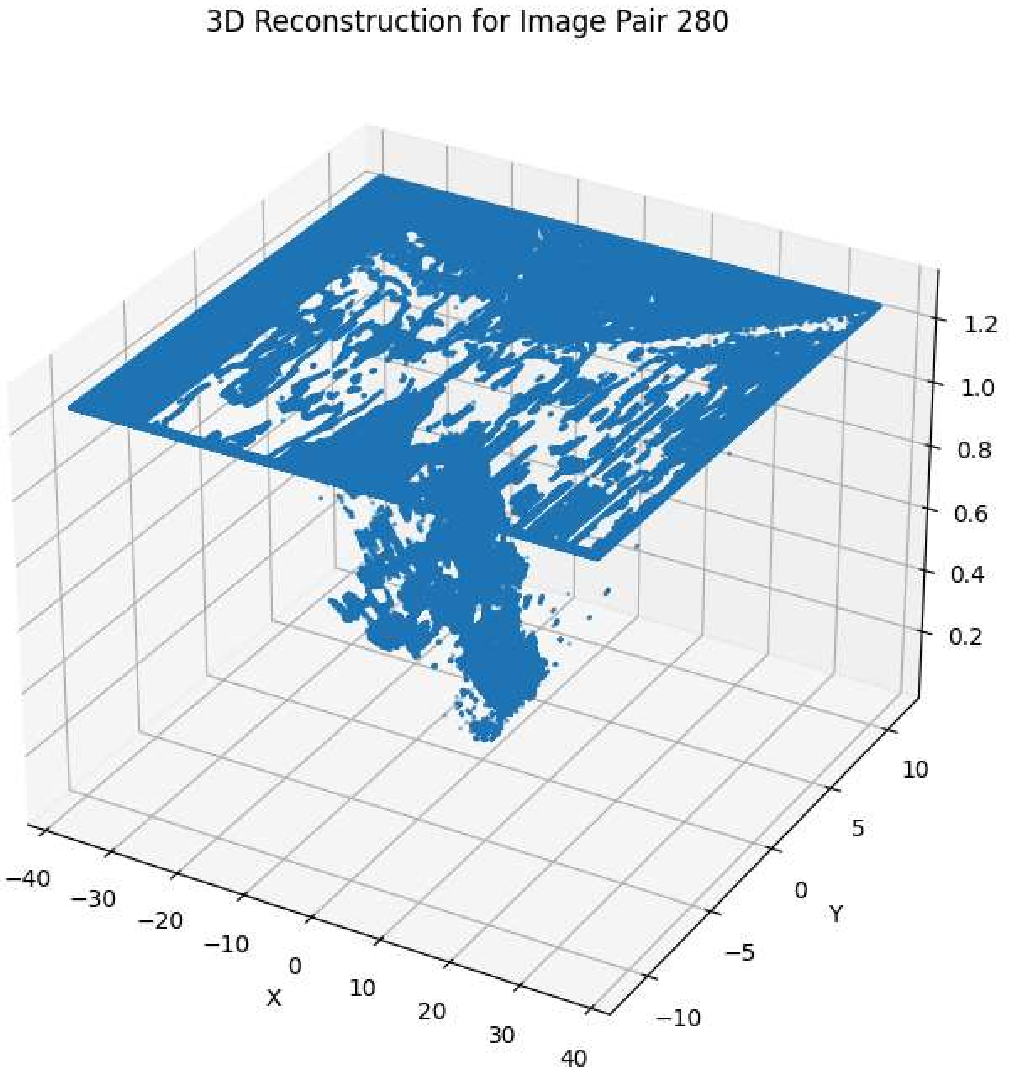

Disparity Map for Image Pair 295
0
200
0
200
400
600
800
1000
1200
250
200
150
100
50
0


Depth Map for Image Pair 295
0
200
0
200
400
600
800
1000
1200
1.2
1.0
0.8
0.6
0.4
0.2

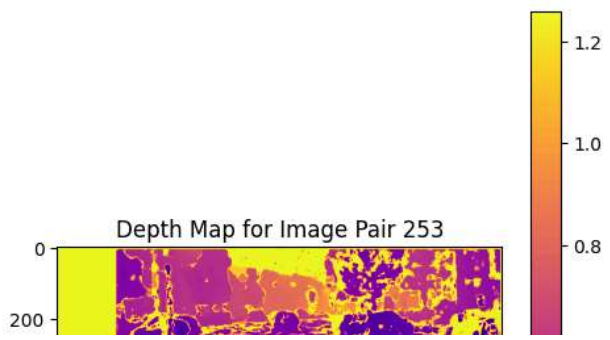

3D Reconstruction for Image Pair 295
1.2
1.0
0.8
0.6
0.4
0.2
−40
−30
−20
−10
0
10
20
30
40
X
10
5
0
−5
−10
Y

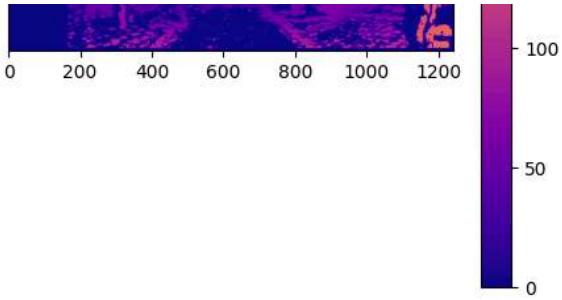

Disparity Map for Image Pair 280
0
200
0 200 400 600 800 1000 1200
250
200
150
100
50
0

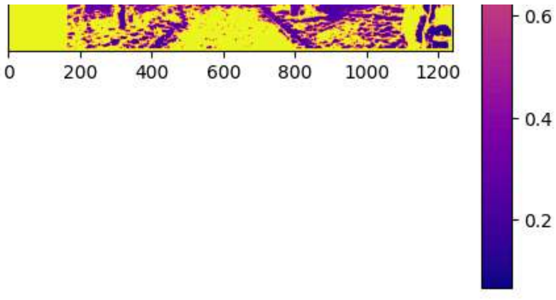

Depth Map for Image Pair 280
0
200
0 200 400 600 800 1000 1200
1.2
1.0
0.8
0.6
0.4
0.2


3D Reconstruction for Image Pair 280
1.2
1.0
0.8
0.6
0.4
0.2
−40 −30 −20 −10 0 10 20 30 40
X
10
5
0
−5
−10
Y

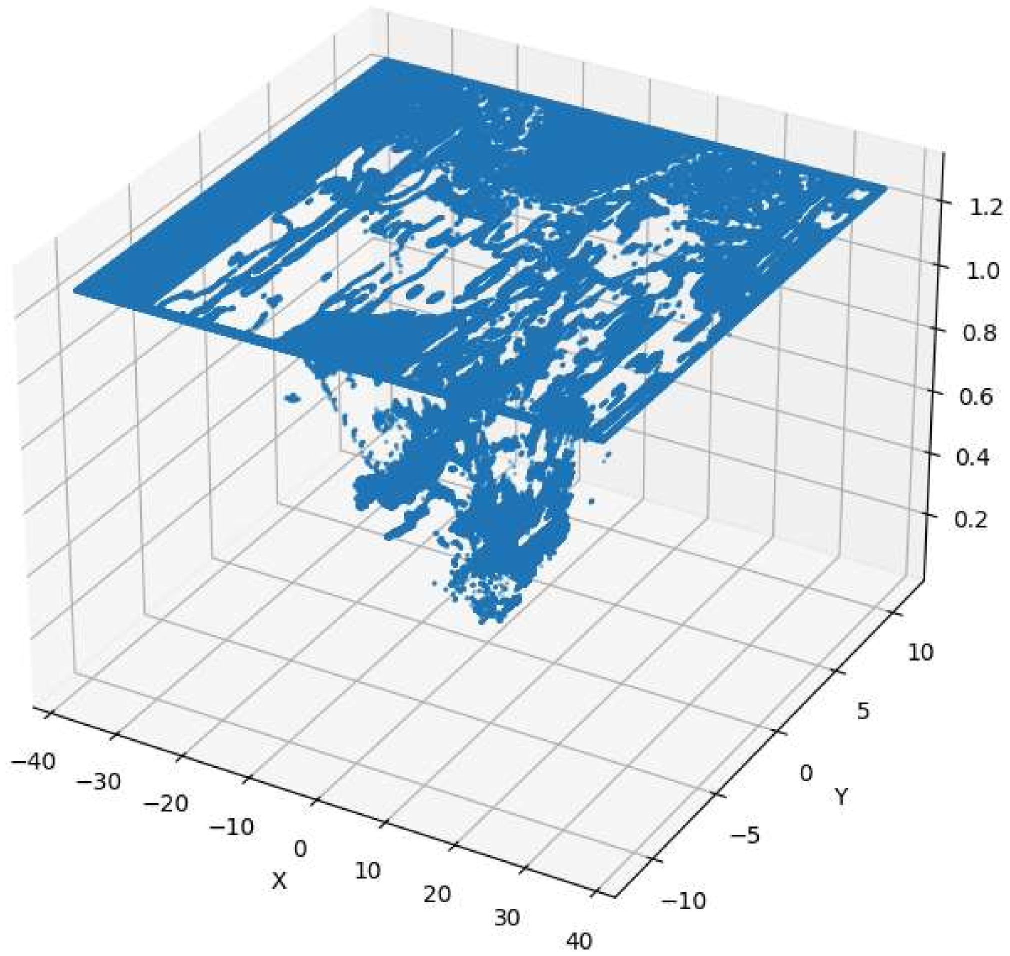

Disparity Map for Image Pair 253
0
200
250
200
150

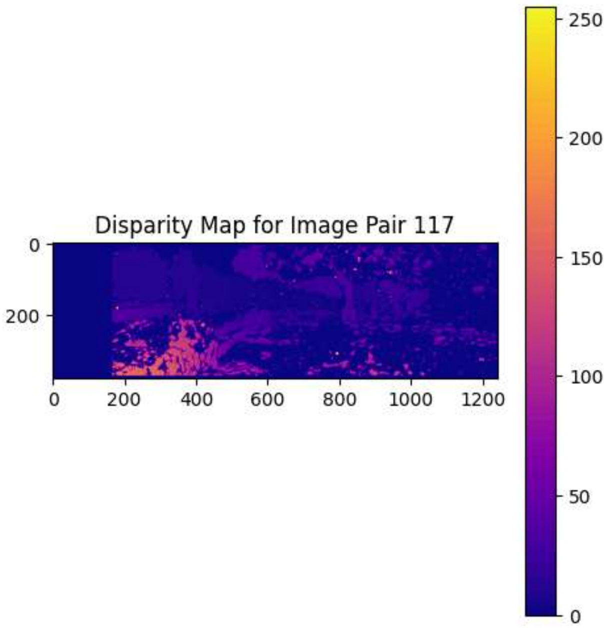

Depth Map for Image Pair 253
0
200
1.2
1.0
0.8

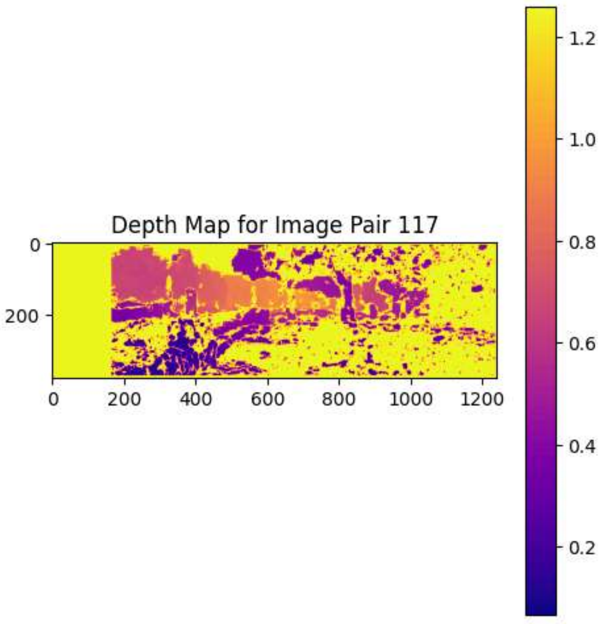




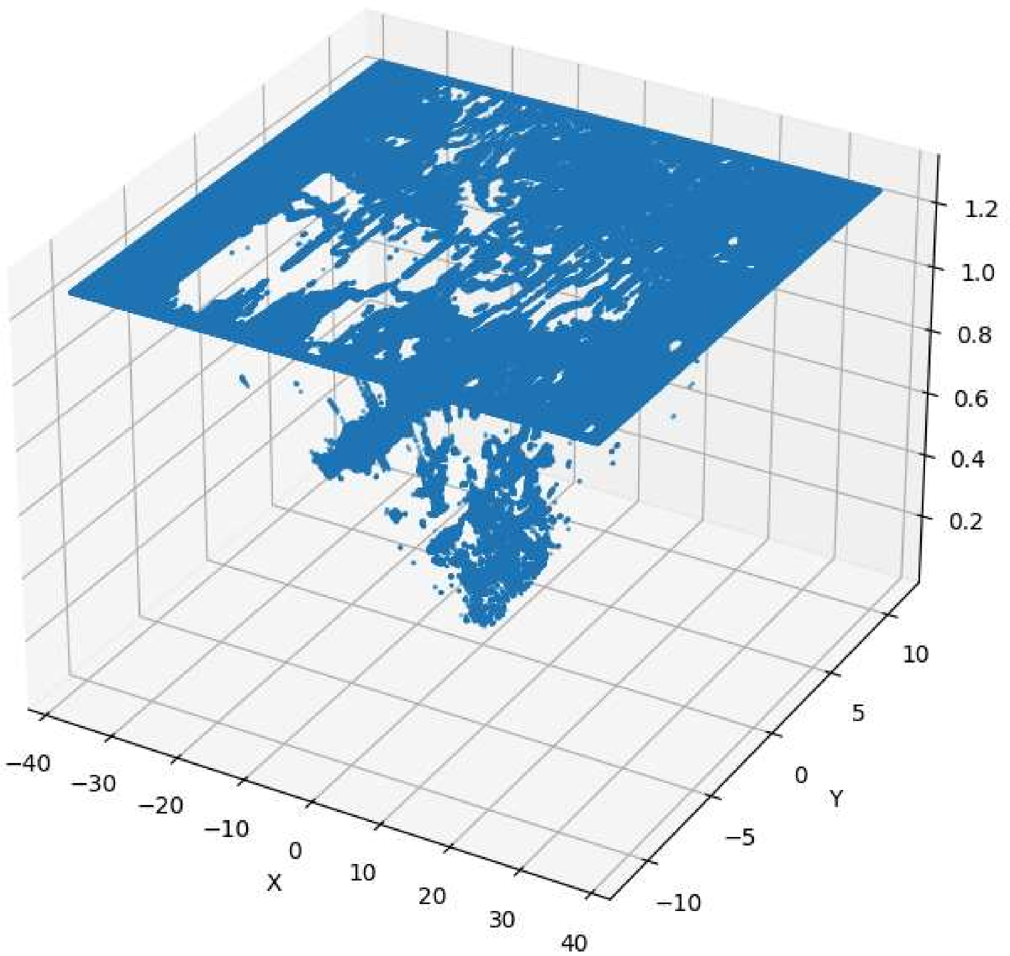
3D Reconstruction for Image Pair 253

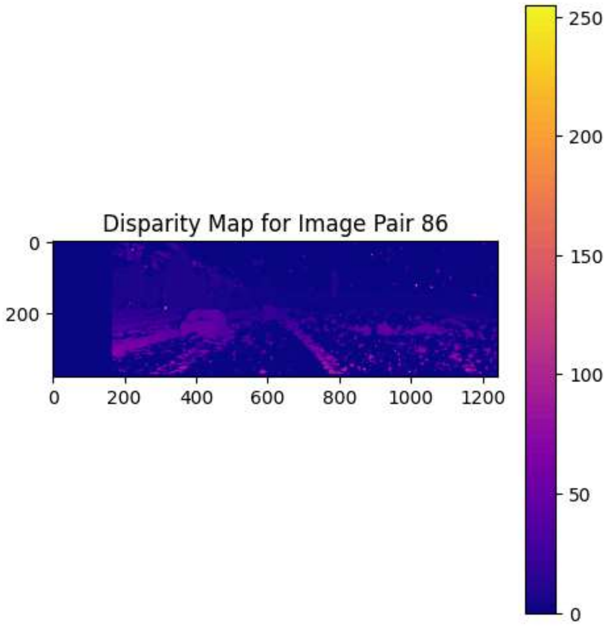


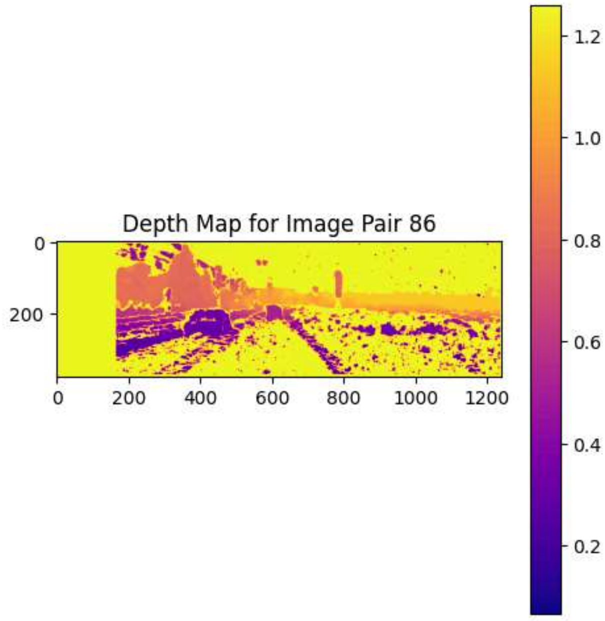




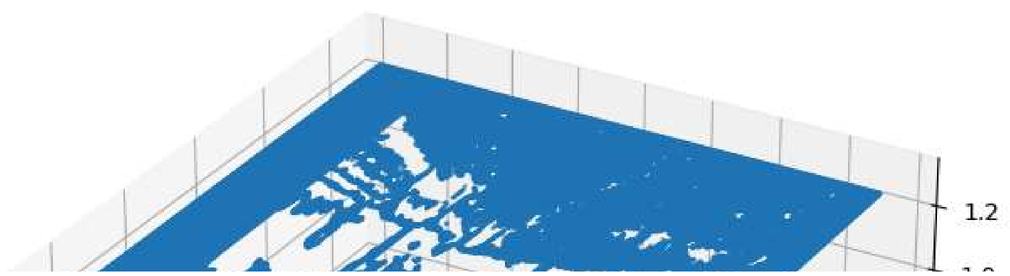
3D Reconstruction for Image Pair 117

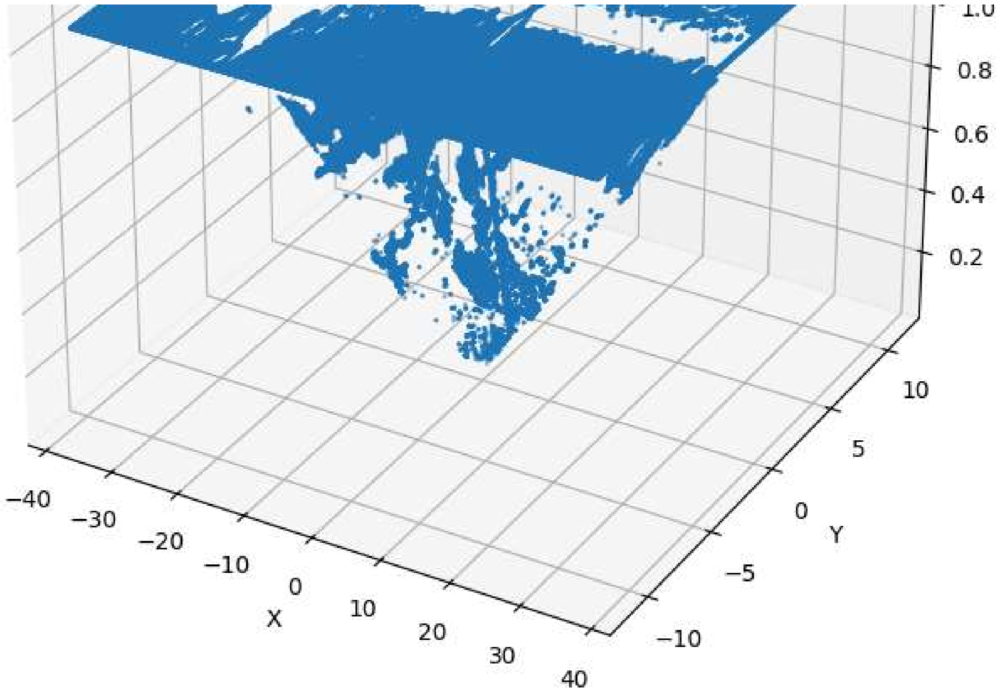






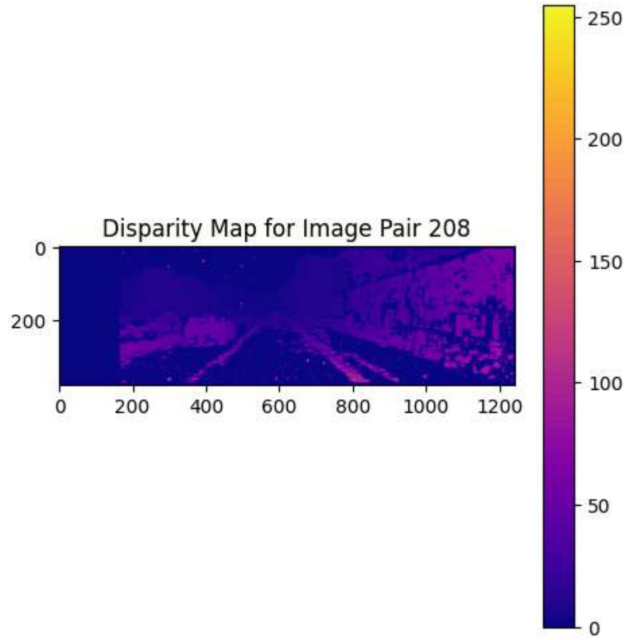
3D Reconstruction for Image Pair 86

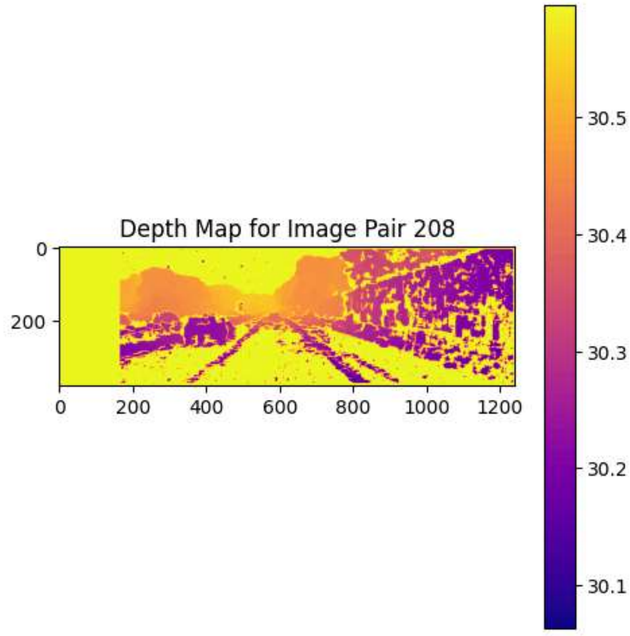


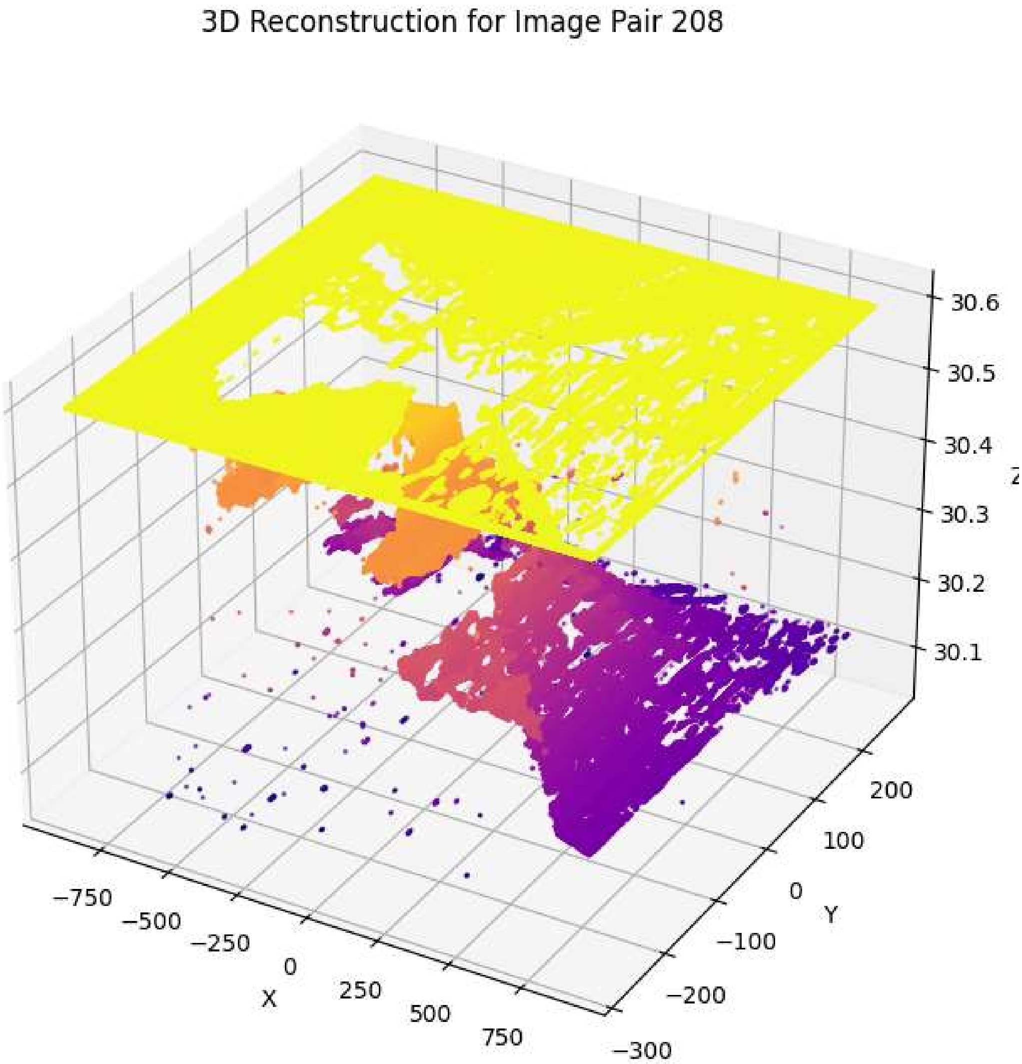

1.0
0.8
0.6
0.4
0.2
−40
−30
−20
−10
0
10
20
30
40
X
10
5
0
Y
−5
−10

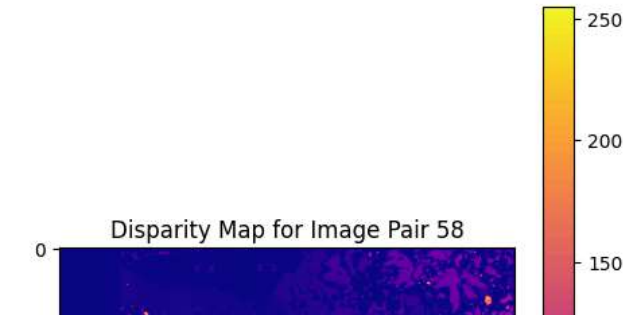
```
import cv2
import numpy as np
import os
import matplotlib.pyplot as plt
import random

# Paths to your stereo image directories
image_2_path = '/content/Image 2'  # Left images
image_3_path = '/content/Image 3'  # Right images

# Function to load images from a folder
def load_images_from_folder(folder):
    images = []
    for filename in sorted(os.listdir(folder)):
        img_path = os.path.join(folder, filename)
        img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
        if img is not None:
            images.append(img)
    return images

# Load left and right images
left_images = load_images_from_folder(image_2_path)
right_images = load_images_from_folder(image_3_path)

# Stereo block matching parameters
block_size = 15
min_disp = 0
num_disp = 160  # Number of disparities (increase based on your data)

# Focal length and baseline (for depth calculation)
focal_length = 21  # Example value (in mm, adjust based on your camera setup)
baseline = 0.54  # Example value (in meters, adjust based on your camera setup)

# Create stereo block matcher
stereo = cv2.StereoBM_create(numDisparities=num_disp, blockSize=block_size)

# Randomly select 10 pairs of images
num_images_to_display = 10
indices = random.sample(range(len(left_images)), num_images_to_display)

# For each randomly selected pair, compute disparity map, depth map and 3D reconstruction
for idx in indices:
    left_img = left_images[idx]
    right_img = right_images[idx]

    # Compute disparity map
    disparity = (stereo.compute(left_img, right_img).astype(np.float32) / 16.0) + 20

    # Normalize disparity for better visualization
    disparity_normalized = cv2.normalize(disparity, None, 0, 255, cv2.NORM_MINMAX)
    disparity_normalized = np.uint8(disparity_normalized)

    # Calculate depth map (D = f * B / d)
    # Convert disparity to depth using the formula
    depth_map = (focal_length * baseline) / (disparity + 1e-5)+30  # Adding small epsilon to avoid division

    # Create a colormap for the depth map for consistent color visualization
    cmap = plt.get_cmap('plasma')
    norm_depth_map = cv2.normalize(depth_map, None, 0, 1, cv2.NORM_MINMAX)
    depth_map_colored = cmap(norm_depth_map)  # Apply colormap

    # Create a 3D point cloud from the depth map
    height, width = depth_map.shape
    points_3D = []
    colors = []
```

```
    for y in range(height):
        for x in range(width):
            # Get depth value at this pixel
            Z = depth_map[y, x]
            if Z > 0:  # Only consider valid depth values
                # Reconstruct the 3D coordinates using the camera intrinsic parameters
                X = (x - width / 2) * Z / focal_length
                Y = (y - height / 2) * Z / focal_length
                points_3D.append([X, Y, Z])
                # Use the corresponding color from the depth map for each point
                colors.append(depth_map_colored[y, x, :3])  # Exclude alpha channel if exists

    # Convert points and colors to numpy arrays for visualization
    points_3D = np.array(points_3D)
    colors = np.array(colors)

    # Display the disparity and depth maps
    plt.figure(figsize=(12, 6))

    # Disparity Map Visualization
    plt.subplot(1, 2, 1)
    plt.imshow(disparity_normalized, cmap='plasma')
    plt.title(f'Disparity Map for Image Pair {idx + 1}')
    plt.colorbar()

    # Depth Map Visualization
    plt.subplot(1, 2, 2)
    plt.imshow(depth_map, cmap='plasma')
    plt.title(f'Depth Map for Image Pair {idx + 1}')
    plt.colorbar()

    plt.show()

    # Plot 3D point cloud using matplotlib
    fig = plt.figure(figsize=(10, 8))
    ax = fig.add_subplot(111, projection='3d')

    # Plot points in 3D space with the color from depth map
    ax.scatter(points_3D[:, 0], points_3D[:, 1], points_3D[:, 2], c=colors, s=1)
    ax.set_title(f'3D Reconstruction for Image Pair {idx + 1}')
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('Z')

    plt.show()

    # Optional: You can save the 3D point cloud to a file, like a .txt or .pcd for later use
    # For now, the point cloud is stored in `points_3D`
    # np.savetxt(f"point_cloud_{idx+1}.txt", points_3D)
```

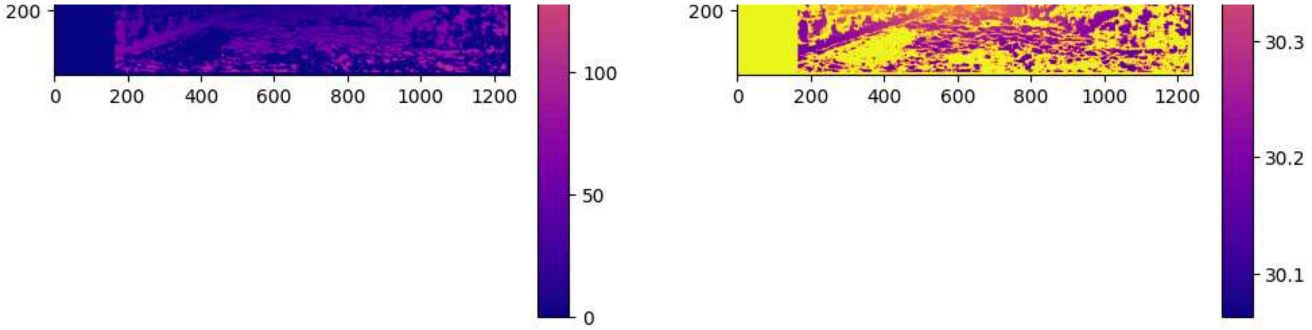

Disparity Map for Image Pair 208
0
200
0 200 400 600 800 1000 1200
250
200
150
100
50
0


Depth Map for Image Pair 208
0
200
0 200 400 600 800 1000 1200
30.5
30.4
30.3
30.2
30.1

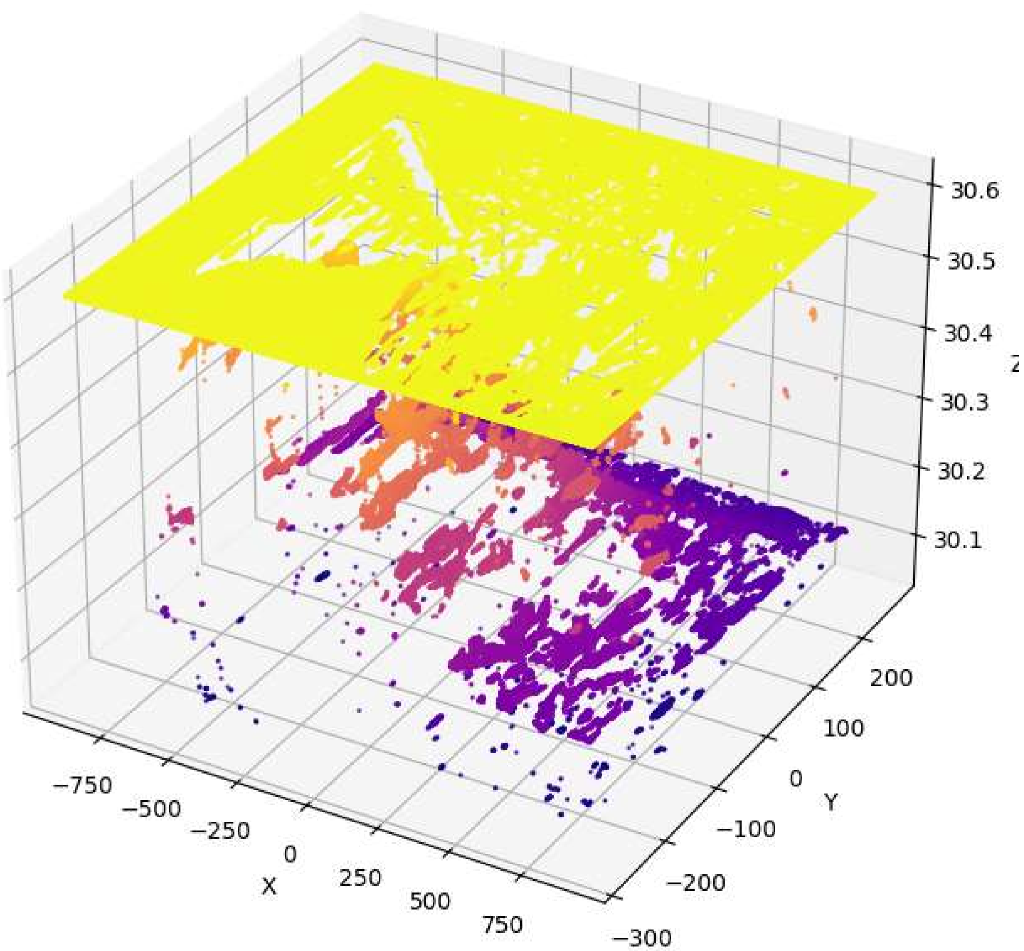

3D Reconstruction for Image Pair 208
30.6
30.5
30.4
30.3
30.2
30.1
Z
−750 −500 −250 0 250 500 750
X
200 100 0 −100 −200 −300
Y

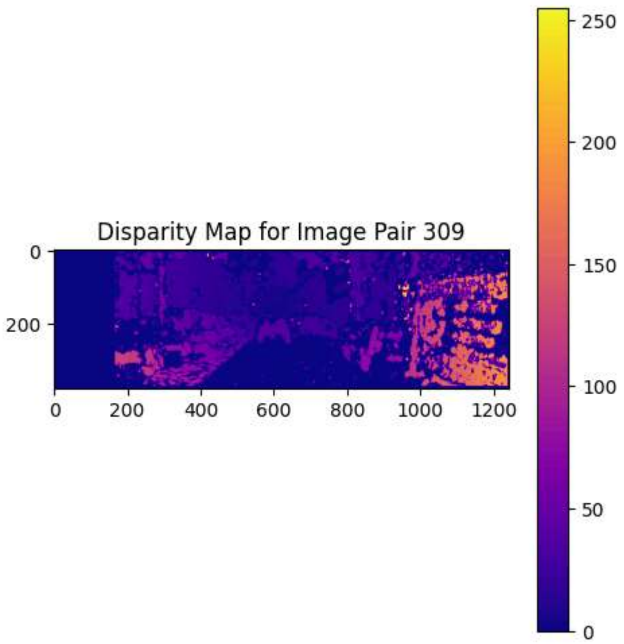

Disparity Map for Image Pair 58
0
250
200
150

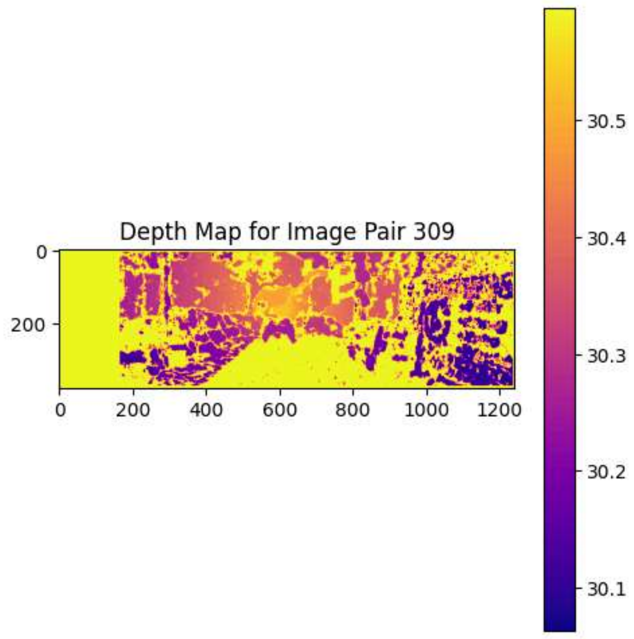

Depth Map for Image Pair 58
0
30.5
30.4



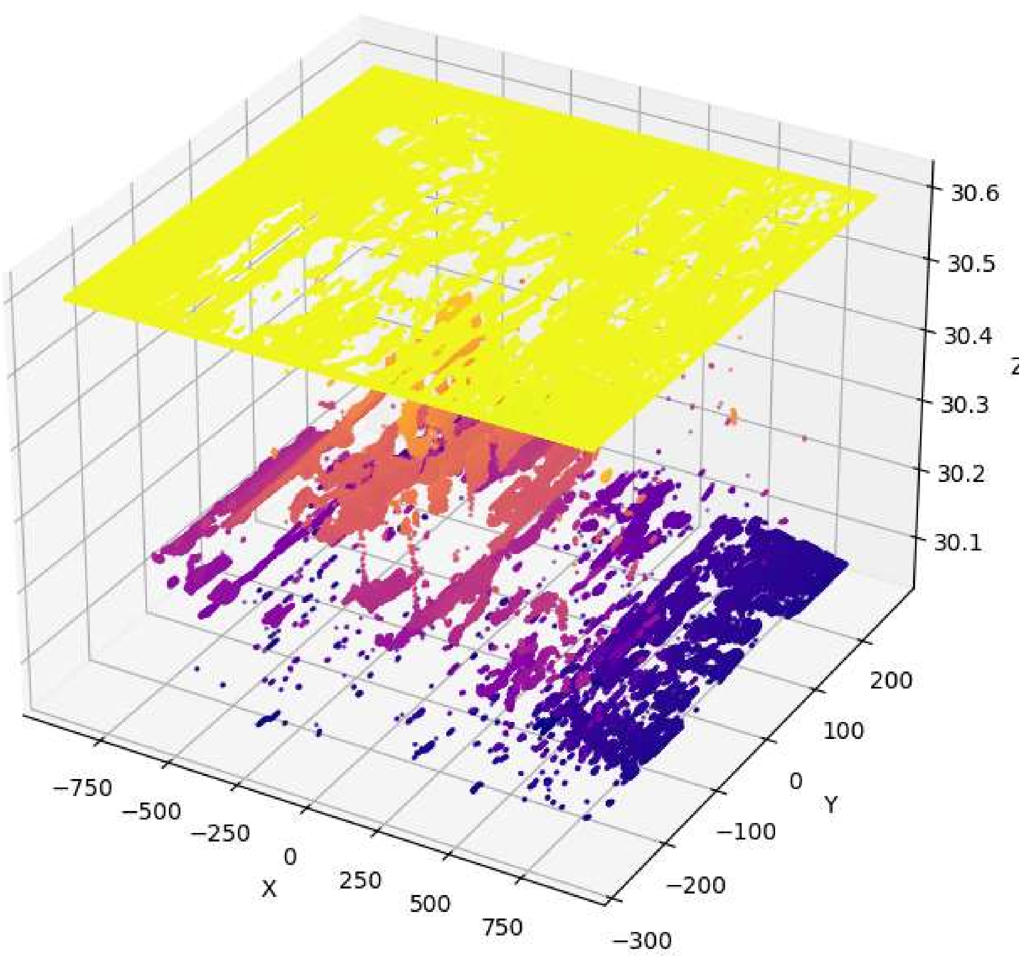
3D Reconstruction for Image Pair 58



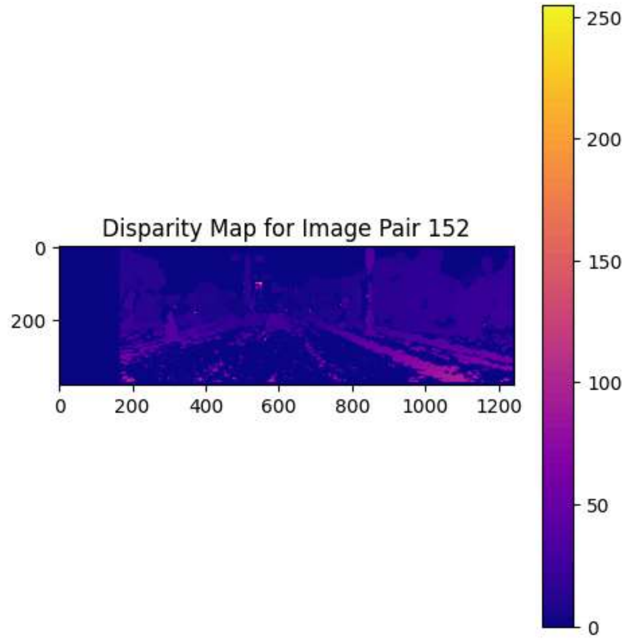




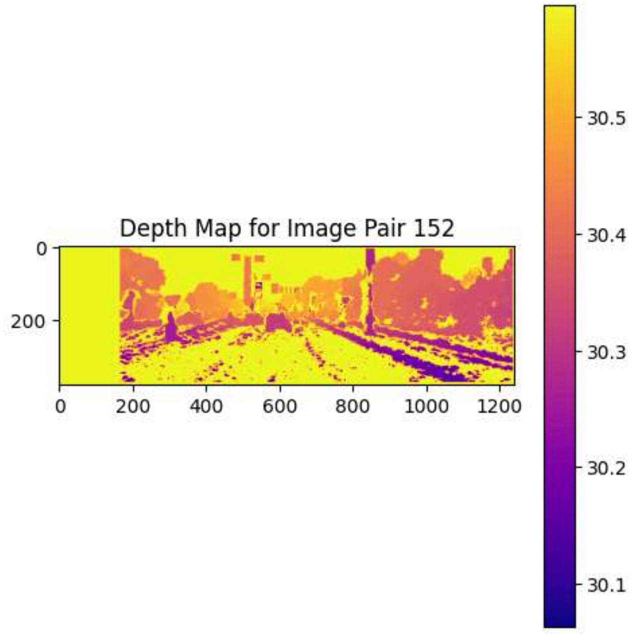
3D Reconstruction for Image Pair 309



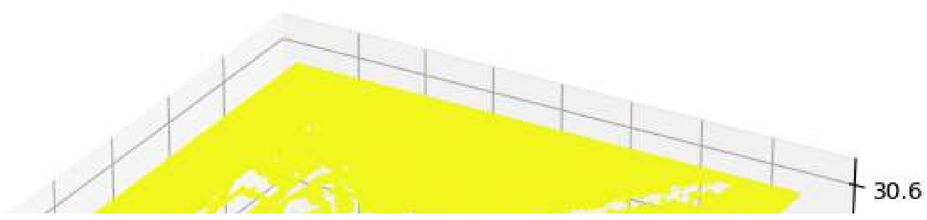
Disparity Map for Image Pair 152

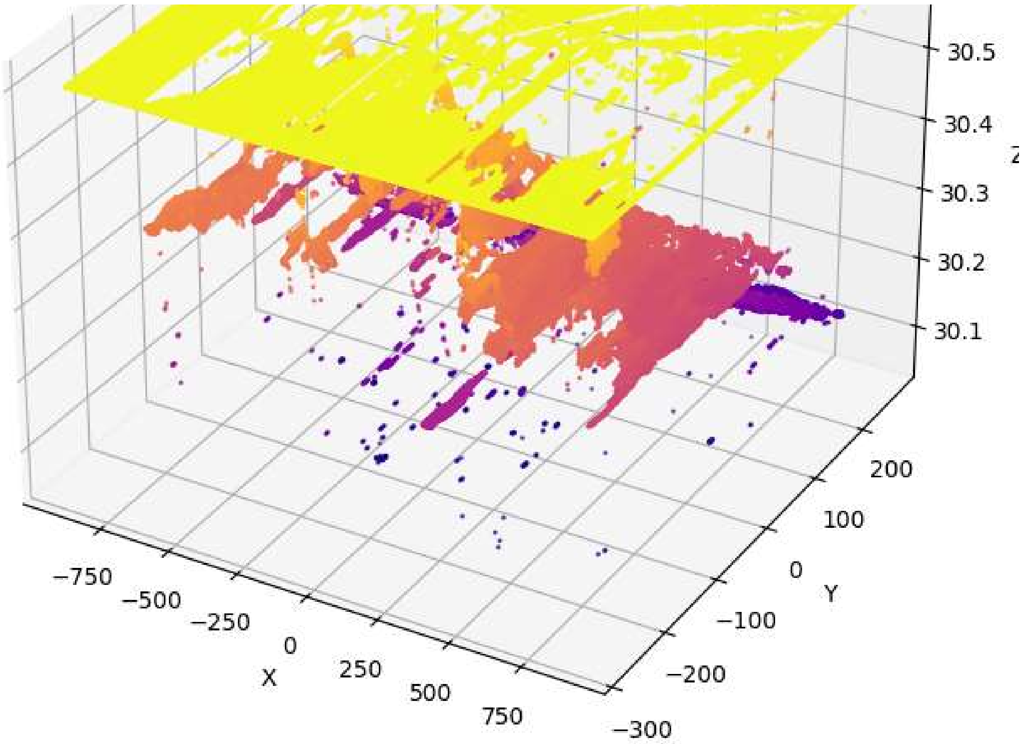


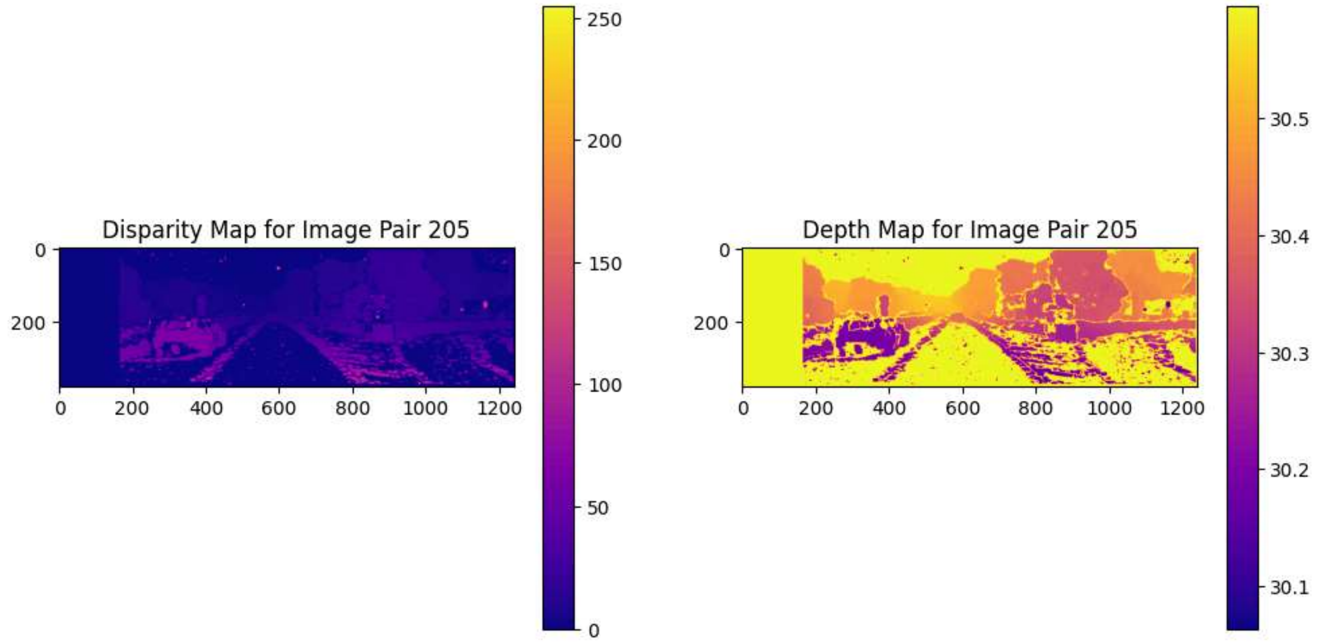
Depth Map for Image Pair 152



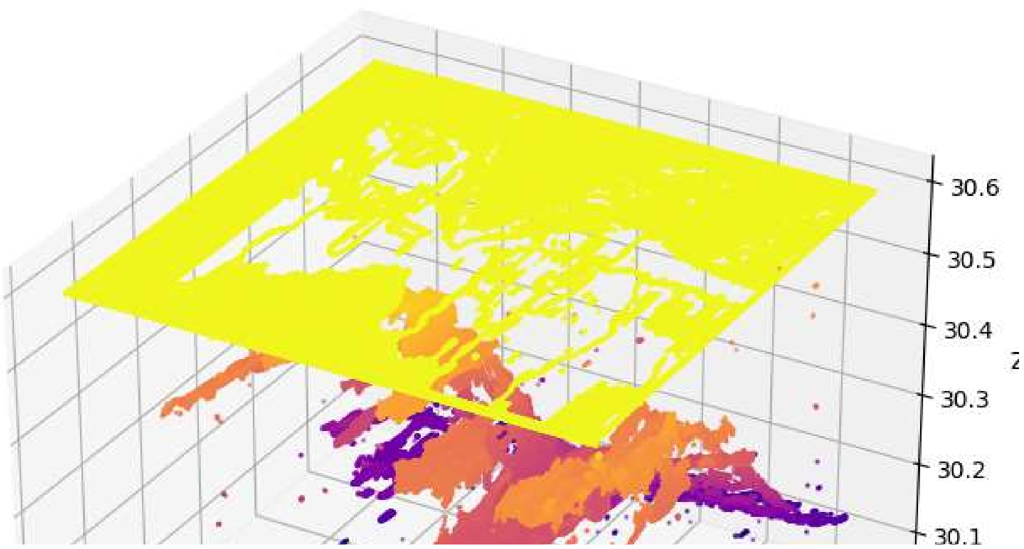
3D Reconstruction for Image Pair 152

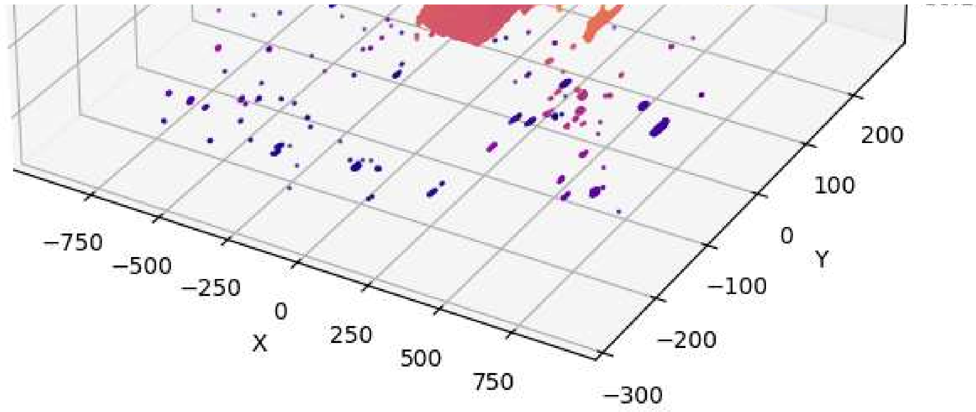


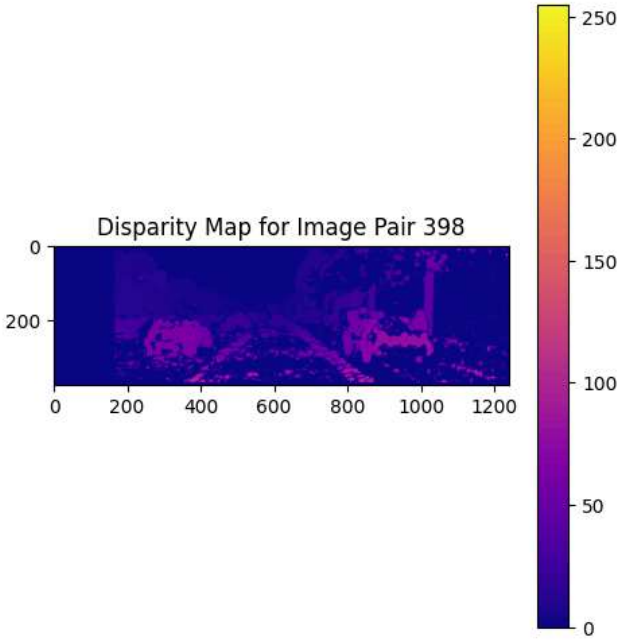


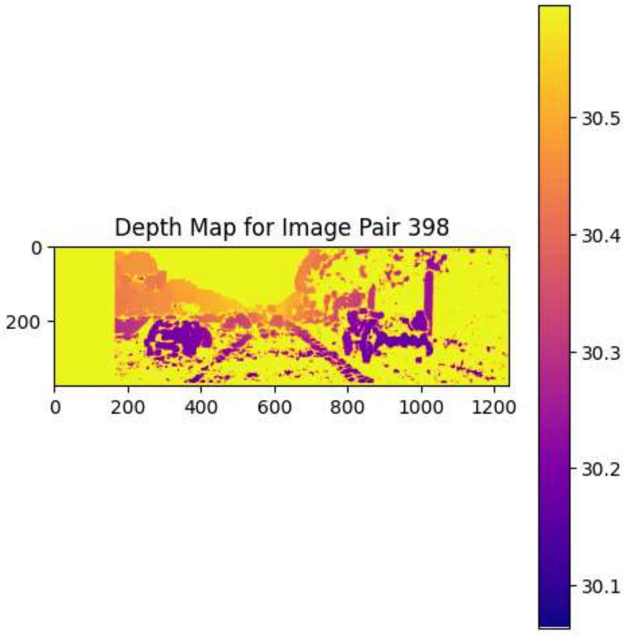


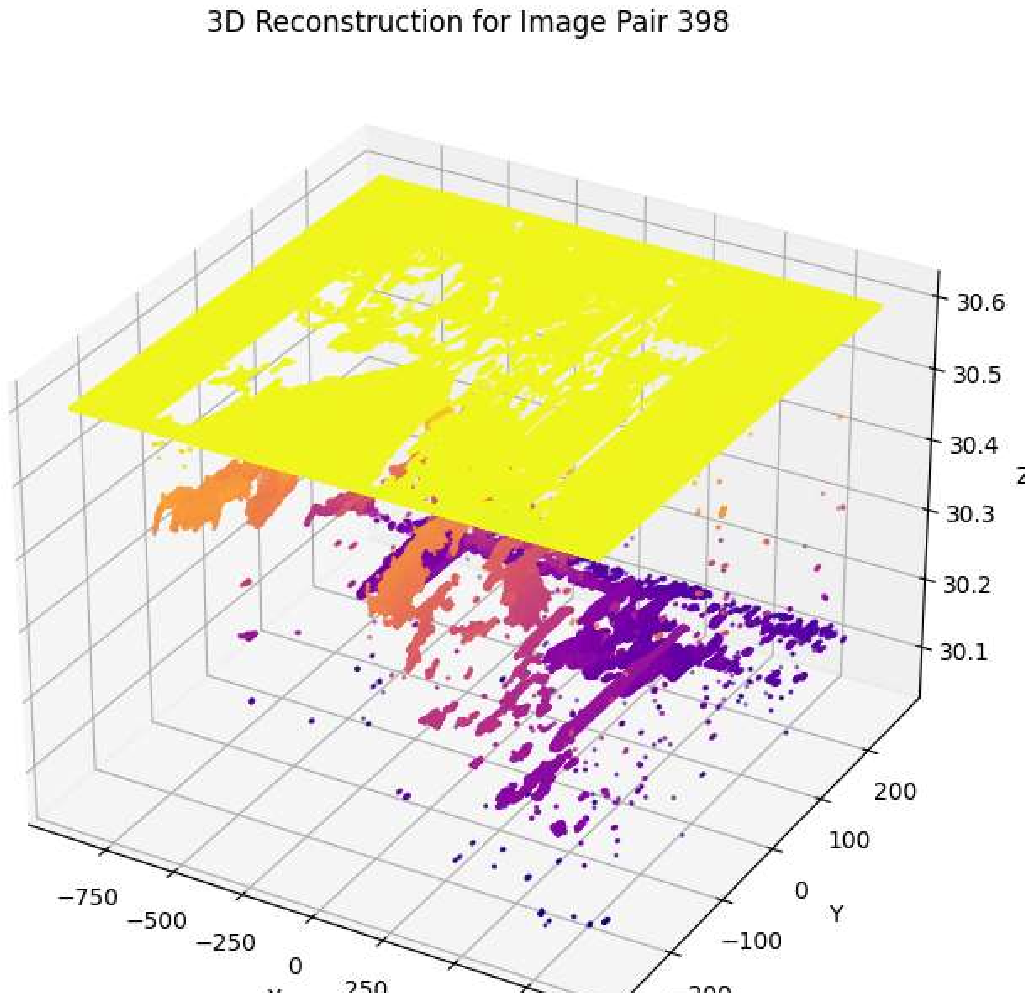
3D Reconstruction for Image Pair 205




−750
−500
−250
0
X
250
500
750
200
100
0
Y
−100
−200
−300


Disparity Map for Image Pair 398
0
200
0
200
400
600
800
1000
1200
250
200
150
100
50
0


Depth Map for Image Pair 398
0
200
0
200
400
600
800
1000
1200
30.5
30.4
30.3
30.2
30.1


3D Reconstruction for Image Pair 398
30.6
30.5
30.4
30.3
30.2
30.1
200
100
0
Y
−100
−750
−500
−250
0