Mohamed Aslam K
21BAD051

EXERCISE-9

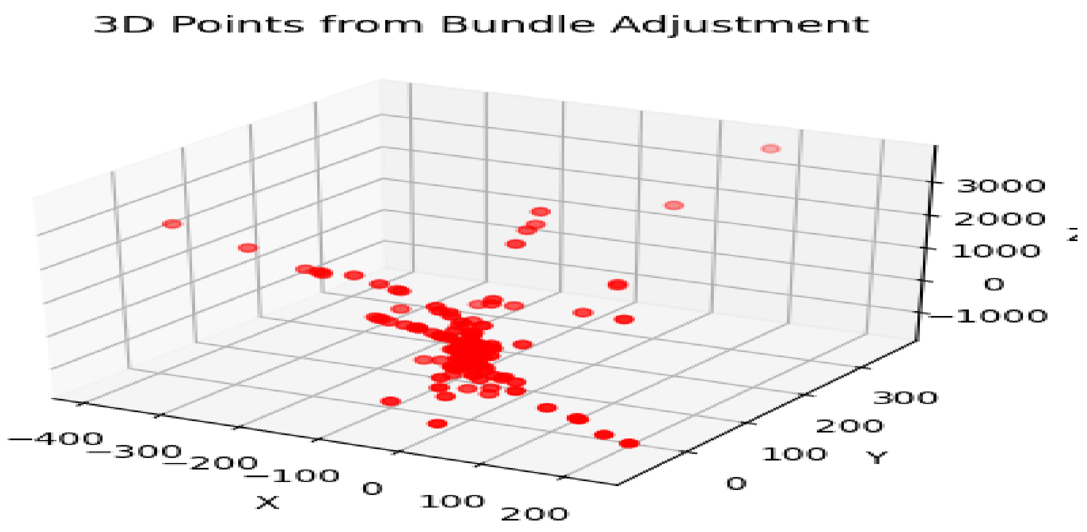
COMPUTER VISION

BUNDLE ADJUSTMENT:

```
import numpy as np

import cv2

import matplotlib.pyplot as plt

img1 = cv2.imread('disparity-1.png')

img2 = cv2.imread('disparity-2.png')

orb = cv2.ORB_create()

kp1, des1 = orb.detectAndCompute(img1, None)

kp2, des2 = orb.detectAndCompute(img2, None)

bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)

matches = bf.match(des1, des2)

matches = sorted(matches, key=lambda x: x.distance)

points1 = np.float32([kp1[m.queryIdx].pt for m in 
matches])

points2 = np.float32([kp2[m.trainIdx].pt for m in 
matches])

focal_length = 800  # Example focal length

center = (img1.shape[1] / 2, img1.shape[0] / 2)

K = np.array([[focal_length, 0, center[0]],

              [0, focal_length, center[1]],

              [0, 0, 1]])

E, mask = cv2.findEssentialMat(points1, points2, K, 
method=cv2.RANSAC, prob=0.999, threshold=1.0)

points1, points2 = points1[mask.ravel() == 1], 
points2[mask.ravel() == 1]
```

Mohamed Aslam K
21BAD051

```
_, R, t, mask_pose = cv2.recoverPose(E, points1, points2, K)


P1 = K @ np.hstack((np.eye(3), np.zeros((3, 1))))  # Camera 1
P2 = K @ np.hstack((R, t))
points_4d_hom = cv2.triangulatePoints(P1, P2, points1.T, points2.T)
points_3d = points_4d_hom[:3] / points_4d_hom[3]
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(points_3d[0], points_3d[1], points_3d[2], c='r', marker='o')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
plt.title('3D Points from Bundle Adjustment')
plt.show()
# Print optimized 3D points
print("3D Points:\n", points_3d.T)
```



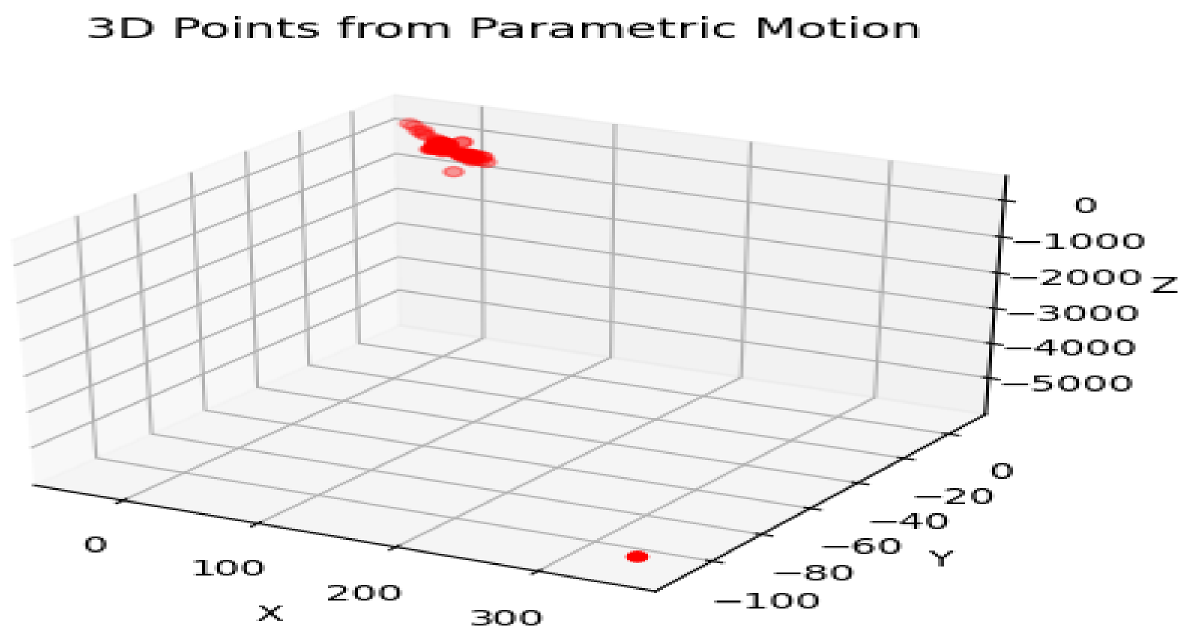
Mohamed Aslam K
21BAD051

PARAMETRIC:

```
import numpy as np

import cv2

import matplotlib.pyplot as plt

img1 = cv2.imread('disparity-1.png')

img2 = cv2.imread('disparity-2.png')

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)

plt.imshow(cv2.cvtColor(img1, cv2.COLOR_BGR2RGB))

plt.title('Image 1')

plt.axis('off')

plt.subplot(1, 2, 2)

plt.imshow(cv2.cvtColor(img2, cv2.COLOR_BGR2RGB))

plt.title('Image 2')

plt.axis('off')

plt.show()

orb = cv2.ORB_create()

kp1, des1 = orb.detectAndCompute(img1, None)

kp2, des2 = orb.detectAndCompute(img2, None)

bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)

matches = bf.match(des1, des2)

matches = sorted(matches, key=lambda x: x.distance)

points1 = np.float32([kp1[m.queryIdx].pt for m in
matches])

points2 = np.float32([kp2[m.trainIdx].pt for m in
matches])
```

```
F, mask = cv2.findFundamentalMat(points1, points2,
method=cv2.FM_RANSAC)

points1 = points1[mask.ravel() == 1]

points2 = points2[mask.ravel() == 1]

focal_length = 800

center = (img1.shape[1] / 2, img1.shape[0] / 2)

K = np.array([[focal_length, 0, center[0]],

              [0, focal_length, center[1]],

              [0, 0, 1]])

t = np.array([0, 0, 0])

R = np.eye(3)

P1 = K @ np.hstack((R, t.reshape(-1, 1)))

t1 = np.array([0, 0, 0])

t2 = np.array([0, 0, 5])

P2 = K @ np.hstack((R, t2.reshape(-1, 1)))

points_4d_hom = cv2.triangulatePoints(P1, P2,
points1.T, points2.T)

points_3d = points_4d_hom[:3] / points_4d_hom[3] fig =
plt.figure()

ax = fig.add_subplot(111, projection='3d')

ax.scatter(points_3d[0], points_3d[1], points_3d[2],
c='r', marker='o')

ax.set_xlabel('X')

ax.set_ylabel('Y')

ax.set_zlabel('Z')

plt.title('3D Points from Parametric Motion')

plt.show()
```

```
print("3D Points:\n", points_3d.T)
```

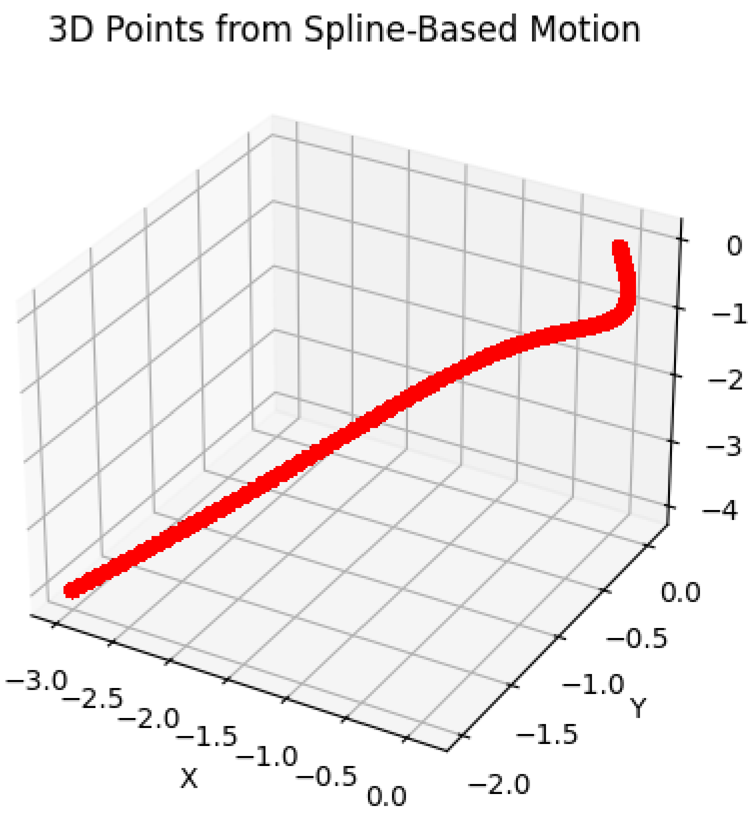


SPLINE:

```
import numpy as np

import cv2

import matplotlib.pyplot as plt

from scipy.interpolate import CubicSpline

img1 = cv2.imread('disparity-1.png')

img2 = cv2.imread('disparity-2.png')

orb = cv2.ORB_create()

kp1, des1 = orb.detectAndCompute(img1, None)

kp2, des2 = orb.detectAndCompute(img2, None)

bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)

matches = bf.match(des1, des2)

matches = sorted(matches, key=lambda x: x.distance)

points1 = np.float32([kp1[m.queryIdx].pt for m in
matches])

points2 = np.float32([kp2[m.trainIdx].pt for m in
matches])
```

```
F, mask = cv2.findFundamentalMat(points1, points2,
method=cv2.FM_RANSAC)

points1 = points1[mask.ravel() == 1]

points2 = points2[mask.ravel() == 1]

focal_length = 800

center = (img1.shape[1] / 2, img1.shape[0] / 2)

K = np.array([[focal_length, 0, center[0]],

              [0, focal_length, center[1]],

              [0, 0, 1]])

key_positions = np.array([

    [0, 0, 0],

    [0, 0, 1],

    [1, 0, 2],

    [2, 1, 3],

    [3, 2, 4]

])

t = np.arange(len(key_positions))

spline = CubicSpline(t, key_positions,
bc_type='natural')

t_spline = np.linspace(0, len(key_positions) - 1, 100)

spline_positions = spline(t_spline)

points_4d_list = []

for pos in spline_positions:

    R = np.eye(3)

    P = K @ np.hstack((R, pos.reshape(-1, 1)))

    points_4d_hom = cv2.triangulatePoints(P, P,
points1.T, points2.T)

    points_4d_list.append(points_4d_hom)
```

```
points_3d = np.hstack([p[:3] / p[3] for p in
points_4d_list])

fig = plt.figure()

ax = fig.add_subplot(111, projection='3d')

ax.scatter(points_3d[0], points_3d[1], points_3d[2],
c='r', marker='o')

ax.set_xlabel('X')

ax.set_ylabel('Y')

ax.set_zlabel('Z')

plt.title('3D Points from Spline-Based Motion')

plt.show()

print("3D Points:\n", points_3d.T)
```


3D Points from Spline-Based Motion
Y
X