Question 2

1. from scipy.optimize import minimize

2. from scipy import linalg

3. import matplotlib.pyplot as plt

4. import tikzplotlib

5. import numpy as np

6.

7. # np.random.seed(0)

8. N = 100

9. half\_n = N//2

10. r = 10

11. x0\_gt, y0\_gt = 2,3 # center

12. s = r /16

13. # angle values in radian

14. t = np.random.uniform(0, 2\*np.pi, half\_n)

15. # noise values from 0 to s

16. n = s\*np.random.randn(half\_n)

17. # creating points on a noisy circle

18. x, y = x0\_gt + (r + n)\*np.cos(t), y0\_gt + (r + n)\*np.sin(t)

19. X\_circ = np.hstack((x.reshape(half\_n,1), y.reshape(half\_n,1)))

20.

21. s = 1

22. m, b = -1, 2

23. x = np.linspace(-12, 12, half\_n)

24. # creating points on a noisy line

25. y = m\*x + b + s\*np.random.randn(half\_n)

26. X\_line = np.hstack((x.reshape(half\_n, 1), y.reshape(half\_n, 1)))

27.

28. X = np.vstack((X\_circ, X\_line)) #All points

29.

30.

31. # Item 2: Total least squares line fitting with scicpy.optimize with RANSAC

32. import math

33. from scipy.optimize import minimize

34. N = X\_line.shape[0] # points

35. X\_ = X\_line

36.

37. def line\_equation\_from\_points(x1, y1, x2, y2):

38. # Calculate the direction vector (Δx, Δy)

39. delta\_x = x2 - x1

40. delta\_y = y2 - y1

41.

42. # Calculate the normalized vector (a, b)

43. magnitude = math.sqrt(delta\_x\*\*2 + delta\_y\*\*2)

44. a = delta\_y / magnitude

45. b = -delta\_x / magnitude

46.

47. # Calculate d

48. d = (a \* x1) + (b \* y1)

49.

50. # Return the line equation in the form ax + by = d

51. return a, b, d

52.

53. # RANSAC to fit a line

54. # returning the error

55. def line\_tls(x, indices):

56. a, b, d = x[0], x[1], x[2]

57. return np.sum(np.square(a\*X\_[indices,0] + b\*X\_[indices,1] - d))

58.

59.

60. # Constraint

61. # x = [a,b]

62. def g(x):

63. return x[0]\*\*2 + x[1]\*\*2 - 1

64.

65. # maintaining a\*\*2 + b\*\*2 = 1

66. cons = ({'type': 'eq', 'fun': g})

67.

68. # Computing the consensus (inliers)

69. def consensus\_line(X\_, x, t):

70. a, b, d = x[0], x[1], x[2]

71. error = np.absolute(a\*X\_[:,0] + b\*X\_[:,1] - d)

72. return error < t

73.

74.

75. t = 1.0 # Threshold value to determine data points that are fit well by model.

76. d = 0.4\*N # Number of close data points required to assert that a model fits well to data.

77. s = 2 # Minimum number of data points required to estimate model parameters.

78.

79. inliers\_line = [] # Indinces of the inliers

80. max\_iterations = 500

81. iteration = 0

82. best\_model\_line = [] # Best model normal (a, b) and distance from origin d

83. best\_error = np.inf

84. best\_sample\_line = [] # Three-point sample leading to the best model computation

85. res\_only\_with\_sample = [] # Result (a, b, d) only using the best sample

86. best\_inliers\_line = [] # Inliers of the model computed form the best sample

87.

88. while iteration < max\_iterations:

89. indices = np.random.randint(0, N, s) # A sample of 2 points selected at random

90. x0 = np.array([1, 1, 0]) # Initial estimate a,b,d

91.

92. #return many things but res.x contains a,b,d

93. res = minimize(fun = line\_tls, args = indices, x0 = x0, tol= 1e-7, constraints=cons, options={'disp': True})

94.

95. #returns indexes of cordinates of inliners

96. inliers\_line = consensus\_line(X\_, res.x, t) # Computing the inliers

97.

98. print('rex.x: ', res.x)

99. print('Iteration = ', iteration, '. No. inliners = ', inliers\_line.sum()) # number of inliners

100.

101. if inliers\_line.sum() > d:

102. x0 = res.x

103. # Computing the new model using the inliers

104. #return many things but res.x contains a,b,d

105. res = minimize(fun = line\_tls, args = inliers\_line, x0 = x0, tol= 1e-6, constraints=cons, options={'disp': True})

106.

107. print(res.x, res.fun) # what is fun

108. if res.fun < best\_error:

109. print('A better model found ... ', res.x, res.fun)

110. best\_model\_line = res.x

111. best\_eror = res.fun

112. best\_sample\_line = X\_[indices,:]

113. res\_only\_with\_sample = x0

114. best\_inliers\_line = inliers\_line

115.

116. iteration += 1

117.

118. print('Best line model', best\_model\_line)

119.

120. fig, ax = plt.subplots(1,1, figsize=(8,8))

121. ax.scatter(X\_line[:,0], X\_line[:,1], label='Line')

122. ax.scatter(X\_circ[:,0], X\_circ[:,1], label='Circle')

123. circle\_gt = plt.Circle((x0\_gt,y0\_gt), r, color='g', fill=False, label='Ground\_truth\_circle')

124. ax.add\_patch(circle\_gt) # add circle

125. ax.plot((x0\_gt), (y0\_gt), '+', color='g') # add centre of the circle

126. x\_min, x\_max = ax.get\_xlim()

127. x\_ = np.array([x\_min, x\_max])

128. y\_ = m\*x\_ + b

129. plt.plot(x\_, y\_, color='m', label='Ground\_truth\_line')

130.

131. ax.scatter(X\_line[:,0],X\_line[:,1], label='All points')

132. ax.scatter(X\_[best\_inliers\_line,0],X\_[best\_inliers\_line,1], color='y', label='Inliers')

133. ax.scatter(best\_sample\_line[:,0],best\_sample\_line[:,1], marker='x', color='r', label='Best sample')

134. x\_min, x\_max = ax.get\_xlim()

135. x\_ = np.array([x\_min, x\_max])

136. y\_ = (-best\_model\_line[0]\*x\_ + best\_model\_line[2])/best\_model\_line[1]

137. plt.plot(x\_, y\_, label='RANSAC line')

138. plt.legend()

139.

A graph of a circle with colored dots

Description automatically generated

Question 3

1. 1.  import cv2 as cv

2. import numpy as np

3. import matplotlib.pyplot as plt

4.

5. clicked\_coordinates = []

6. click\_count = 0

7.

8. # Mouse callback function

9. def mouse\_callback(event, x, y, flags, param):

10. global clicked\_coordinates, click\_count

11.

12. # If left mouse button is clicked, capture the coordinates and draw a marker

13. if event == cv.EVENT\_LBUTTONDOWN:

14. clicked\_coordinates.append((x, y))

15. click\_count += 1

16. cv.circle(image, (x, y), 5, (0, 0, 255), -1) # Draw a red circle as a marker

17. cv.imshow("Image", image)

18.

19. # If four clicks are captured, stop capturing

20. if click\_count == 4:

21. cv.destroyAllWindows()

22.

23.

24. image = cv.imread('images/collage.jpg')

25. cv.imshow("Image", image)

26.

27. cv.setMouseCallback("Image", mouse\_callback)

28.

29. cv.waitKey(0)

30.

31. # Output the coordinates

32. print("Clicked Coordinates:")

33. for i, (x, y) in enumerate(clicked\_coordinates, start=1):

34. print(f"Point {i}: ({x}, {y})")

35.

36. # display the image with markers in Matplotlib

37. image\_rgb = cv.cvtColor(image, cv.COLOR\_BGR2RGB)

38. # Set the plot size to 15x15 inches

39. fig, ax = plt.subplots(figsize=(5, 5))

40.

41. ax.imshow(image\_rgb)

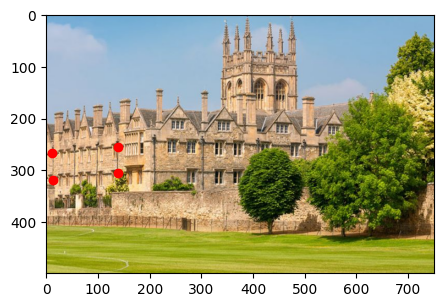
42. for (x, y) in clicked\_coordinates:

43. ax.plot(x, y, 'ro') # Draw a red marker

44.

45. plt.show()

46.



1. import matplotlib.pyplot as plt

2.

3. img\_building = cv.imread('images/collage.jpg')

4. img = cv.cvtColor(img\_building, cv.COLOR\_BGR2RGB)

5. img\_flag = cv.imread('images/flag2.jpg')

6.

7. points\_building = np.array(clicked\_coordinates, dtype=np.float32)

8.

9. # should be in cliking order

10. point\_flag = np.array([[0, 0], [img\_flag.shape[1], 0], [img\_flag.shape[1], img\_flag.shape[0]], [0, img\_flag.shape[0]]], dtype=np.float32)

11.

12. # Calculate the homography matrix

13. homography\_matrix, \_ = cv.findHomography(point\_flag, points\_building)

14.

15. # Warp the flag image

16. flag\_warped = cv.warpPerspective(img\_flag, homography\_matrix, (img\_building.shape[1], img\_building.shape[0]))

17. warped\_flag=cv.cvtColor(flag\_warped, cv.COLOR\_BGR2RGB)

18.

19. # Adjust transparency

20. alpha = 0.5

21.

22. # # Create the composite image

23. composite\_image = cv.addWeighted(img\_building, 1, flag\_warped, alpha, 0, dst=img\_building)

24. composite=cv.cvtColor(composite\_image, cv.COLOR\_BGR2RGB)

25.

26. fig, ax = plt.subplots(1, 3, figsize=(15,15))

27.

28. ax[0].imshow(img)

29. ax[0].axis('off')

30. ax[0].legend('Image of the building')

31. ax[1].imshow(warped\_flag)

32. ax[1].axis('off')

33. ax[1].legend('Image of the building')

34. ax[2].imshow(composite)

35. ax[2].axis('off')

36. ax[2].legend('Image of the building')

37. plt.show()

38.

A flag on a black background

Description automatically generated

Question 4