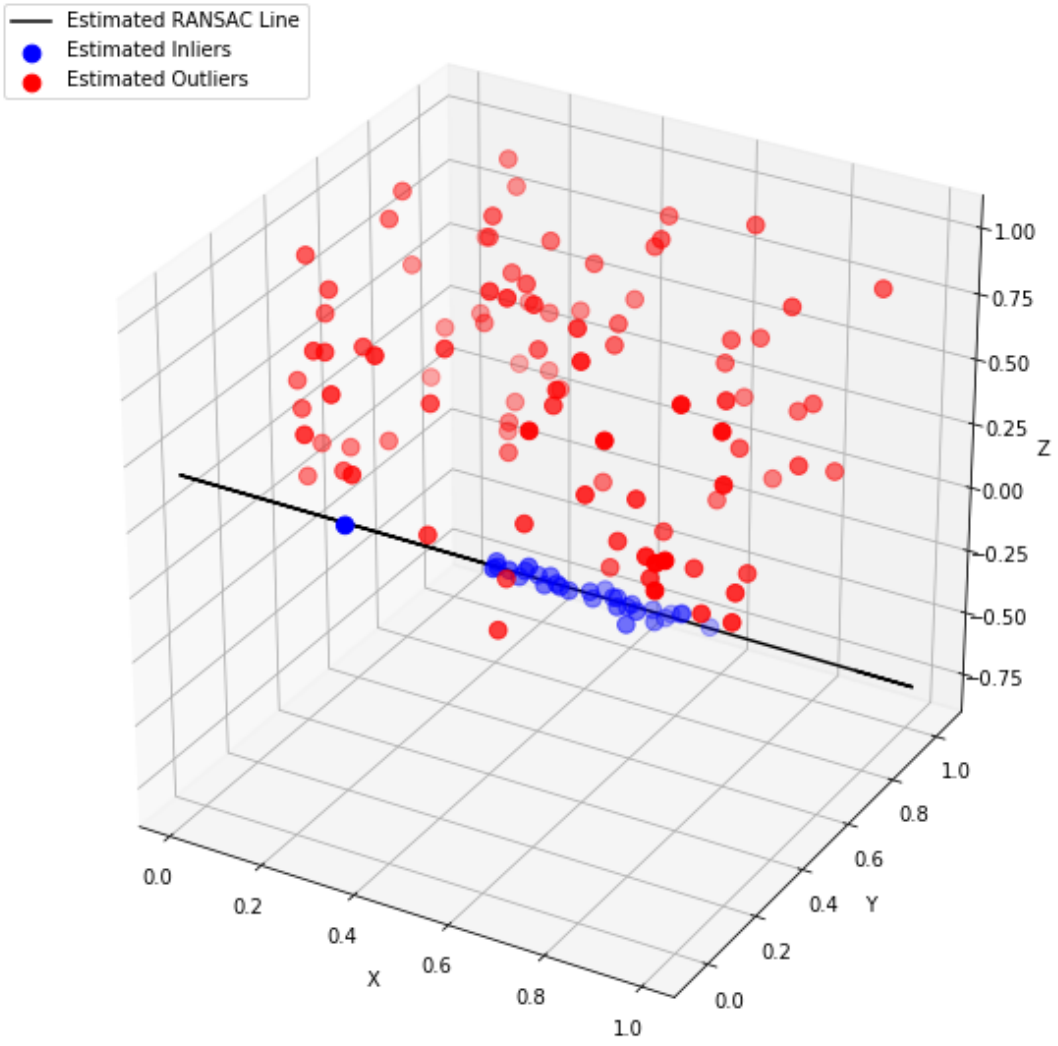
BIOS 7718-Assignment 2 Source Code

April 8, 2019

In [44]:
```
##### Problem 1.1 #####
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from skimage.measure import LineModelND, ransac

# generate inliers prior to adding Gaussian noise
x = np.arange(0.4, 0.70, 0.01).reshape(-1,1)
y = x
z = (1 - x - 2*y)/3
inl = np.concatenate((x, y, z), axis = 1)

# generate Gaussian noise
# set seed
np.random.seed(1212)
mu = (0, 0, 0)
cov = [[(0.03)**2,0,0], [0,(0.03)**2,0], [0,0,(0.03)**2]]
err = np.random.multivariate_normal(mu, cov, 30)

# add noise to inliers
inl_final = inl + err

# generate outliers
# set seed
np.random.seed(1212)
outl = []

for i in range(1,101):
    single_outl = np.random.uniform([0,0,0], [1,1,1])
    outl.append(single_outl)
outl_final = np.array(outl)

# combine all data
all_data = np.vstack((inl_final, outl_final))

# designate inliers and outliers
indices = np.arange(all_data.shape[0])
```

```
inlier_indices = indices[0:30]
outlier_indices = indices[30:]

# robustly fit line only using inlier data with RANSAC algorithm
# set seed
np.random.seed(1212)
model_ransac, inliers = ransac(all_data, LineModelND,
                               min_samples=2,
                               residual_threshold=0.08,
                               max_trials=500,
                               stop_probability = 0.99)
outliers = inliers == False

# 3D line coordinates
pred_line = model_ransac.predict(all_data[:,0])

# plot 3D line and data points fit using RANSAC
fig = plt.figure(figsize = (10,10))
ax = plt.axes(projection='3d')
ax.scatter(all_data[inliers][:, 0],
           all_data[inliers][:, 1],
           all_data[inliers][:, 2], c='b',
           marker='o', label='Estimated Inliers', s=75)
ax.scatter(all_data[outliers][:, 0],
           all_data[outliers][:, 1],
           all_data[outliers][:, 2], c='r',
           marker='o', label='Estimated Outliers', s=75)
ax.plot(pred_line[:,0], pred_line[:,1],
        pred_line[:,2], color = 'black',
        label='Estimated RANSAC Line')
ax.set_xlabel('X'), ax.set_ylabel('Y'), ax.set_zlabel('Z')
ax.legend(loc='upper left')
plt.show()
```



In [45]:
```
# number of true inliers detected
np.sum(inliers[0:30])
```

Out[45]: 30

In [46]:
```
# number of outliers classified as inliers
np.sum(inliers[30:])
```

Out[46]: 1

In [47]:
```
##### Problem 1.2 #####
# generate inliers prior to adding Gaussian noise
x_2 = np.arange(0.4, 0.60, 0.01).reshape(-1,1)
y_2 = x_2
```

```
z_2 = 1 - x_2 - y_2
inl_2 = np.concatenate((x_2, y_2, z_2), axis = 1)

# generate Gaussian noise
# set seed
np.random.seed(1212)
mu_2 = (0, 0, 0)
cov_2 = [[(0.01)**2,0,0], [0,(0.01)**2,0], [0,0,(0.01)**2]]
err_2 = np.random.multivariate_normal(mu_2, cov_2, 20)

# add noise to inliers
inl_2_final = inl_2 + err_2

# combine all data
all_data_2 = np.vstack((inl_2_final, all_data))

# fit first line
# set seed
np.random.seed(1212)
model_ransac_1, inliers_1 = ransac(all_data_2, LineModelND,
                                   min_samples=2,
                                   residual_threshold=0.02,
                                   max_trials=500,
                                   stop_probability = 0.99)
outliers_1 = inliers_1 == False

# 3D line coordinates
pred_line_1 = model_ransac_1.predict(all_data_2[:,0])
```

In [48]:
```
# number of inliers detected (1st It.)
np.sum(inliers_1)
```

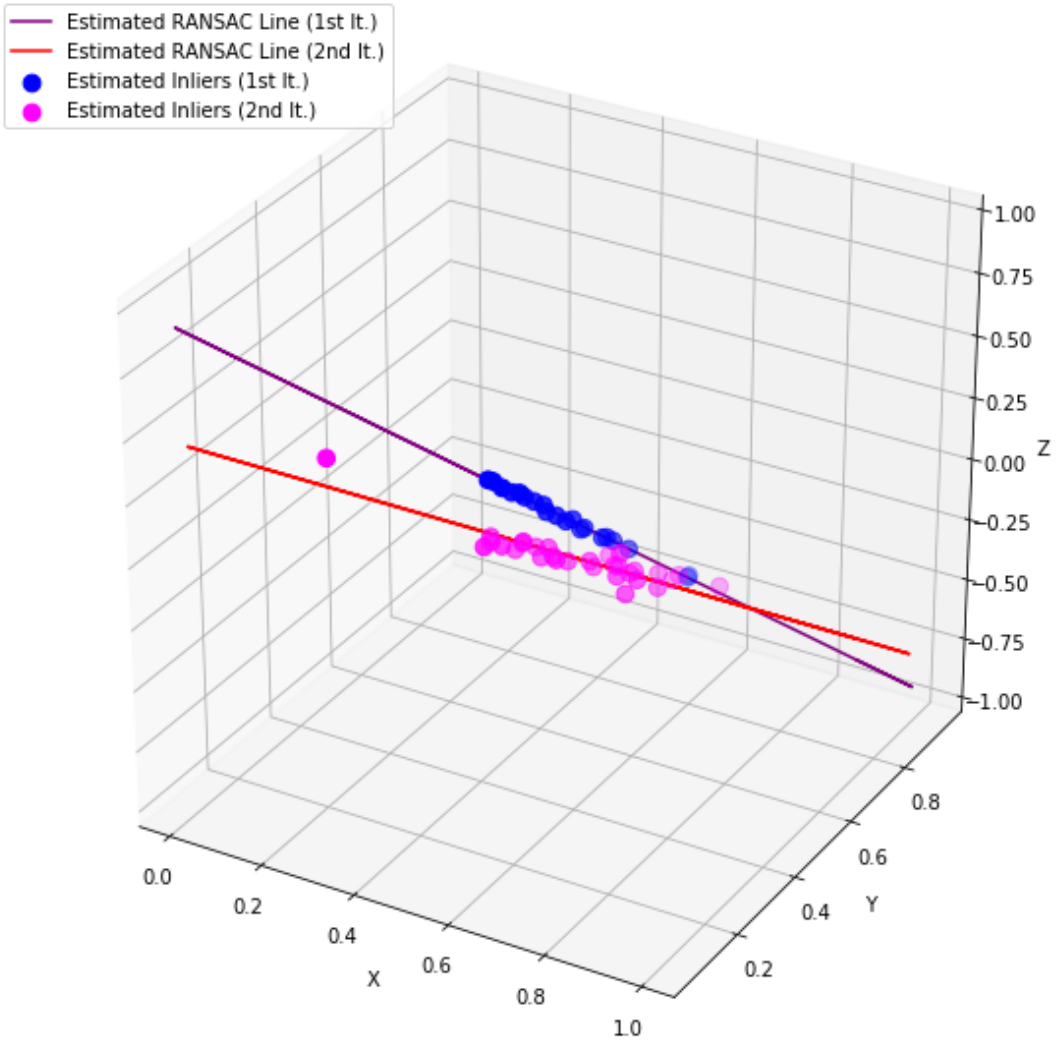
Out[48]: 20

In [49]:
```
# number of TRUE inliers detected (1st It.)
np.sum(inliers_1[0:20])
```

Out[49]: 18

In [50]:
```
# exlude inliers detected from second RANSAC fitting
all_data_2_exc = all_data_2[outliers_1]

# fit first line
# set seed
np.random.seed(1212)
model_ransac_2, inliers_2 = ransac(all_data_2_exc,
                                   LineModelND,
                                   min_samples=2,
                                   residual_threshold=0.08,
```

```
                                    max_trials=500,
                                    stop_probability = 0.99)
outliers_2 = inliers_2 == False

# 3D line coordinates
pred_line_2 = model_ransac_2.predict(all_data_2_exc[:,0])

# plot 3D line and data points fit using RANSAC
fig = plt.figure(figsize = (10,10))
ax = plt.axes(projection='3d')
ax.scatter(all_data_2[inliers_1][:, 0],
           all_data_2[inliers_1][:, 1],
           all_data_2[inliers_1][:, 2], c='b',
           marker='o',
           label='Estimated Inliers (1st It.)', s=75)
ax.plot(pred_line_1[:,0],
        pred_line_1[:,1],
        pred_line_1[:,2],
        color = 'purple',
        label='Estimated RANSAC Line (1st It.)')
ax.scatter(all_data_2_exc[inliers_2][:, 0],
           all_data_2_exc[inliers_2][:, 1],
           all_data_2_exc[inliers_2][:, 2],
           c='magenta',
           marker='o',
           label='Estimated Inliers (2nd It.)', s=75)
ax.plot(pred_line_2[:,0], pred_line_2[:,1],
        pred_line_2[:,2], color = 'r',
        label='Estimated RANSAC Line (2nd It.)')

ax.set_xlabel('X'), ax.set_ylabel('Y'), ax.set_zlabel('Z')
ax.legend(loc='upper left')
plt.show()
```




```
In [51]: # number of inliers detected (2nd It.)
         np.sum(inliers_2)
```

```
Out[51]: 30
```

```
In [52]: ##### Problem 2.1 #####
         import cv2

         # read image in
         car_img = cv2.imread('/Users/piper/Piper Documents/Biomedical Imaging/Assignments/Ass
         car_img_grey = cv2.imread('/Users/piper/Piper Documents/Biomedical Imaging/Assignments

         # smooth image with Gaussian filter
         car_img_smooth = cv2.GaussianBlur(car_img_grey, (19,19), 3)
```

```
        # Hough transform to detect circles
        circles = cv2.HoughCircles(car_img_smooth, cv2.HOUGH_GRADIENT, 1, 80,
                                   param1=100, param2=30, minRadius=0, maxRadius=0)
        circles = np.uint16(np.around(circles))

        for i in circles[0,:]:
            # draw the outer circle
            cv2.circle(car_img,(i[0],i[1]),i[2],(0,255,0),2)
            # draw the center of the circle
            cv2.circle(car_img,(i[0],i[1]),2,(255,0,0),5)

        cv2.imwrite('/Users/piper/Piper Documents/Biomedical Imaging/Assignments/Assignment 2/
```

Out[52]: True

```
In [53]: ##### Problem 2.2 #####
         # read image in
         nuclei_img = cv2.imread('/Users/piper/Piper Documents/Biomedical Imaging/Assignments/A
         nuclei_img_grey = cv2.cvtColor(nuclei_img, cv2.COLOR_BGR2GRAY)

         # smooth image with Gaussian filter
         nuclei_img_smooth = cv2.GaussianBlur(nuclei_img_grey, (31,31), 5)

         # Hough transform to detect circles
         circles = cv2.HoughCircles(nuclei_img_grey, cv2.HOUGH_GRADIENT, 1, 40,
                                    param1=150, param2=12, minRadius=0, maxRadius=30)

         circles = np.uint16(np.around(circles))

         for i in circles[0,:]:
             # draw the center of the circle
             cv2.circle(nuclei_img,(i[0],i[1]),2,(0,0,255),10)

         cv2.imwrite('/Users/piper/Piper Documents/Biomedical Imaging/Assignments/Assignment 2/
```

Out[53]: True

```
In [54]: # number of detected nuclei
         np.shape(circles)[1]
```

Out[54]: 44

```
In [55]: ##### Problem 3.1 #####
         coin_img = cv2.imread('/Users/piper/Piper Documents/Biomedical Imaging/Assignments/Ass
         coin_img_grey = cv2.imread('/Users/piper/Piper Documents/Biomedical Imaging/Assignment
         coin_img_grey_final = ~coin_img_grey

         # create kernel
```

```
kernel = np.ones((11,11), np.uint8)

# thresholding grey-scale image
ret, thresh_img = cv2.threshold(coin_img_grey_final, 0, 255, cv2.THRESH_BINARY + cv2.

# erode image to get foreground
erode_img = cv2.erode(thresh_img, kernel, iterations=2)

# dilate image to get background
dilate_img = cv2.dilate(thresh_img, kernel, iterations=2)

# subtract eroded and dilated images to get 'unknown' region
unknown = cv2.subtract(dilate_img, erode_img)

# label markers
ret, markers = cv2.connectedComponents(erode_img)

# add 1 to all labels so that sure background is 1
markers = markers+1

# mark unknown region with 0
markers[unknown==255] = 0

# apply watershed transform
markers = cv2.watershed(coin_img, markers)

# make each segment a different color
coin_img[markers == -1] = [0,0,0]
coin_img[markers == 1] = [211,211,211]
coin_img[markers == 2] = [128,0,0]
coin_img[markers == 3] = [220,20,60]
coin_img[markers == 4] = [255,0,0]
coin_img[markers == 5] = [255,99,71]
coin_img[markers == 6] = [255,69,0]
coin_img[markers == 7] = [255,140,0]
coin_img[markers == 8] = [255,215,0]
coin_img[markers == 9] = [218,165,32]
coin_img[markers == 10] = [0,100,0]
coin_img[markers == 11] = [0,255,0]
coin_img[markers == 12] = [50,205,50]
coin_img[markers == 13] = [0,250,154]
coin_img[markers == 14] = [46,139,87]
coin_img[markers == 15] = [32,178,170]
coin_img[markers == 16] = [47,79,79]
coin_img[markers == 17] = [0,255,255]
coin_img[markers == 18] = [0,191,255]
coin_img[markers == 19] = [135,206,250]
coin_img[markers == 20] = [0,0,128]
```

```
coin_img[markers == 21] = [0,0,255]
coin_img[markers == 22] = [138,43,226]
coin_img[markers == 23] = [75,0,130]

# save image
cv2.imwrite('/Users/piper/Piper Documents/Biomedical Imaging/Assignments/Assignment 2/
```

Out[55]:
```
True
```

In [56]:
```
np.unique(markers)
```

Out[56]:
```
array([-1,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23], dtype=int32)
```

In [57]:
```
##### Problem 3.2 #####
particle_img = cv2.imread('/Users/piper/Piper Documents/Biomedical Imaging/Assignments
particle_img_grey = cv2.imread('/Users/piper/Piper Documents/Biomedical Imaging/Assig
particle_img_grey_final = ~particle_img_grey

# thresholding
ret, thresh_img = cv2.threshold(particle_img_grey_final, 0, 255, cv2.THRESH_BINARY +

# noise removal using opening operation
kernel = np.ones((11,11), np.uint8)
open_img = cv2.morphologyEx(thresh_img, cv2.MORPH_OPEN, kernel, iterations = 1)

# sure background area
sure_bg = cv2.dilate(open_img, kernel, iterations = 1)

# sure foreground area
dist_transform = cv2.distanceTransform(open_img, cv2.DIST_L2, 5)
ret, sure_fg = cv2.threshold(dist_transform, 0.15*dist_transform.max(), 255, 0)

# unknown region
sure_fg = np.uint8(sure_fg)
unknown = cv2.subtract(sure_bg, sure_fg)

# label markers
ret, markers = cv2.connectedComponents(sure_fg)

# add 1 to all labels so that sure background is 1
markers = markers + 1

# mark unknown region with 0
markers[unknown==255] = 0

# apply watershed transform
markers = cv2.watershed(particle_img, markers)
```

```
        # make each segment a different color
        particle_img[markers == -1] = [0,0,0]
        particle_img[markers == 1] = [211,211,211]
        particle_img[markers == 2] = [128,0,0]
        particle_img[markers == 3] = [220,20,60]
        particle_img[markers == 4] = [255,0,0]
        particle_img[markers == 5] = [255,99,71]
        particle_img[markers == 6] = [255,69,0]
        particle_img[markers == 7] = [255,140,0]
        particle_img[markers == 8] = [255,215,0]
        particle_img[markers == 9] = [218,165,32]
        particle_img[markers == 10] = [0,100,0]
        particle_img[markers == 11] = [0,255,0]
        particle_img[markers == 12] = [50,205,50]
        particle_img[markers == 13] = [0,250,154]
        particle_img[markers == 14] = [46,139,87]
        particle_img[markers == 15] = [32,178,170]
        particle_img[markers == 16] = [47,79,79]
        particle_img[markers == 17] = [0,255,255]
        particle_img[markers == 18] = [0,191,255]
        particle_img[markers == 19] = [135,206,250]
        particle_img[markers == 20] = [0,0,128]
        particle_img[markers == 21] = [0,0,255]
        particle_img[markers == 22] = [138,43,226]
        particle_img[markers == 23] = [75,0,130]
        particle_img[markers == 24] = [147,112,219]
        particle_img[markers == 25] = [128,0,128]
        particle_img[markers == 26] = [255,0,255]
        particle_img[markers == 27] = [199,21,133]
        particle_img[markers == 28] = [255,105,180]
        particle_img[markers == 29] = [219,112,147]
        particle_img[markers == 30] = [255,192,203]

        # save image
        cv2.imwrite('/Users/piper/Piper Documents/Biomedical Imaging/Assignments/Assignment 2/
```

Out[57]:
```
True
```

In [58]:
```
np.unique(markers)
```

Out[58]:
```
array([-1,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30],
      dtype=int32)
```

In [59]:
```
# generate overlay images for Problem 3
img_1 = cv2.imread('/Users/piper/Piper Documents/Biomedical Imaging/Assignments/Assig
img_2 = cv2.imread('/Users/piper/Piper Documents/Biomedical Imaging/Assignments/Assig
w1 = cv2.imread('/Users/piper/Piper Documents/Biomedical Imaging/Assignments/Assignme
w2 = cv2.imread('/Users/piper/Piper Documents/Biomedical Imaging/Assignments/Assignme
```

```
combine_1 = cv2.addWeighted(img_1, 0.6, w1, 0.4, 0)
combine_2 = cv2.addWeighted(img_2, 0.6, w2, 0.4, 0)

# save images
cv2.imwrite('/Users/piper/Piper Documents/Biomedical Imaging/Assignments/Assignment 2/
cv2.imwrite('/Users/piper/Piper Documents/Biomedical Imaging/Assignments/Assignment 2/
```

```
Out[59]: True
```