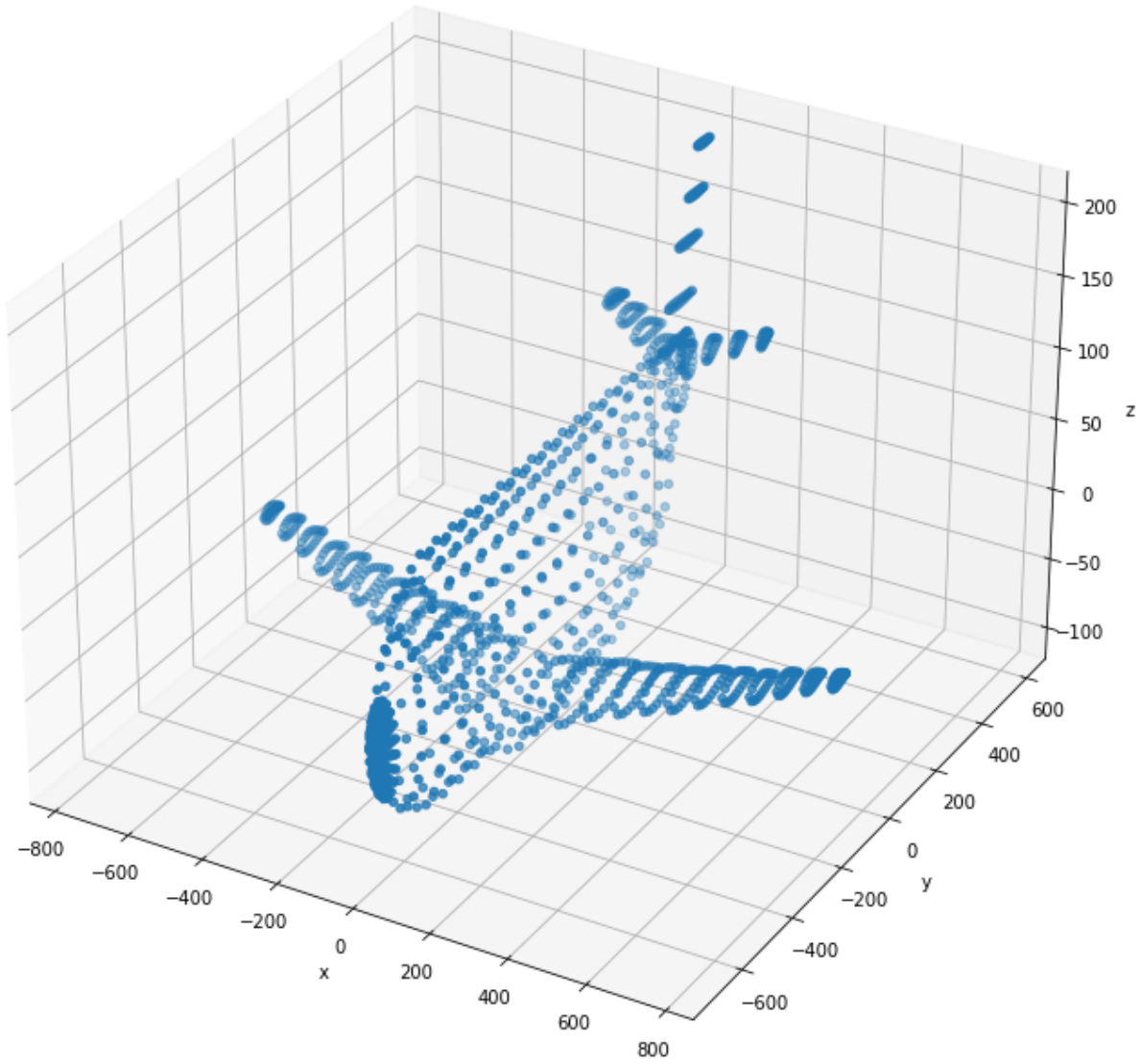
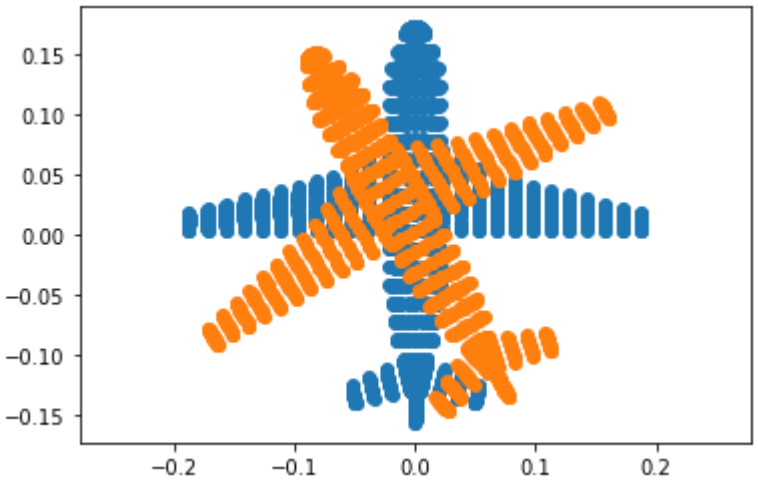
In [ ]:
```
import numpy as np
from plyfile import PlyData, PlyElement
import matplotlib.pyplot as plt

pcd = PlyData.read('Images/airplane.ply')
assert pcd is not None

points = np.concatenate((pcd['vertex']['x'].reshape(1,-1),pcd['vertex']['y'].reshape(1,
points = points - np.mean(points, axis = 1).reshape(3,1)
```

In [ ]:
```
fig = plt.figure(figsize=(12,12))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(points[0,:], points[1,:],points[2,:])
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
```

Out[ ]: Text(0.5, 0, 'z')



In [ ]:
```
ones=np.ones((1, points.shape[1]))
x = np.concatenate((points,ones),axis=0)

R = np.array([[1,0,0],[0,1,0],[0,0,1]])
K = np.array([[1,0,0],[0,1,0],[0,0,1]])
t = np.array([[0],[0],[-4000]])
P1 = K @ np.concatenate((R,t),axis =1)

R = np.array([[np.cos(np.pi / 6),-np.sin(np.pi /6),0],
              [np.sin(np.pi /6),np.cos(np.pi / 6),0],
              [0,0,1]])

K2 = np.array([[2,0,0],
              [0,2,0],
              [0,0,2]])# scaling
t = np.array([[0],[0],[-4000]])
P2 = K2 @ np.concatenate((R,t),axis =1)

x1 = P1 @ x
x2 = P2 @ x

x1 = x1 / x1[2,:]
x2 = x2 / x2[2,:]

fig, ax = plt.subplots(1,1,sharex=True , sharey=True)
ax.scatter(x1[0, :], x1[1,:])
ax.scatter(x2[0, :], x2[1,:])
ax.axis('equal')
plt.show()
```

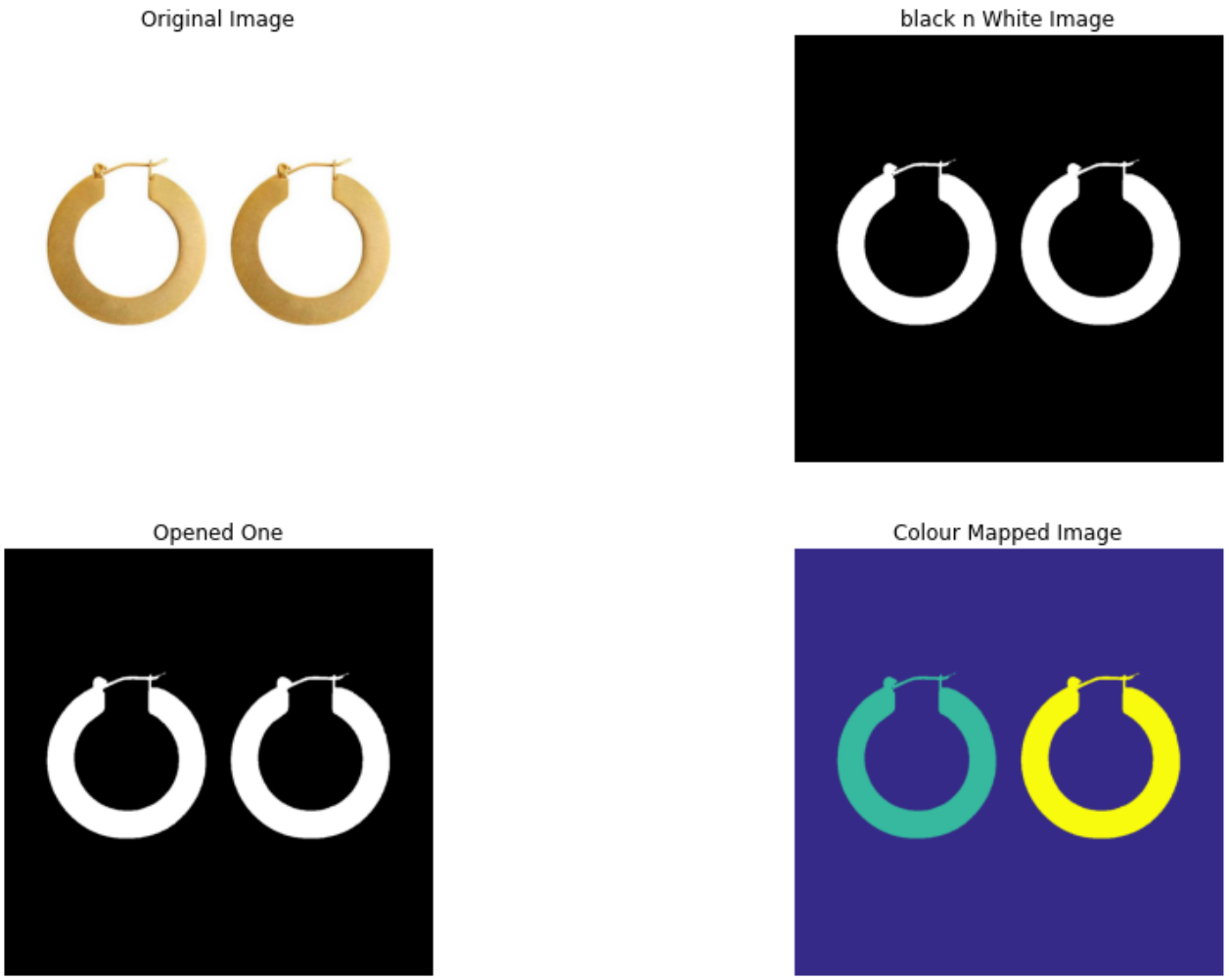


In [ ]:
```
import cv2 as cv
import numpy as np
im = cv.imread('Images/earrings.jpg', cv.IMREAD_COLOR)
assert im is not None
hsv = cv.cvtColor(im, cv.COLOR_BGR2HSV)

th, bw = cv.threshold(hsv[:,:,1], 0 ,255, cv.THRESH_BINARY + cv.THRESH_OTSU)

# Remove dots in the object
w =5
kernel = np.ones((w,w), np.uint8)
opened = cv.morphologyEx(bw, cv.MORPH_CLOSE, kernel)

retval, labels, stats, centroids = cv.connectedComponentsWithStats(bw)
```

```
colormaped = cv.applyColorMap((labels/np.amax(labels)*255).astype('uint8'), cv.COLORMAP

z = 720
f=8
for i, s in enumerate(stats):
    if i !=0:
        print('item ', i, "area in pixels =", s[4])
        print('item ', i, "area in mm^2 =", s[4]*(2.2e-6)**2*(z*z)/(f*f))

fig, ax = plt.subplots(2, 2, figsize=(15,10))
ax[0, 0].imshow(cv.cvtColor(im, cv.COLOR_BGR2RGB))
ax[0, 0].axis('off')
ax[0, 0].set_title("Original Image")
ax[0, 1].imshow(cv.cvtColor(bw,cv.COLOR_BGR2RGB))
ax[0, 1].axis('off')
ax[0, 1].set_title("black n White Image")
ax[1, 0].imshow(cv.cvtColor(opened,cv.COLOR_BGR2RGB))
ax[1, 0].axis('off')
ax[1, 0].set_title("Opened One")
ax[1, 1].imshow(cv.cvtColor(colormaped,cv.COLOR_BGR2RGB))
ax[1, 1].axis('off')
ax[1, 1].set_title("Colour Mapped Image")
```

```
item  1 area in pixels = 59143
item  1 area in mm^2 = 0.0023186421720000003
item  2 area in pixels = 59211
item  2 area in mm^2 = 0.002321308044
```

Out[ ]: Text(0.5, 1.0, 'Colour Mapped Image')



In [ ]:

```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
file_name = 'Images/allenkeys.jpg'
im = cv.imread ( file_name , cv.IMREAD_REDUCED_GRAYSCALE_2)
canny = cv.Canny( im, 50 , 150)

# Copy edges to the images that will display the results in BGR
canny_color = cv . cvtColor ( canny , cv .COLOR_GRAY2BGR)
lines = cv .HoughLines ( canny , 1 , np . pi / 180 , 170 , None , 0 , 0)
if lines is not None:
    for i in range (0 ,len(lines)):
        rho = lines [ i ] [ 0 ] [ 0 ]
        theta = lines [ i ] [ 0 ] [ 1 ]
        a = np.cos( theta )
        b = np . sin ( theta )
        x0 = a * rho
        y0 = b * rho
        pt1 = (int(x0 + 1000*(-b ) ) , int ( y0 + 1000*(a ) ) )
        pt2 = ( int ( x0 - 1000*(-b ) ) , int ( y0 - 1000*(a ) ) )
        cv.line ( canny_color , pt1 , pt2 , (0 ,0 ,255) , 1 , cv .LINE_AA)
cv.namedWindow( 'Image', cv .WINDOW_AUTOSIZE)
cv.imshow( 'Image', im)
cv . waitKey ( 0 )
cv . imshow( 'Image', canny )
cv . waitKey ( 0 )
cv . imshow( 'Image' , canny_color )
r = cv . selectROI ( 'Image' , canny_color , showCrosshair = True , fromCenter = False
cv.waitKey( 0 )
print( r )
x0 , y0 = int ( r [ 0 ] + r [ 2 ] / 2 ) , int ( r [ 1 ] + r [ 3 ] / 2 )
m = b / a # Gradient
m = np . tan(np.median(lines [ : , 0 ,1] ) )
c = y0 = m*x0 # Intercept

cv.line(canny_color,(0,int(c)),(im.shape[0],int(m*im.shape[0]+c)), (0,25,0),2,cv.LINE_A

cv . imshow('Image' , canny_color )
cv . waitKey ( 0 )
cv . destroyAllWindows()

dy = 1
y_sub_pixel = np.arange(0 , im.shape[0]-1 , dy )
f_sub_pixel = np.zeros_like( y_sub_pixel )
f_sub_pixel_nn = np.zeros_like ( y_sub_pixel )

#for i , y in enumerate ( y_sub_pixel ) :
# Your code hear to generate the pixe l values along the line
    #fig , ax = plt.subplots(figsize =(30 ,5))
    #ax . plot ( f_sub_pixel_nn )
# Your code hear to compute the widths . Keep in mind of the angle .
```