```python
#%% Part 2
#Setting up the window with 20 by 20 dots
import numpy as np
import cv2
import math
Spacing_Para = 20 # Spacing_Para represents the space between each 2 dots.
img = np.zeros((521,512,3),dtype="uint8")

#The following 'for loop' is setting up the 20 by 20 dots in the window.
for i in range (20):
    for j in range (20):
        cv2.rectangle(img,(Spacing_Para*i+9, Spacing_Para*j+9),(Spacing_Para*i+16, Spacing_Para*j+16),(255,255,255),-1)
        # Dimension for each dot is 7 by 7.

cv2.rectangle(img,(412,472),(512,512),(255,255,255),-1) # Build the rectangular for the 'Generate' button.
cv2.putText(img,'GENERATE',(422,495),cv2.FONT_HERSHEY_COMPLEX,0.5,(0,0,0),1) # Insert 'Generate text' to the button.

ix,iy=-1,-1 # Just initial values with no meaning and will be redefined later.
D_x = []
D_y = [] # Empty sets for later storing corrdinates for each highlighted points.

def I_wanna_get_hired(event,x,y,flags,param):
#Building a function  'I_wanna_get_hired'
#(1) ALlowing users to highlight dots waiting to be fitted.
#(2) Storing the coordinates for each highlighted dot.
#(3) Allowing users to deselect highlighted dots.
#(4) Remove useless coordinates when users perform deselection.
#(5) Once users click the 'GENREATE' button, using least square method
#    to calcualte the center, the short and long axes and the rotation angle for the fitted ellipse.
#(6) Plotting the fitted ellipse.

    global ix,iy
    global iX,iY
    global D_x,D_y # Setting these variable as global, so once the 'if loop' fininshed, the updated variables won't be erased.

    if event == cv2.EVENT_LBUTTONDOWN: # When users single-click the left button,
                                       # execute the following command to
                                       #(1) make highlighted dots in green and
                                       #(2) store coordinates of highlighted dots into
                                       #    D_x and D_y.
        ix,iy = x,y
```

```
    for i in range (20):
        for j in range (20):
            d = pow((Spacing_Para*i+(9+16)/2 - ix)**2 + (Spacing_Para*j+(9+16)/2 -iy)**2,0.5)
            # When single left-click happens,
            # d represents the distance between the cursor's position to each one of those 400 dots

            if d >= 0 and d <= 3.5*(pow(3.5,0.5)):
                # The tolerance we used here is half of the dignaol line of each dot, (7*sqrt(2))/2.
                # This condition satisfied when cursor's position is within the (i,j)th dot.

                cv2.rectangle(img,(Spacing_Para*i+9, Spacing_Para*j+9),(Spacing_Para*i+16, Spacing_Para*j+16),(0,255,0),-1)
                # Once the if statement regarding d is satified, I make the qualified (i,j) th dot into green.

                D_x.append(Spacing_Para*i+(9+16)/2)
                D_y.append(Spacing_Para*j+(9+6)/2)
                # At the same time, I store all the coordinates, x and y respectively, of the highlighted dots by users.

if event == cv2.EVENT_RBUTTONDOWN: # When users single-click the right button,
                                   # execute the following command to
                                   #(1) make highlighted green dots back to original white color
                                   #(2) remove coordinates of those deselected dots from
                                   #    D_x and D_y
    iX,iY = x,y


    for i in range (20):
        for j in range (20):
            d = pow((Spacing_Para*i+(9+16)/2 - iX)**2 + (Spacing_Para*j+(9+16)/2 -iY)**2,0.5)
            # When 'right single click' happens
            # d represents the distance between the cursor's position to each one of those 400 dots.

            if d >= 0 and d <= 3.5*(pow(3.5,0.5)):
                # The tolerance we used here is half of the dignaol line of each dot, (7*sqrt(2))/2.
                # This condition satisfied when cursor's position is within the (i,j)th dot.

                cv2.rectangle(img,(Spacing_Para*i+9, Spacing_Para*j+9),(Spacing_Para*i+16, Spacing_Para*j+16),(255,255,255),-1)
                # Once the if statement regarding d is satified, I make the qualified (i,j) th dot back in white color.

                D_x.remove(Spacing_Para*i+(9+16)/2)
```

```
                D_y.remove(Spacing_Para*j+(9+6)/2)
                # At the same time, I remove all the coordinates, x and y respectively, of the deselected points by users.


if event == cv2.EVENT_LBUTTONDBLCLK:
# After users highlighted points and double clicked the left button, excecuting the following commands.

    if x>= 412 and x<= 512:
        if y>= 472 and y <= 512:
    # When users made left double click, figuring out if the cursor's coordinate is within the 'GENERATE' button boundary.

            cv2.rectangle(img,(412,472),(512,512),(0,255,0),-1)
            cv2.putText(img,'GENERATE',(422,495),cv2.FONT_HERSHEY_COMPLEX,0.5,(255,255,255),1)
            # If statement is satisfied, we change the color of the button and the 'GENERATE' text.
            # The color change serves as a feedback, telling users 'GENERATE' button was pressed.


            # The following steps are calculation for the paramaters for the fitted ellipse
            N = len(D_x)
            D_x = np.array(D_x)
            D_y = np.array(D_y)

            a11 = sum(D_x * D_x * D_y * D_y)/N
            a12 = sum(D_x * D_y * D_y * D_y)/N
            a13 = sum(D_x * D_x * D_y)/N
            a14 = sum(D_x * D_y * D_y)/N
            a15 = sum(D_x * D_y)/N
            R1 = (a11,a12,a13,a14,a15)

            a21 = sum(D_x * D_y * D_y * D_y)/N
            a22 = sum(D_y * D_y * D_y * D_y)/N
            a23 = sum(D_x * D_y * D_y)/N
            a24 = sum(D_y * D_y * D_y)/N
            a25 = sum(D_y * D_y)/N
            R2 = (a21,a22,a23,a24,a25)

            a31 = sum(D_x * D_x * D_y)/N
            a32 = sum(D_x * D_y * D_y)/N
            a33 = sum(D_x * D_x)/N
            a34 = sum(D_x * D_y)/N
            a35 = sum(D_x)/N
```

```
R3 = (a31,a32,a33,a34,a35)

a41 = sum(D_x * D_y * D_y)/N
a42 = sum(D_y * D_y * D_y)/N
a43 = sum(D_x * D_y)/N
a44 = sum(D_y * D_y)/N
a45 = sum(D_y)/N
R4 = (a41,a42,a43,a44,a45)

a51 = sum(D_x * D_y)/N
a52 = sum(D_y * D_y)/N
a53 = sum(D_x)/N
a54 = sum(D_y)/N
a55 = 1
R5 = (a51,a52,a53,a54,a55)

AA = np.matrix((R1,R2,R3,R4,R5))

b1 = -1 * sum(D_x * D_x * D_x * D_y)/N
b2 = -1 * sum(D_x * D_x * D_y * D_y)/N
b3 = -1 * sum(D_x * D_x * D_x)/N
b4 = -1 * sum(D_x * D_x * D_y)/N
b5 = -1 * sum(D_x * D_x)/N
bb = np.matrix((b1,b2,b3,b4,b5))

bb = bb.T

output = AA.I * bb

g = np.asscalar(output[0])
c = np.asscalar(output[1])
d = np.asscalar(output[2])
e = np.asscalar(output[3])
f = np.asscalar(output[4])

AAA = np.matrix(((-2,-g),(-g,-2*c)))

bbb = np.matrix(((d,e)))
bbb = bbb.T

output1 = AAA.I * bbb
```

```
                x_center = int(output1[0])
                y_center = int(output1[1]) # Here I get the x,y coordinate for the center of the ellipse.

                A = 1/(x_center**2 + c*y_center**2 + g*x_center*y_center - f)
                B = c*A
                C = g*A


                xita = math.atan(C/(A-B)) # Here I get the 'angle' parameter for the fitted ellipse.
                xita_degree = int(xita/math.pi*180)   # And I make this angle in integer degrees.

                AAAA = np.matrix(((math.cos(xita)**2,math.sin(xita)**2),(math.sin(xita)**2,math.cos(xita)**2)))
                bbbb = np.matrix(((A,B)))
                bbbb = bbbb.T

                output2 = AAAA.I*bbbb

                a = int(pow(1/np.asscalar(output2[0]),0.5)) # Here we get 2 axes for the fitted ellipse.
                b = int(pow(1/np.asscalar(output2[1]),0.5))


                # Once the calculation is done, we draw the ellipse.
                cv2.ellipse(img, (x_center, y_center), (a, b), xita_degree, 0, 360, (255, 0, 0), 2)
cv2.namedWindow('image')
cv2.setMouseCallback('image',I_wanna_get_hired) # Run the 'I_wanna_get_hired' function I just built.

while(1):
    cv2.imshow('image',img)
    if cv2.waitKey(20) & 0xFF == 27:
        break
cv2.destroyAllWindows()
```