```
#%% Part 2

#Setting up the window with 20 by 20 dots
import numpy as np
import cv2
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 dots.

def I_wanna_get_hired(event,x,y,flags,param):
#Building a function called '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 calcualte the center and the radius of the fitted circle.
#(6) Plotting the fitted circle.

    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 the user single-click the left button,
                                       # execute the following command to
                                       #(1) make highlighted dots in green
                                       #(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 the user 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 user.


    if event == cv2.EVENT_LBUTTONDBLCLK:
    # After the user is done with selection and double clicks the left button, excecuting the following commands.

        if x>= 412 and x<= 512:
            if y>= 472 and y <= 512:
        # When the user 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 user 'GENERATE' button was pressed.


                # The following steps are calculation for the center and radius of the fitted circle using least square method.
                N = len(D_x)
                D_x = np.array(D_x)
                D_y = np.array(D_y)
                sum_x = sum(D_x)
                sum_x2 = sum(D_x*D_x)
                sum_y = sum(D_y)
                sum_y2 = sum(D_y*D_y)
                sum_x3 = sum(D_x*D_x*D_x)
                sum_y3 = sum(D_y*D_y*D_y)
                sum_xy = sum(D_x*D_y)
                sum_x1y2 = sum(D_x*D_y*D_y)
                sum_x2y1 = sum(D_y*D_x*D_x)

                C = N * sum_x2 - sum_x * sum_x
                D = N * sum_xy - sum_x * sum_y
                E = N * sum_x3 + N * sum_x1y2 - (sum_x2 + sum_y2) * sum_x
                G = N * sum_y2 - sum_y * sum_y
                H = N * sum_x2y1 + N * sum_y3 - (sum_x2 + sum_y2) * sum_y
                a = (H * D - E * G) / (C * G - D * D)
                b = (H * C - E * D) / (D * D - G * C)
                c = -(a * sum_x + b * sum_y + sum_x2 + sum_y2) / N

                x_center = a/(-2) # The x value is for the center of the fitted circle.
                y_center = b/(-2) # The y value is for the center of the fitted circle.
                r = 0.5*pow((a**2 + b**2 - 4*c),0.5) # The radius of the fitted circle.

                cv2.circle(img,(int(x_center),int(y_center)), int(r) ,(255,0,0),2)
                # Once the calculation is done, we draw the circle.

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()
```