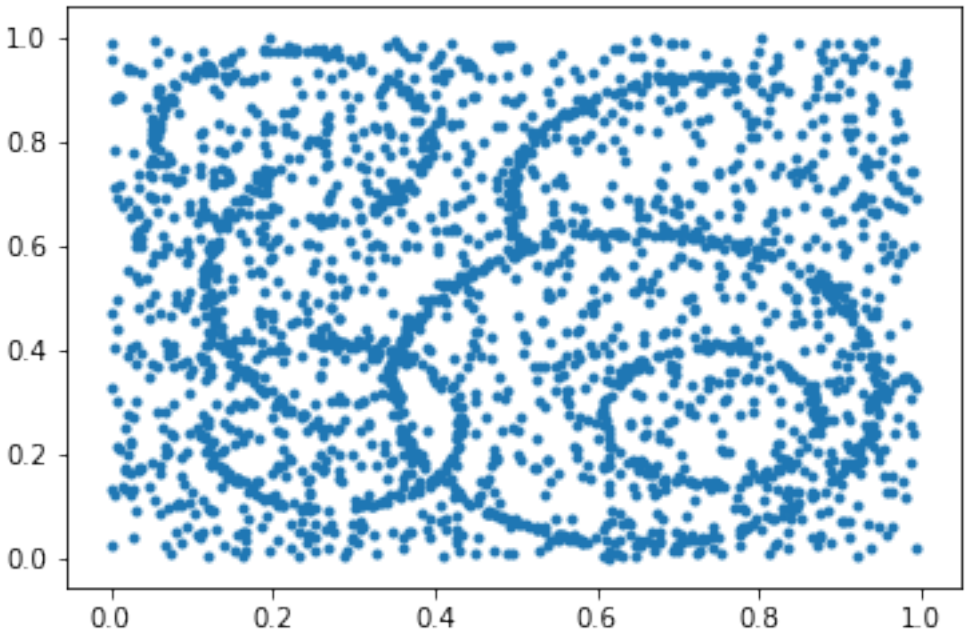
fitting-circles

December 19, 2017

0.1 3.1

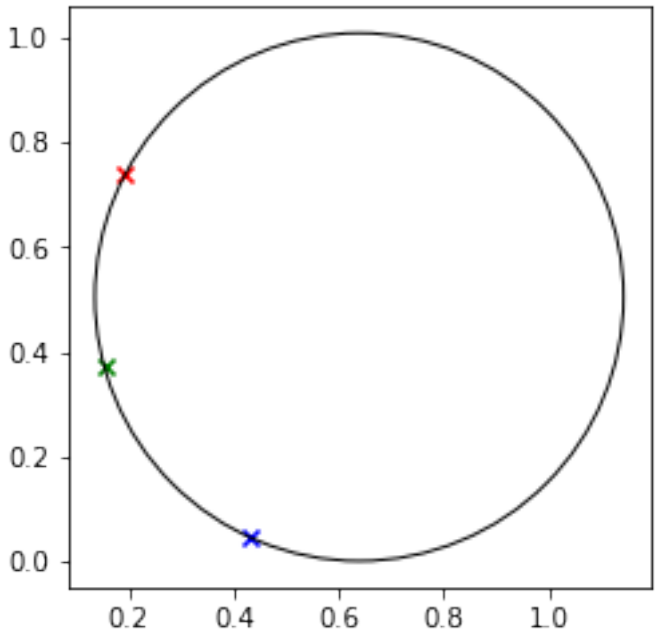
In [1]:
```
import numpy as np
import matplotlib.pyplot as plt

data = np.load("circles.npy")

plt.scatter(data[:,0], data[:,1],  marker=".")
plt.show()
```



In this picture I can see four clearly recognizable circles

In [2]:
```
# RANSAC Test if the parameters are correct

xdata = data[:,0]
ydata = data[:,1]
```

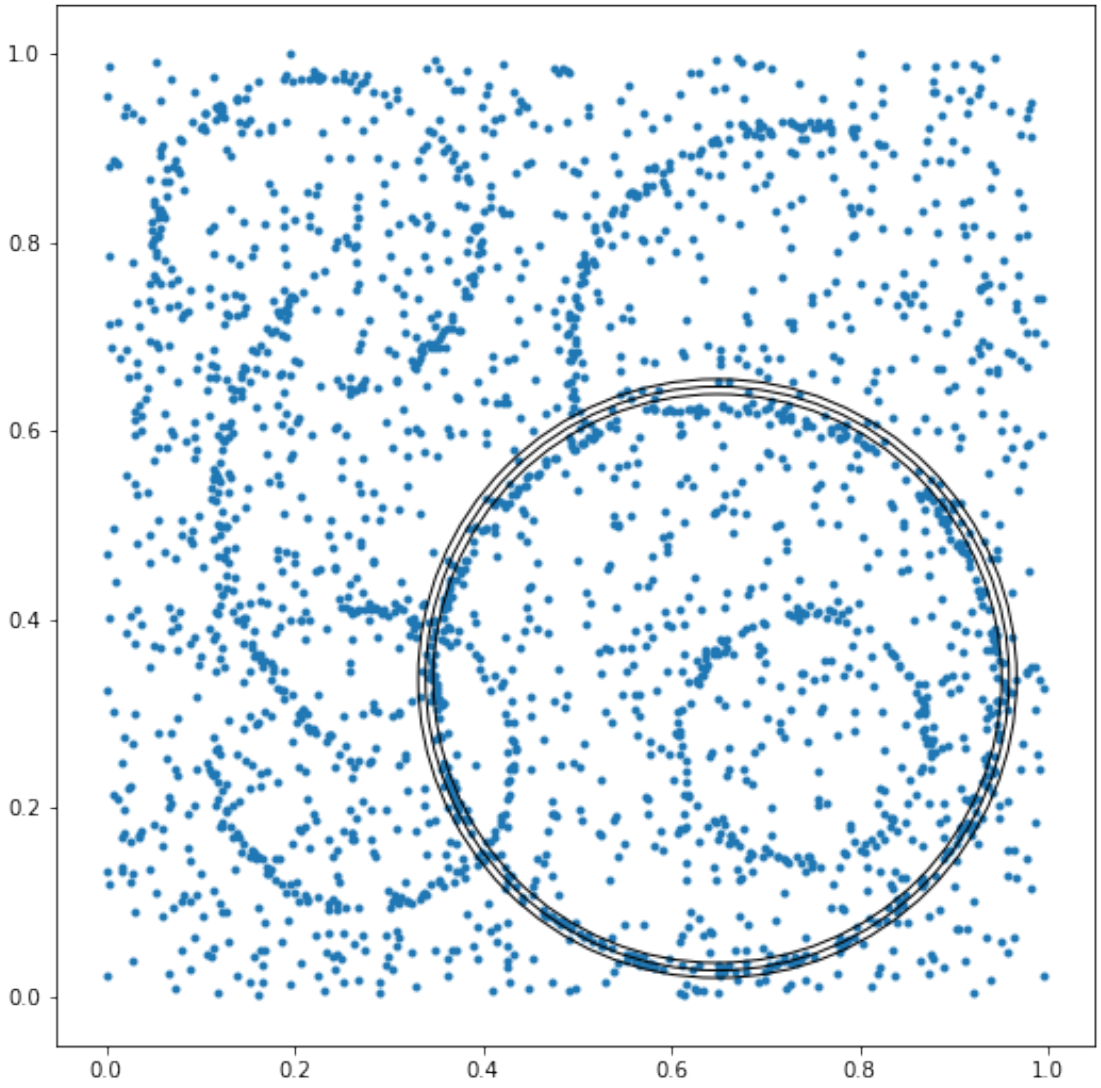
```
dim = np.shape(xdata)


N = 1
for i in range(N):
    rands = np.random.choice(dim[0], 3, replace=False)
    X = xdata[rands]
    Y = ydata[rands]
    cy = (Y[2]**2+X[2]**2-X[0]**2-Y[0]**2+((Y[2]**2-Y[1]**2+X[2]**2-X[1]**2)/(X[1]-X[2
    cy = cy/(-2*Y[0]+2*Y[2]+2*Y[2]*(X[2]-X[0])/(X[1]-X[2])-2*Y[1]*(X[2]-X[0])/(X[1]-X[
    cx = ((Y[2]-cy)**2-(Y[1]-cy)**2+X[2]**2-X[1]**2)/(-2*X[1]+2*X[2])
    r = np.sqrt((Y[2]-cy)**2+(X[2]-cx)**2)



plt.scatter(X[0],Y[0],marker="x", color = 'r') # First point, color red
plt.scatter(X[1],Y[1],marker="x", color = 'g') # Second Point
plt.scatter(X[2],Y[2],marker="x", color = 'b') # Third Point

circle = plt.Circle((cx, cy), radius=r, fill=False) # Create a circle
plt.gca().add_patch(circle) # Add it to the plot
plt.axis('scaled')
plt.show()
```



In [18]:
```
# RANSAC function
def ransac(data, e, N):
```

```
    xdata = data[:,0]
    ydata = data[:,1]
    dim = np.shape(xdata)

    max_inliers = 0

    for i in range(N):
        rands = np.random.choice(dim[0], 3, replace=False)
        X = xdata[rands]
        Y = ydata[rands]
        cy = (Y[2]**2+X[2]**2-X[0]**2-Y[0]**2+((Y[2]**2-Y[1]**2+X[2]**2-X[1]**2)/(X[1]
        cy = cy/(-2*Y[0]+2*Y[2]+2*Y[2]*(X[2]-X[0])/(X[1]-X[2])-2*Y[1]*(X[2]-X[0])/(X[
        cx = ((Y[2]-cy)**2-(Y[1]-cy)**2+X[2]**2-X[1]**2)/(-2*X[1]+2*X[2])
        r = np.sqrt((Y[2]-cy)**2+(X[2]-cx)**2)

        index = np.arange(dim[0])
        X_rest = xdata
        Y_rest = ydata

        inliers = 0
        inlier_points = []
        inlier_index = []
        for j in range(np.shape(index)[0]):
            dist = np.sqrt((X_rest[j]-cx)**2+(Y_rest[j]-cy)**2)
            if (dist < r+e) and (dist > r-e):
                inliers = inliers + 1
                inlier_points.append([X_rest[j], Y_rest[j]])
                inlier_index.append(j)

        if inliers > max_inliers:
            parameters = [cx, cy, r]
            max_inliers = inliers
            max_inlier_points = inlier_points
            max_inlier_index = inlier_index

    return parameters, max_inliers, max_inlier_points, max_inlier_index


fig = plt.figure(figsize=(9, 9))
plt.scatter(xdata,ydata,marker=".")

e = 0.008 #accuracy
parameters, max_inliers, max_inlier_points, max_inlier_index = ransac(data, e, 400)

circle = plt.Circle((parameters[0], parameters[1]), radius=parameters[2], fill=False)
plt.gca().add_patch(circle) # Add it to the plot
```

```
        circle = plt.Circle((parameters[0], parameters[1]), radius=parameters[2]-e, fill=Fals
        plt.gca().add_patch(circle) # Add it to the plot

        circle = plt.Circle((parameters[0], parameters[1]), radius=parameters[2]+e, fill=Fals
        plt.gca().add_patch(circle) # Add it to the plot

        plt.axis('scaled')
        plt.show()
        print("The Inliers for the best fit are: ", max_inliers)
```



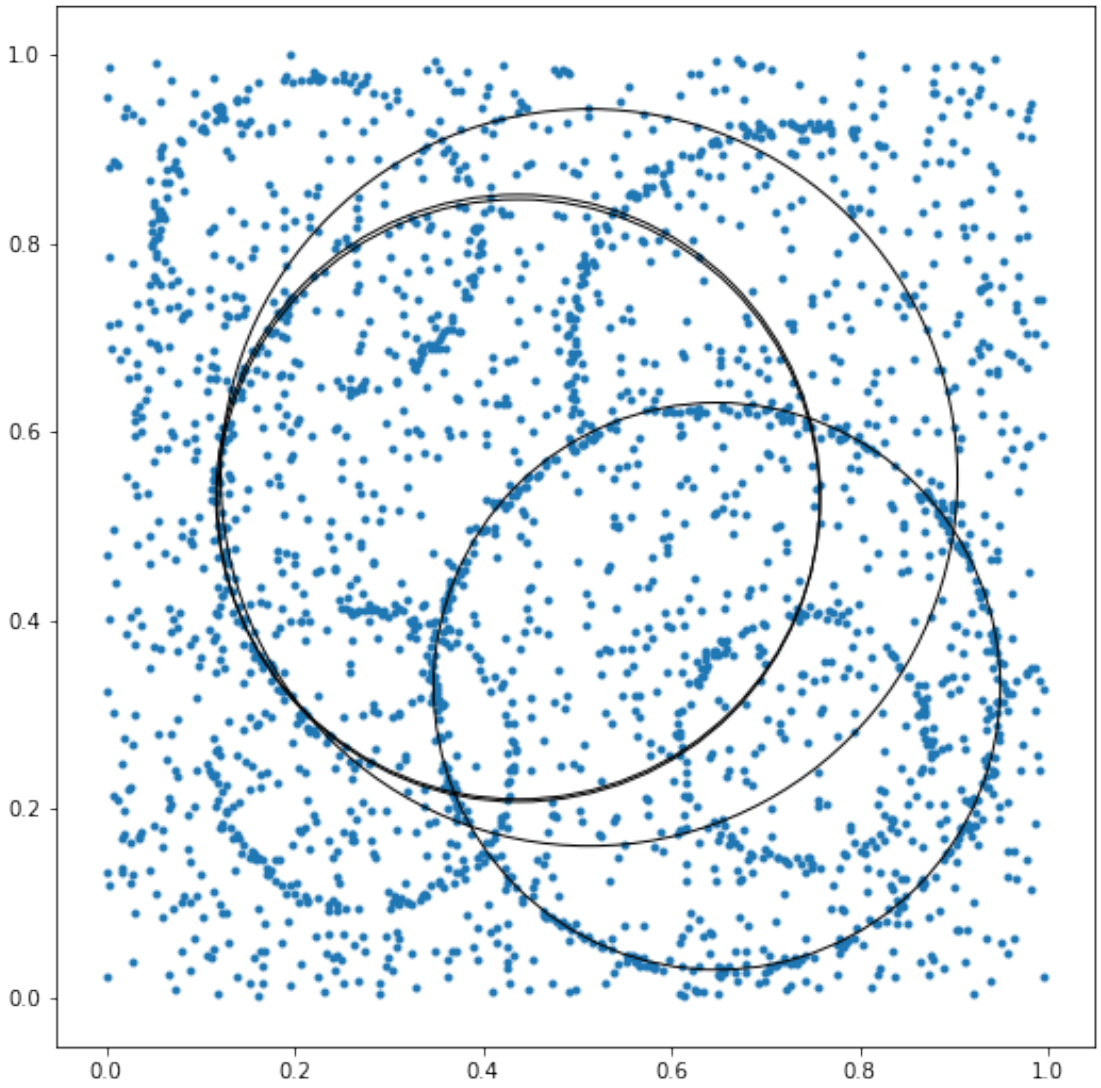

```
The Inliers for the best fit are:  218
```

```
In [43]: data = np.load("circles.npy")
```

```
xdata = np.array(data[:,0])
ydata = np.array(data[:,1])


#FIRST CIRCLE
e = 0.008 #accuracy
parameters, max_inliers, max_inlier_points, max_inlier_index = ransac(data, e, 400)

max_inlier_index = np.array(max_inlier_index)
max_inlier_points = np.array(max_inlier_points)

N = np.shape(xdata)[0]

red_index = np.setdiff1d(np.arange(N), max_inlier_index)
red_x = xdata[red_index]
red_y = ydata[red_index]

red_data = np.array([red_x,red_y]).T
print(np.shape(red_data))

fig = plt.figure(figsize=(9, 9))
plt.scatter(xdata,ydata,marker=".")

circle = plt.Circle((parameters[0], parameters[1]), radius=parameters[2], fill=False)
plt.gca().add_patch(circle) # Add it to the plot


print("The Inliers for the best fit are: ", max_inliers)


#SECOND CIRCLE

parameters, max_inliers, max_inlier_points, max_inlier_index = ransac(red_data, e, 40

circle = plt.Circle((parameters[0], parameters[1]), radius=parameters[2], fill=False)
plt.gca().add_patch(circle) # Add it to the plot


print("The Inliers for the best fit are: ", max_inliers)

max_inlier_index = np.array(max_inlier_index)

red_index = np.setdiff1d(red_index, max_inlier_index)
red_x = xdata[red_index]
red_y = ydata[red_index]

red_data = np.array([red_x,red_y]).T
print(np.shape(red_data))
```

```
        #THIRD CIRCLE

        parameters, max_inliers, max_inlier_points, max_inlier_index = ransac(red_data, e, 40

        circle = plt.Circle((parameters[0], parameters[1]), radius=parameters[2], fill=False)
        plt.gca().add_patch(circle) # Add it to the plot


        print("The Inliers for the best fit are: ", max_inliers)
        max_inlier_index = np.array(max_inlier_index)

        red_index = np.setdiff1d(red_index, max_inlier_index)
        red_x = xdata[red_index]
        red_y = ydata[red_index]

        red_data = np.array([red_x,red_y]).T
        print(np.shape(red_data))


        #FOURTH CIRCLE

        parameters, max_inliers, max_inlier_points, max_inlier_index = ransac(red_data, e, 40

        circle = plt.Circle((parameters[0], parameters[1]), radius=parameters[2], fill=False)
        plt.gca().add_patch(circle) # Add it to the plot

        plt.axis('scaled')
        print("The Inliers for the best fit are: ", max_inliers)

        plt.show()
```

```
(2081, 2)
The Inliers for the best fit are:  350
The Inliers for the best fit are:  150
(1947, 2)
The Inliers for the best fit are:  133
(1833, 2)
The Inliers for the best fit are:  119
```



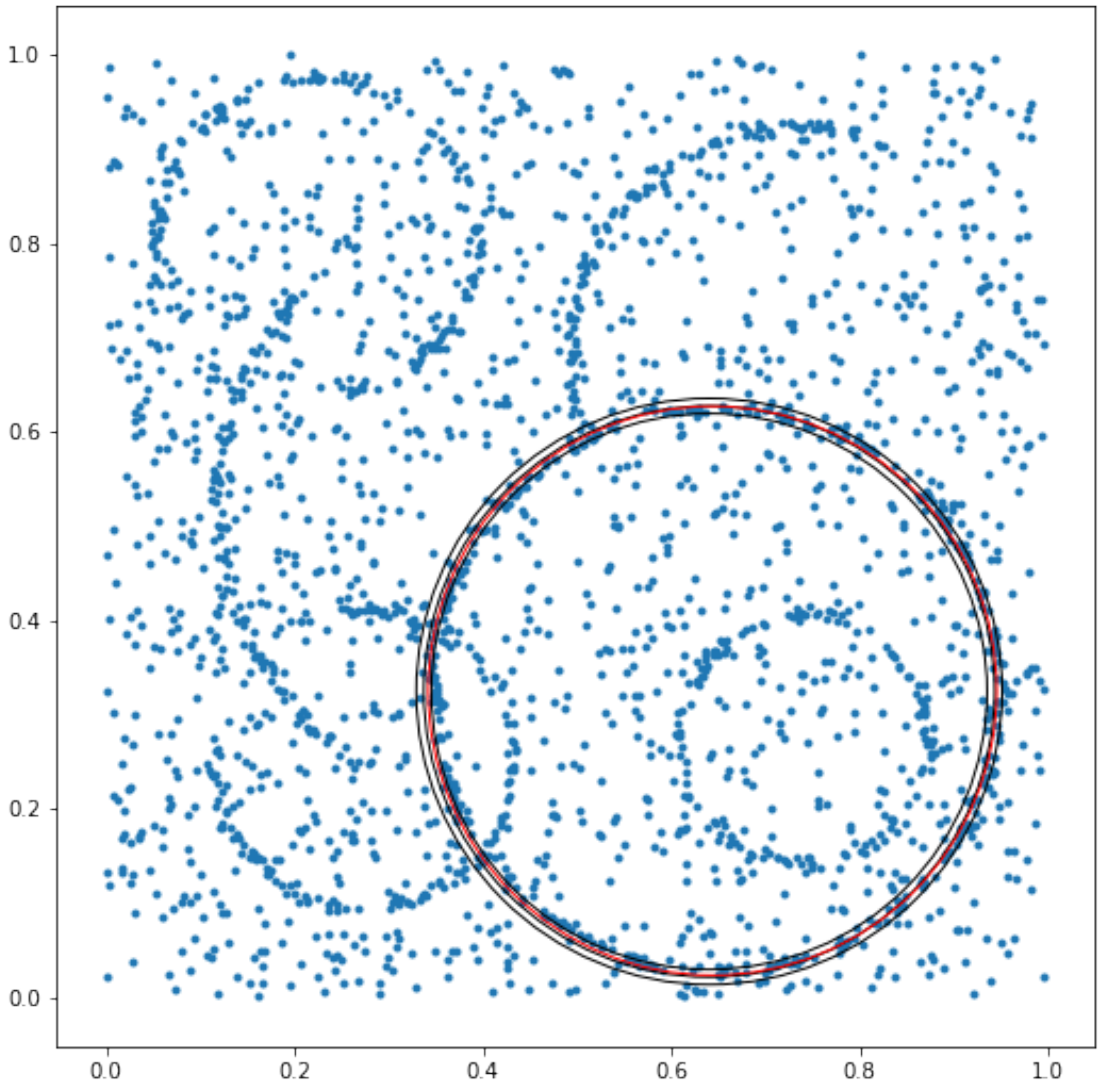
The result is sensitive to e because if you choose it to big, then there is no difference if the circle is shifted in an arbitrary direction. If it's too small none of the points get detected.

0.2 3.2

```
In [5]: data = np.load("circles.npy")

        xdata = data[:,0]
        ydata = data[:,1]

        fig = plt.figure(figsize=(9, 9))
        plt.scatter(xdata,ydata,marker=".")

        e = 0.008 #accuracy
```

```
parameters, max_inliers, max_inlier_points = ransac(data, e, 400)

circle = plt.Circle((parameters[0], parameters[1]), radius=parameters[2], fill=False) #
plt.gca().add_patch(circle) # Add it to the plot

circle = plt.Circle((parameters[0], parameters[1]), radius=parameters[2]-e, fill=False)
plt.gca().add_patch(circle) # Add it to the plot

circle = plt.Circle((parameters[0], parameters[1]), radius=parameters[2]+e, fill=False)
plt.gca().add_patch(circle) # Add it to the plot

plt.axis('scaled')
print("The Inliers for the best fit are: ", max_inliers)

max_inlier_points = np.array(max_inlier_points)

Y = []
for i in range(np.shape(max_inlier_points)[0]):
    Y.append(max_inlier_points[i,:].T.dot(max_inlier_points[i,:]))

inlier_x = max_inlier_points[:,0]
inlier_y = max_inlier_points[:,1]

X = np.ones((inlier_x.shape[0],3))

X[:,0] = inlier_x
X[:,1] = inlier_y

b = np.linalg.lstsq(X,Y)[0]

cx = b[0]/2
cy = b[1]/2
r = np.sqrt(b[2]+cx**2+cy**2)

parameters = [cx,cy,r]

print(cx,cy,r)

circle = plt.Circle((parameters[0], parameters[1]), radius=parameters[2], fill=False, c
plt.gca().add_patch(circle) # Add it to the plot
plt.show()
```

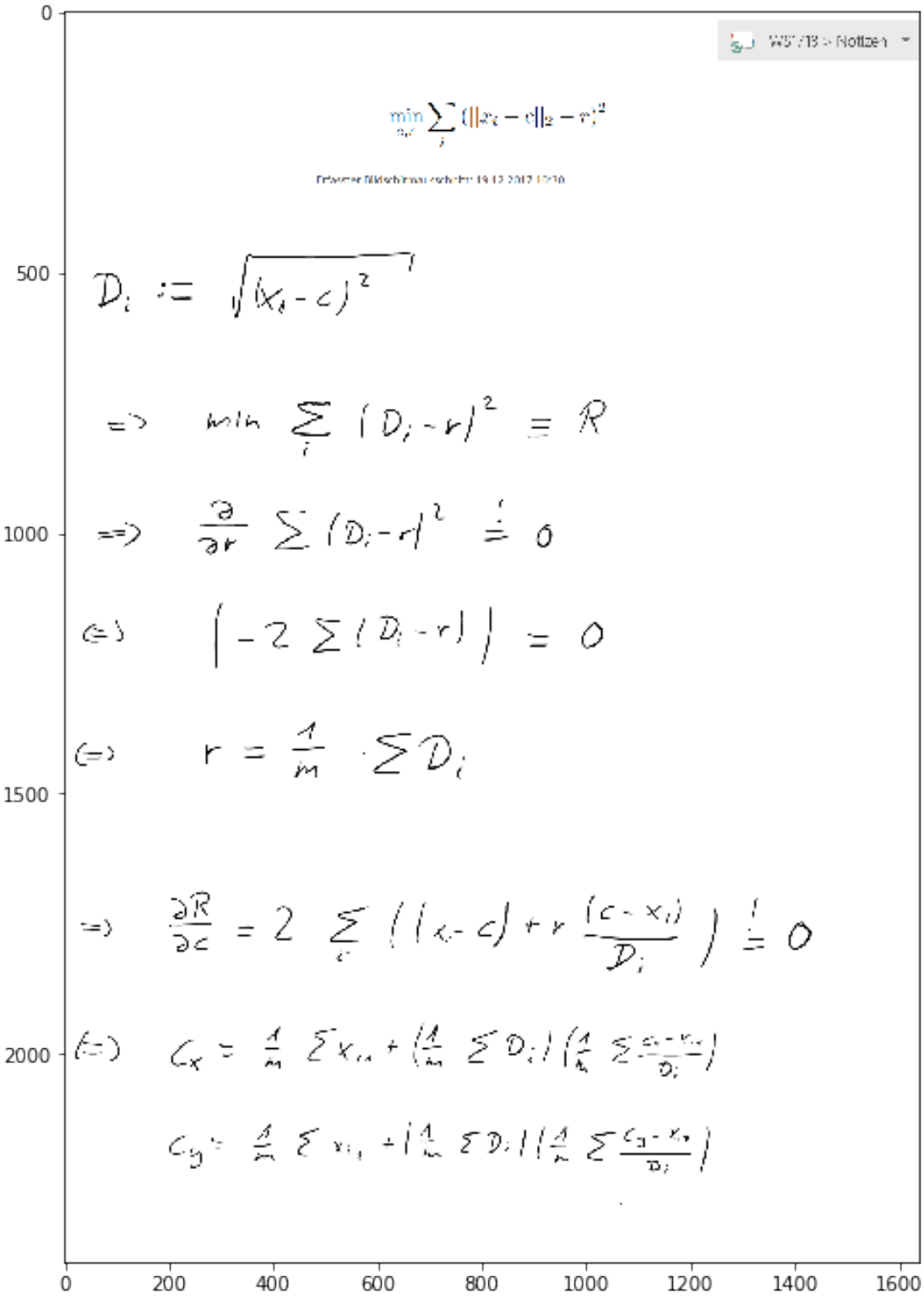
```
The Inliers for the best fit are:  259
0.643612382035 0.32526563289 0.301740050641
```



As one can see in the plot, the algebraic-distance-method (red circle) gives a similar result as the RANSAC-algorithm

0.3 3.3

```
In [48]: import matplotlib.pyplot as plt
         import matplotlib.image as mpimg
         fig = plt.figure(figsize=(12, 12))
         img=mpimg.imread('derivation.png')
         imgplot = plt.imshow(img)
         plt.show()
```




```
In [13]: import scipy.optimize as opt
```

```
data = np.load("circles.npy")

xdata = data[:,0]
ydata = data[:,1]

fig = plt.figure(figsize=(9, 9))
plt.scatter(xdata,ydata,marker=".")

e = 0.008 #accuracy
parameters, max_inliers, max_inlier_points = ransac(data, e, 400)

circle = plt.Circle((parameters[0], parameters[1]), radius=parameters[2], fill=False)
plt.gca().add_patch(circle) # Add it to the plot

circle = plt.Circle((parameters[0], parameters[1]), radius=parameters[2]-e, fill=Fals
plt.gca().add_patch(circle) # Add it to the plot

circle = plt.Circle((parameters[0], parameters[1]), radius=parameters[2]+e, fill=Fals
plt.gca().add_patch(circle) # Add it to the plot

plt.axis('scaled')

print("The Inliers for the best fit are: ", max_inliers)

max_inlier_points = np.array(max_inlier_points)



def f(x, c):
    xdata = x[:,0]
    ydata = x[:,1]
    m = xdata.shape[0]
    cx = 1/m*np.sum(xdata) + (1/m*np.sum(np.sqrt((xdata-c[0])**2+(ydata-c[1])**2))*1/
    cy = 1/m*np.sum(ydata) + (1/m*np.sum(np.sqrt((xdata-c[0])**2+(ydata-c[1])**2))*1/
    return [cx,cy]

def residual(p, data):
    return f(data, *p)

p0 = [1]

popt, pcov = opt.leastsq(residual, p0, args=(data))

print(popt)
```

```
The Inliers for the best fit are:  350
```

```
---------------------------------------------------------------------------

IndexError                                Traceback (most recent call last)

<ipython-input-13-42afd07d6355> in <module>()
     42 p0 = [1]
     43
---> 44 popt, pcov = opt.leastsq(residual, p0, args=(data))
     45
     46 print(popt)


~\Anaconda3\lib\site-packages\scipy\optimize\minpack.py in leastsq(func, x0, args, Dfun
    375     if not isinstance(args, tuple):
    376         args = (args,)
--> 377     shape, dtype = _check_func('leastsq', 'func', func, x0, args, n)
    378     m = shape[0]
    379     if n > m:


~\Anaconda3\lib\site-packages\scipy\optimize\minpack.py in _check_func(checker, argname
     24 def _check_func(checker, argname, thefunc, x0, args, numinputs,
     25                 output_shape=None):
---> 26     res = atleast_1d(thefunc(*((x0[:numinputs],) + args)))
     27     if (output_shape is not None) and (shape(res) != output_shape):
     28         if (output_shape[0] != 1):


<ipython-input-13-42afd07d6355> in residual(p, data)
     38
     39 def residual(p, data):
---> 40     return f(data, *p)
     41
     42 p0 = [1]


<ipython-input-13-42afd07d6355> in f(x, c)
     33     ydata = x[:,1]
     34     m = xdata.shape[0]
---> 35     cx = 1/m*np.sum(xdata) + (1/m*np.sum(np.sqrt((xdata-c[0])**2+(ydata-c[1])**2))*
     36     cy = 1/m*np.sum(ydata) + (1/m*np.sum(np.sqrt((xdata-c[0])**2+(ydata-c[1])**2))*
     37     return [cx,cy]


IndexError: invalid index to scalar variable.
```