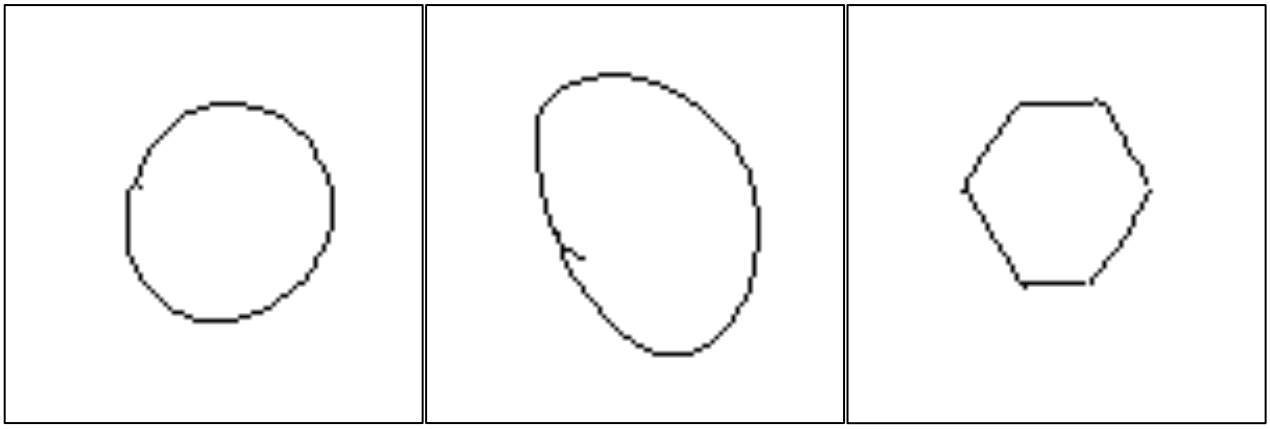
Perfecting Your Drawing

學號: 109062129 姓名: 林奕廷

1.



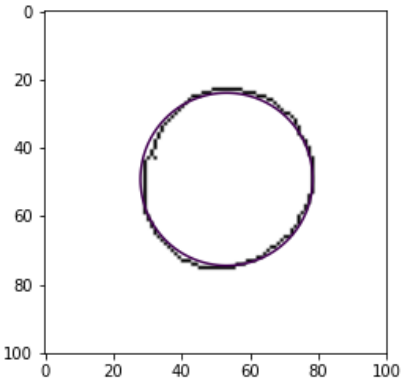
2.

a.

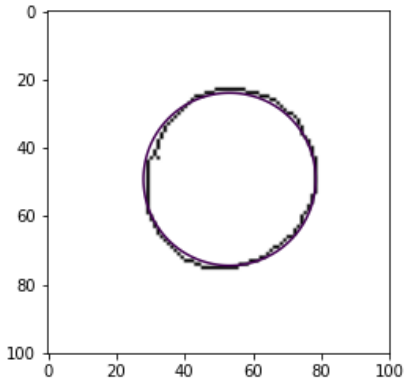
```
#----------Q2----------#
# normal equation method. A^T*Ax=A^T*b -> x = (A^T*A)^-1*(A^T*b)
x = np.dot(np.linalg.inv(np.dot(A.T, A)), np.dot(A.T, b))
e = b - np.dot(A, x) #error(residual)
e = np.dot(e, e) #squared
#print(e)
```

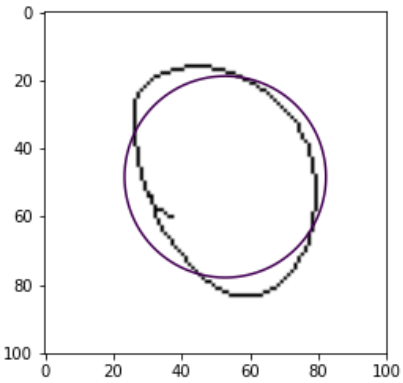
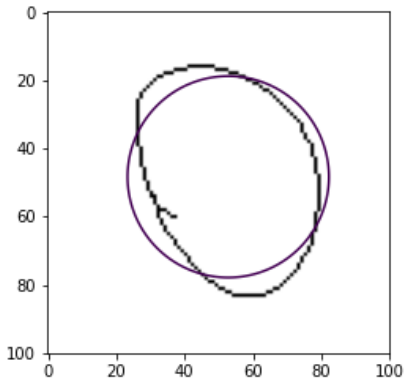
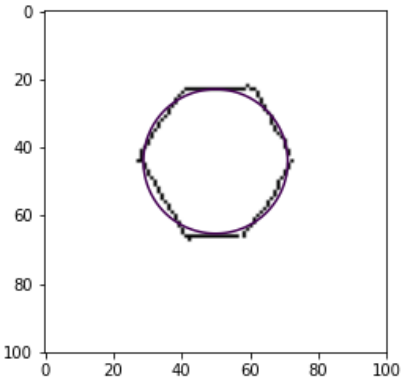
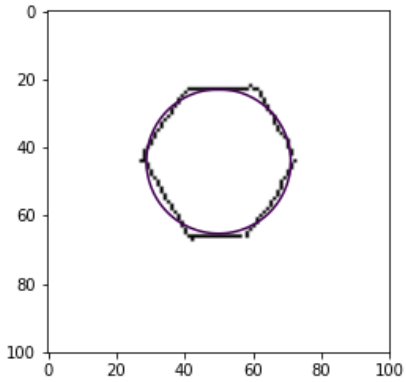
Here are the results:

Normal equation:



lstsq:



由這三個例子可以知道，這兩個方式在一般情況下，得到的結果是相同的。

b.

Let x be the best fit solution to Ax ~= b.

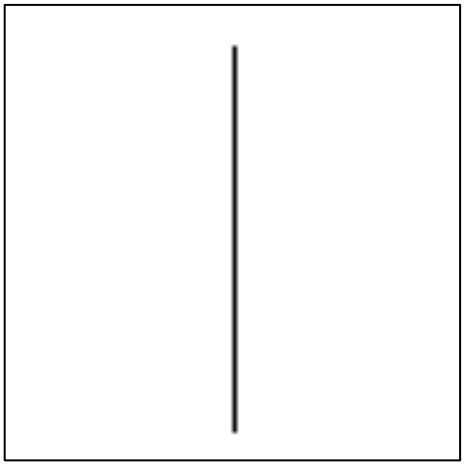
Define error e = b – Ax (also called residuals), where b is the expected result, Ax is the best fit one. This error e can be obtained from the normal equation easily.

the returned value residual of numpy.linalg.lstsq is the e squared.

The closeness can be measured by this error e. The smaller, the better.

c.

The rank of all my examples are 3. The image below has the rank 2.

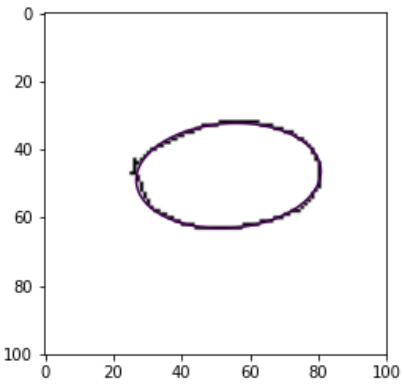


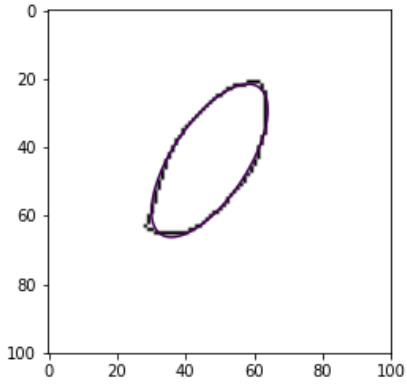
3.

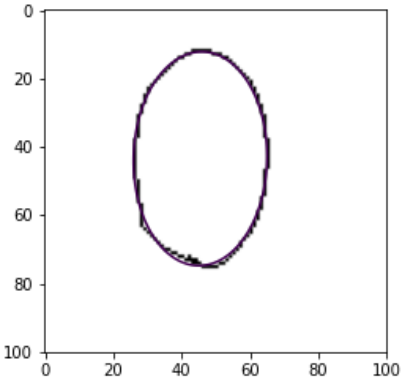
```
# TODO: ellipse formula here.
A[i, :] = [1.*x*x, 1.*x*y, 1.*y*y, 1.*x, 1.*y]
b[i] = 1.
```

I simply use the formula provided.

Here are the results:







4.

Code snippet:

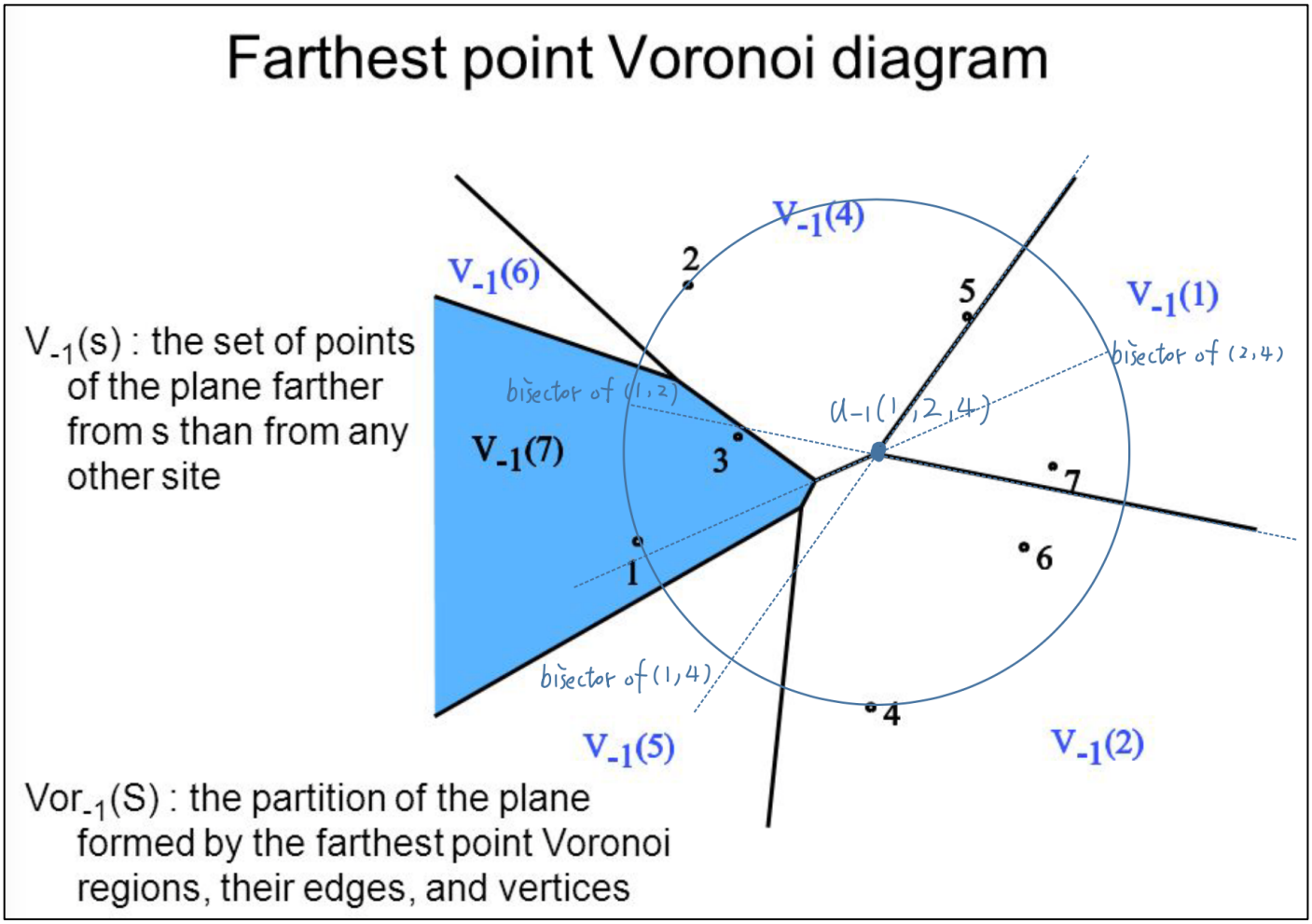
```
def dataSampling(x):
    ret = []
    for i in x :
      flag = 1
      for j in ret :
        # 60*60 circle area
        if (i[0] - j[0]) * (i[0] - j[0]) + (i[1] - j[1]) * (i[1] - j[1]) < 3600 :
          flag = 0
          break
      if(flag) :
        ret.append(i)

    return ret
```

The algorithm I implement has a nested for-loop. The condition to add a new point to the solution is as follow:

- If the new point is not in the 60*60 circular area of any existing solution points, append it to the ret list.

As for the time complexity, the worst case is n^2, since it is a nested for-loop structure.
For the best case, if for the input image , the points are seperated enough from each other, such that every points have distance to existing points larger than 60, every points will be added to the sample. The inner loop will break after entering. So the best case has time complexity n.

5.

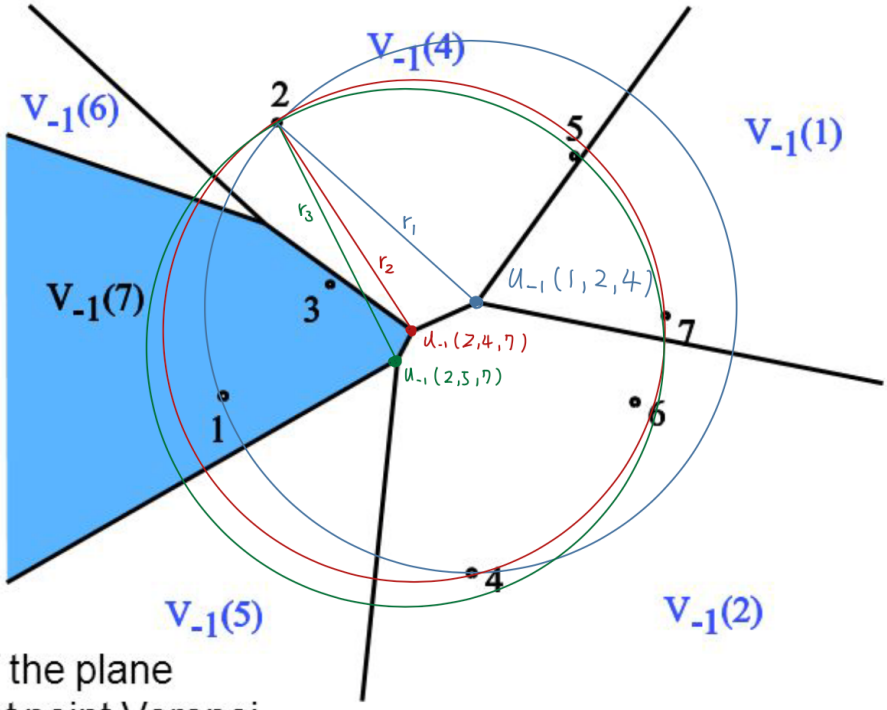


Take this diagram for example. After complete the Voronoi diagram, there are some vertices, either constructed from two or three bisectors. The vertex, denoted u-1(1, 2, 4), is the intersection of three bisector: bisector of(1, 2), (1, 4) and (2, 4). Take this as the center of the circle, we can draw a candidate enclosing circle which passes through point 1, 2 and 4.

Repeats this step for each vertices, we call draw three circles:

Farthest point Voronoi diagram

$V_{-1}(s)$: the set of points of the plane farther from s than from any other site



$Vor_{-1}(S)$: the partition of the plane formed by the farthest point Voronoi regions, their edges, and vertices

For these three circles, compare the radius of them , we can find the smallest enclosing circle.