실내 측위
(Calibrating Indoor Positioning)

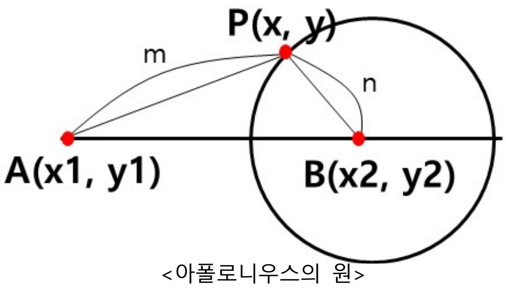
베스텔라랩 박형준

목차

1. 사용 이론

1. 아폴로니우스의 원(Apollonius Circle)

평면 위의 두 정점 A, B에 대하여 거리의 비가 m : n인 점의 자취 P가 나타내는 도형으로 선분 AB를 m : n으로 외분하는 점을 지름의 양 끝점으로 하는 원.



<아폴로니우스의 원>

$$\sqrt{x^2+y^2+z^2} + \sqrt{(x-a)^2+(y-b)^2+z^2} = m : n$$

$$(x-al)^2 + (y-bl)^2 + z^2 = (a^2+b^2)(l^2-l) \quad \left(l = \frac{m^2}{k}, k = m^2 - m^2\right)$$

이웃한 두 개의 RSSI 신호를 이용하여 두 신호의 거리의 비가 일정한 점들의 자취 중 절편(Intercept)을 이용.

2. 베이지안 추정(Bayesian Estimation)

주어진 데이터 $\{x_1,\cdots,x_N\}$를 기반으로 모수 μ의 조건부 확률분포 $p(\mu|x_1,\cdots,x_N)$를 계산하는 작업.

$$p(\mu \mid x_1,\cdots,x_N) = p(x_1,\cdots,x_N \mid \mu) \cdot p(\mu)/p(x_1,\cdots,x_N) \propto p(x_1,\cdots,x_N \mid \mu) \cdot p(\mu)$$

- $p(\mu)$: 모수의 사전(Prior)분포. 사전 분포는 베이지안 추정 작업을 하기 전에 이미 알고 있던 모수 μ의 분포를 의미.
- 모수의 분포에 대해 아무런 지식이 없는 경우, 균일(uniform) 분포 Beta(1,1)나 0을 중심으로 가지는 정규분포 N(0,σ02) 등의 무정보 분포(non-informative distribution)를 사용.
- $p(\mu \mid x1,\cdots,x\text{N})$: 모수의 사후(Posterior)분포. 수학적으로는 데이터 $x1,\cdots,x\text{N}$가 주어진 상태에서 μ에 대한 조건부 확률 분포. 베이즈 추정 작업을 통해 구하고자 하는 결과물.
- $p(x1,\cdots,x\text{N} \mid \mu)$: 가능도(likelihood)분포. 모수 μ가 특정한 값으로 주어졌을 때, 주어진 데이터 $\{x1,\cdots,x\text{N}\}$가 나올 수 있는 확률값을 의미.

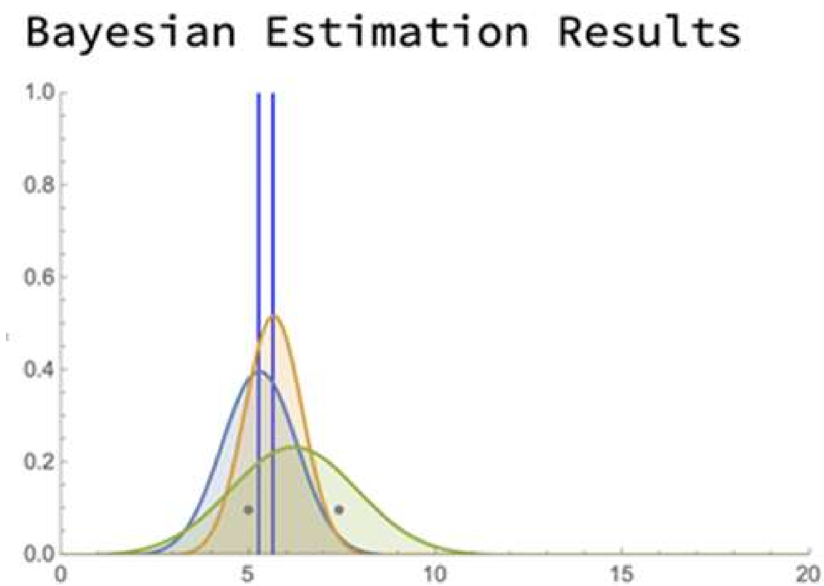
데이터의 분포가 정규분포일 때 베이즈룰(bayes rule)에 따라 사후확률을 베이지안 방법으로 추정.

$$p(\mu|D) = \frac{p(D|\mu,\sigma^2)p(\mu|\mu_0,\sigma_0^2)}{p(D)}$$

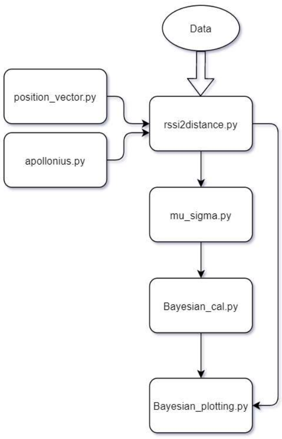
- N개의 관측 데이터, $p(\mu|\mu_0,\sigma_0^2)$를 파라미터 μ의 사전확률.
- 초기 모수의 사전 분포는 무정보 분포로 사용하고 이후에는 이 전의 모수를 사용.
- 베이즈 정리를 이용하여 사후 분포를 계산하여 하이퍼 모수(Hypermu, Hypersigma)를 갖는 정규 분포 계산.

$$\mu_0' = \frac{\sigma^2\mu_0}{N\sigma_0^2+\sigma^2} + \frac{N\sigma_0^2}{N\sigma_0^2+\sigma^2}\frac{\sum x_i}{N}$$

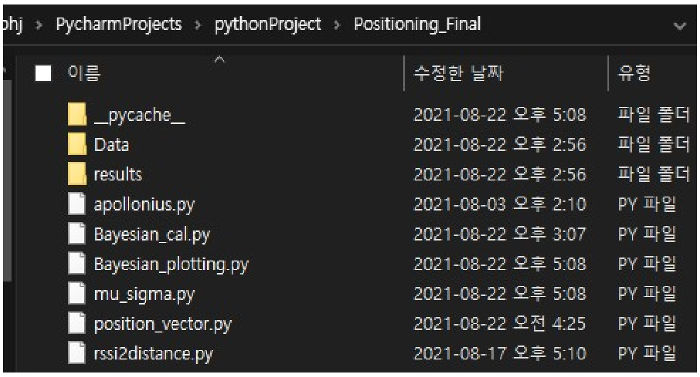
$$\frac{1}{\sigma_0'^2} = \frac{1}{\sigma_0^2} + \frac{N}{\sigma^2}$$



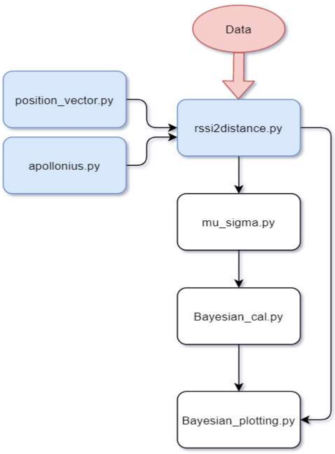
2. 주요 코드 설명



<코드 흐름>



<코드 실행에 필요한 파일 목록>




```
Sample_data = []
count = []
# 같은 시간 Data에서 Combonation으로 2개씩 추출 후 Sample_data list에 저장.
for i in range(0, lentime_data):
    if len(time_data[i]) >= 2:
        Sample = list(itertools.combinations(time_data[i], 2))
        Sample_data.append(list(Sample))
        count.append(len(Sample))
        # Check
        # print(Sample)
        # print(len(Sample))
    elif len(time_data[i]) < 1:
        continue

sum_count = sum(count)
lenSample_data = len(Sample_data)
# Check
print('Sample_data')
print(Sample_data)
print(lenSample_data)
```

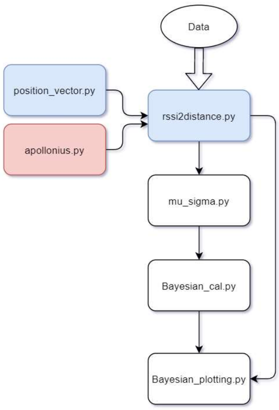
<같은 시간에서 데이터를 뽑아내는 코드>

- rssi2distance.py에서 Data 폴더에서 저장된 csv 파일을 읽어와 같은 시간에 대해서 조합(Combination)으로 Sample_data에 저장.
- Sample_data 형태.
  [[([time, minor, rssi], [time, minor, rssi]), ([· · ·] , [· · ·]), · · ·, ([time, minor, rssi], [time, minor, rssi])], [([· · ·])]]
- itertools.combinations를 사용하면 tuple로 추출.

```python
for i in range(0, lenSample_data - 1):
    len1 = len(Sample_data[i])
    for j in range(0, len1):
        # Minor가 같을 경우.
        if Sample_data[i][j][0][1] == Sample_data[i][j][1][1]:
            continue
        else:
            # Sample_data에서 앞의 Data의 RSSI가 더 클 때.
            if Sample_data[i][j][0][2] > Sample_data[i][j][1][2]:
                a, b = dirvector(int(Sample_data[i][j][1][1]), int(Sample_data[i][j][0][1]))
                rssidif = abs(int(Sample_data[i][j][0][2]) - int(Sample_data[i][j][1][2]))
                ratio = 10 ** (rssidif / 20)
                apollo_data = apollo(ratio, a, b)
                if (a == 0 and b == distance) or (a == 0 and b == -distance) or \
                    (a == distance and b == 0) or (a == -distance and b == 0) or \
                    (a == distance and b == distance) or (a == distance and b == -distance) or \
                    (a == -distance and b == distance) or (a == -distance and b == -distance):
                    # rssidif가 1 ~ 5일 때 apollonius 원이 크게 그려지기 때문에 큰 값들 제거.
                    if rssidif < 6:
                        if apollo_data[0] > halfdis or apollo_data[0] < -halfdis:
                            apollo_data[0] = False
                        if apollo_data[1] > halfdis or apollo_data[1] < -halfdis:
                            apollo_data[1] = False
                        if apollo_data[2] > halfdis or apollo_data[2] < -halfdis:
                            apollo_data[2] = False
                        if apollo_data[3] > halfdis or apollo_data[3] < -halfdis:
                            apollo_data[3] = False

                    intercept.append(Sample_data[i][j][0][0])
                    intercept.append(apollo_data)
                    intercept.append(int(Sample_data[i][j][0][1]))
                    position.append(a)
                    position.append(b)
                    position.append(rssidif)
                    position.append(11111)
                    position.append(Sample_data[i][j][0][0])
                    position.append(apollo_data)
```

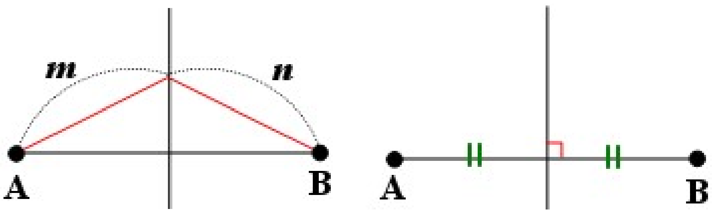
<두 개의 RSSI를 받아 처리>

- Sample_data에서 Minor값이 이웃하지 않거나 같을 경우 data 사용X
- 두 개의 RSSI 신호를 이용하여 아폴로니우스 원에 절편값을 계산할 때, RSSI값의 차이가 5 이하면 아폴로니우스 원이 너무 크게 그려지는 문제가 있음.
- 이를 해결하기 위해 두 신호의 RSSI 차이가 5 이하일 때 ±10 범위를 넘어가면 False처리.
- 결과값을 Intercept 리스트에 append, Position 리스트는 벡터와 두 RSSI의 차이, 절편값을 확인해보기 위해 따로 저장.




```python
def apollo(r, a, b):
    if r == 1 and a == 0:
        y1 = (b / 2)
        return False, False, False, y1
    elif r == 1 and b == 0:
        x1 = (a / 2)
        return x1, False, False, False
    elif r == 1 and (a != 0 and b != 0):
        x1 = (b ** 2) / (2 * a) + (a / 2)
        y1 = (a ** 2) / (2 * b) + (b / 2)
        return x1, False, y1, False
    else:
        l = 1 / (1 - r ** 2)
        x1 = a * l + (a ** 2 * l ** 2 - a ** 2 * l - b ** 2 * l - 9) ** (1/2)
        x2 = a * l - (a ** 2 * l ** 2 - a ** 2 * l - b ** 2 * l - 9) ** (1/2)
        y1 = b * l + (b ** 2 * l ** 2 - a ** 2 * l - b ** 2 * l - 9) ** (1/2)
        y2 = b * l - (b ** 2 * l ** 2 - a ** 2 * l - b ** 2 * l - 9) ** (1/2)
        return [x1, x2, y1, y2]
```

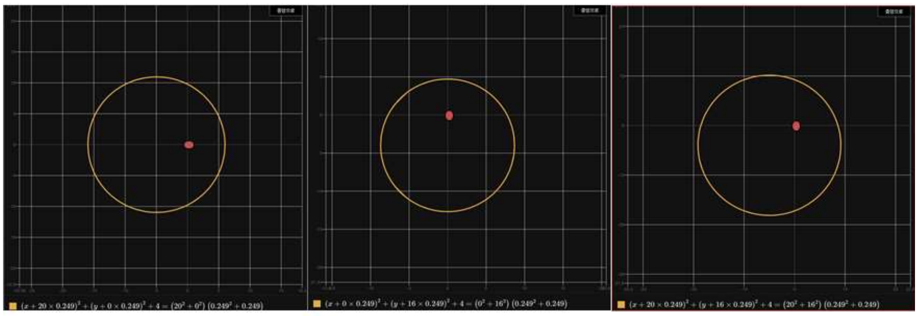
<조합으로 뽑아낸 Sample_Data에서 아폴로니우스 원 계산>



<ratio = 1일 때, 예외 Case>

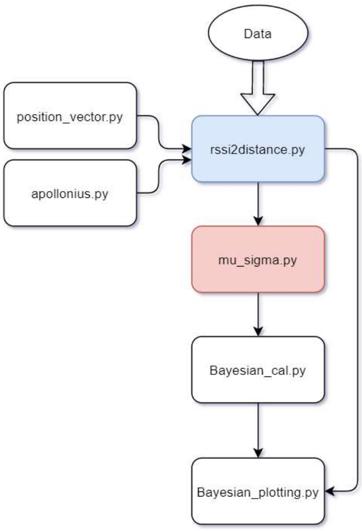
- rssi2distance.py에서 Apollonius.py를 호출하여 아폴로니우스 절편 계산.
- ratio = 1일 때 예외 처리.

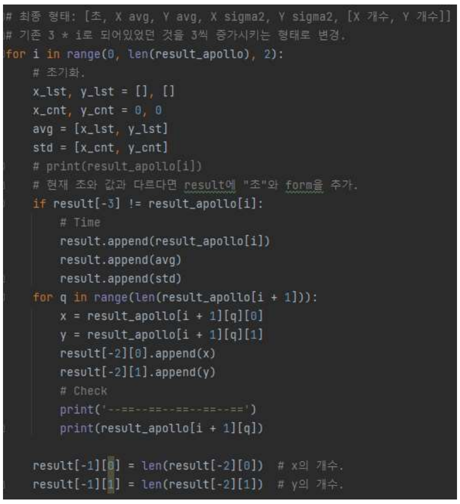
아폴로니우스 원(Apollonius Circle) 예시



<141 - 142>　<141 - 131>　<141 - 132>

- 빨간색 점이 원점(0, 0)
- 아폴로니우스 원을 그리고 원의 절편값을 list에 append.
- rssidif가 5 이하일 때, 아폴로니우스 원이 매주 크게 그려짐.





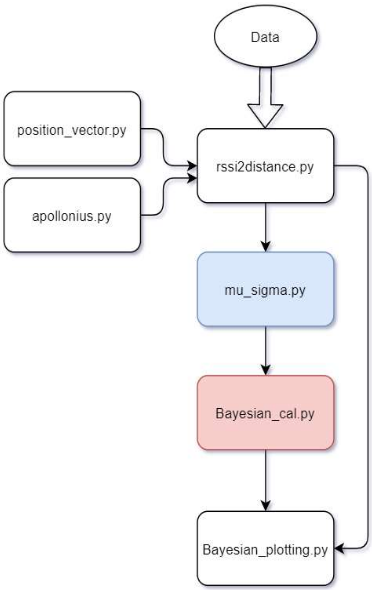
```
# 최종 형태: [초, X avg, Y avg, X sigma2, Y sigma2, [X 개수, Y 개수]]
# 기존 3 * i로 되어있었던 것을 3씩 증가시키는 형태로 변경.
for i in range(0, len(result_apollo), 2):
    # 초기화.
    x_lst, y_lst = [], []
    x_cnt, y_cnt = 0, 0
    avg = [x_lst, y_lst]
    std = [x_cnt, y_cnt]
    # print(result_apollo[i])
    # 현재 조와 값과 다르다면 result에 "초"와 form을 추가.
    if result[-3] != result_apollo[i]:
        # Time
        result.append(result_apollo[i])
        result.append(avg)
        result.append(std)
    for q in range(len(result_apollo[i + 1])):
        x = result_apollo[i + 1][q][0]
        y = result_apollo[i + 1][q][1]
        result[-2][0].append(x)
        result[-2][1].append(y)
        # Check
        print('--==--==--==--==--==')
        print(result_apollo[i + 1][q])

    result[-1][0] = len(result[-2][0])  # x의 개수.
    result[-1][1] = len(result[-2][1])  # y의 개수.
```

<계산을 위해 data 형태 수정>

- Intercept를 이용하여 베이지안 계산을 위해 mu_sigma.py에서 같은 시간일 때의 x, y 좌표의 평균과, 표준편차 계산하기 위해 data 수정.

최종 Data 형태.

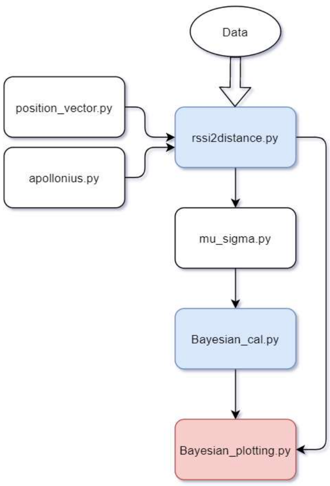
[time, x_avg, y_avg, x_sigma, y_sigma, [x_len, y_ len]
[time, x_avg, y_avg, x_sigma, y_sigma, [x_len, y_ len]
. . .
[time, x_avg, y_avg, x_sigma, y_sigma, [x_len, y_ len]
[time, x_avg, y_avg, x_sigma, y_sigma, [x_len, y_ len]




```python
for i in range(0, data_len, 6):
    # Time Check.
    # print('time')
    # print(mu_sigma.result_summary[i])
    Bayesian_result.append(mu_sigma.result_summary[i])
    xavg = mu_sigma.result_summary[i + 1]       # xmu
    yavg = mu_sigma.result_summary[i + 2]       # ymu
    xsigma = mu_sigma.result_summary[i + 3]     # xsigma
    xsigma2 = xsigma ** 2                       # xsigma2
    ysigma = mu_sigma.result_summary[i + 4]     # ysigma
    ysigma2 = ysigma ** 2                       # ysigma2
    X_N = mu_sigma.result_summary[i + 5][0]     # 원소 x의 갯수
    Y_N = mu_sigma.result_summary[i + 5][1]     # 원소 y의 갯수

    # xx = np.linspace(1.8, 2.2, 1000)
    np.random.seed(1)
    # Intercept avg, sigma => mu, sigma
    # 이전 mu, sigma => mu0, sigma0
    for j in range(4):  # 3차 추정.
        # 데이터를 정규 분포로.
        x = sp.stats.norm(xavg).rvs(X_N)
        y = sp.stats.norm(yavg).rvs(Y_N)
        X_Hypermu0 = xsigma2 / (X_N * xsigma20[-1] + xsigma2) * xmu0[-1] + \
                     (X_N * xsigma20[-1]) / (X_N * xsigma20[-1] + xsigma2) * x.mean()
        X_Hypersigma20 = 1 / (1 / xsigma20[-1] + X_N / xsigma2)
        Y_Hypermu0 = ysigma2 / (Y_N * ysigma20[-1] + ysigma2) * ymu0[-1] + \
                     (Y_N * ysigma20[-1]) / (Y_N * ysigma20[-1] + ysigma2) * y.mean()
        Y_Hypersigma20 = 1 / (1 / ysigma20[-1] + Y_N / ysigma2)

        if xmu0 == 0 or xsigma20 == 0:
            X_Hypermu0 = xmu0[-1]
            X_Hypersigma20 = xsigma20[-1]
        if Y_Hypermu0 == 0 or Y_Hypersigma20 == 0:
            Y_Hypermu0 = ymu0[-1]
            Y_Hypersigma20 = ysigma20[-1]
```

<Bayesian 계산>

- mu_sigma에서 계산된 avg, sigma를 이용하여 Bayesian_cal.py에서 베이지안 계산.

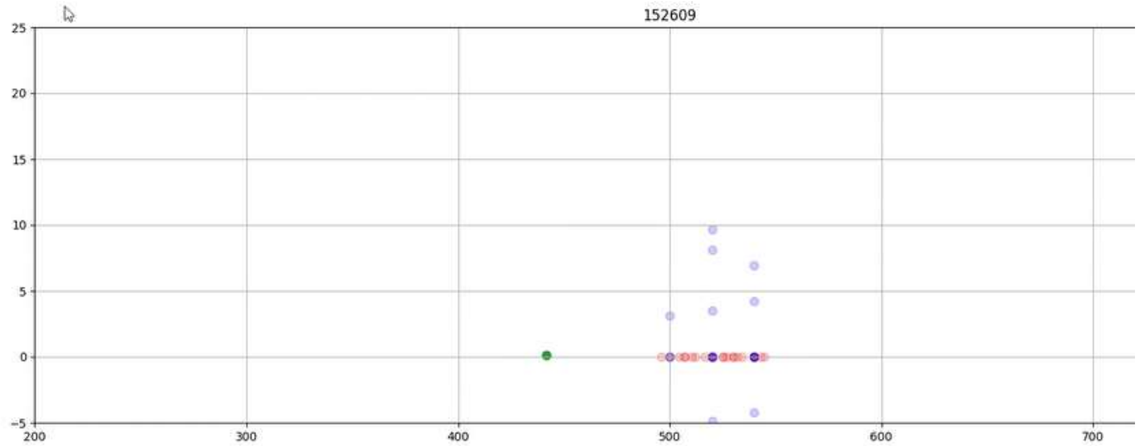


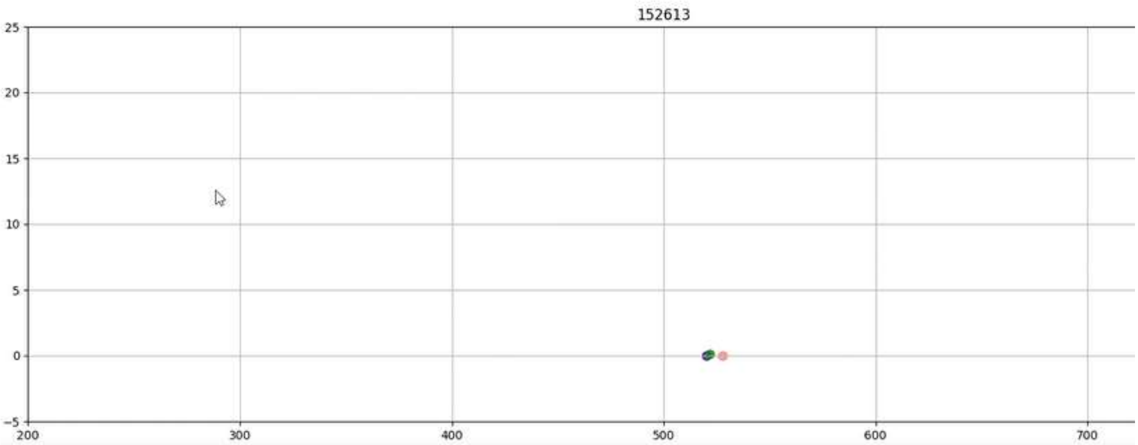

```python
q = 0
for i in range(0, rssi2distance_1.sum_count):
    # count = 0
    for j in range(2):
        plt.scatter(rssi2distance_1.intercept[3 * i + 1][j] \
                    + posvector(rssi2distance_1.intercept[3 * i + 2])[0], posvector(rssi2distance_1.intercept[3 * i + 2])[1], s=50, c='r', alpha=0.2)

        plt.scatter(posvector(rssi2distance_1.intercept[3 * i + 2])[0], \
                    rssi2distance_1.intercept[3 * i + 1][j + 2] + posvector(rssi2distance_1.intercept[3 * i + 2])[1], s=50, c='b', alpha=0.2)

    plt.title(rssi2distance_1.intercept[3 * i])
    # print('Check')
    # print(rssi2distance.intercept.index(rssi2distance.intercept[3 * i]))
    # print(rssi2distance.count.index(rssi2distance.count[i]))
    if (rssi2distance_1.intercept[3 * i] != rssi2distance_1.intercept[3 * (i + 1)]):  # 현재 시간과 다음 시간이 다르다면.
        # for q in range(0, mu_sigma.lenresultsum, 6):
        plt.scatter(Bayesian_cal.Bayesian_result[q + 1] \
                    + posvector(rssi2distance_1.intercept[3 * i + 2])[0], posvector(rssi2distance_1.intercept[3 * i + 2])[1], s=50, c='g', alpha=0.8)
        plt.scatter(posvector(rssi2distance_1.intercept[3 * i + 2])[0], \
                    Bayesian_cal.Bayesian_result[q + 2] + posvector(rssi2distance_1.intercept[3 * i + 2])[1], s=50, c='g', alpha=0.8)
        # plt.scatter(mu_sigma.result_summary[q + 1] \
        #     +  posvector(rssi2distance.intercept[3 * i + 2])[0], posvector(rssi2distance.intercept[3 * i + 2])[1], s=50, c='g', alpha=0.8)
        # plt.scatter(posvector(rssi2distance.intercept[3 * i + 2])[0],\
        #     mu_sigma.result_summary[q + 2] + posvector(rssi2distance.intercept[3 * i + 2])[1], s=50, c='g', alpha=0.8)
        q = q + 3

        plt.pause(0.1)
        plt.clf()
        plt.grid()
        plt.xlim([200, 800])
        plt.ylim([-5, 25])
```

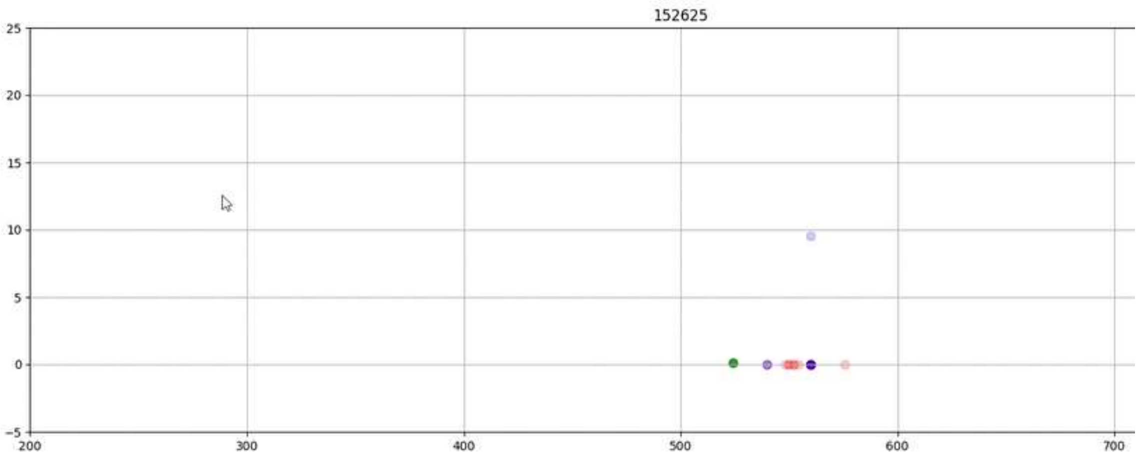
<Plotting>

- Bayesian_plotting.py는 아폴로니우스 원과 베이지안을 통해 얻은 점을 같은 시간에 대해서 plotting.

3. 동작 예시


152609
25
20
15
10
5
0
−5
200
300
400
500
600
700

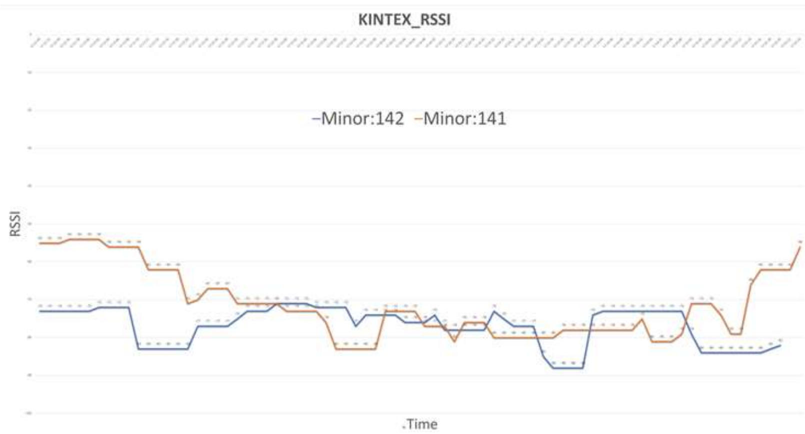

152613
25
20
15
10
5
0
−5
200
300
400
500
600
700


152625
25
20
15
10
5
0
−5
200
300
400
500
600
700

4. 해결해야 할 것.

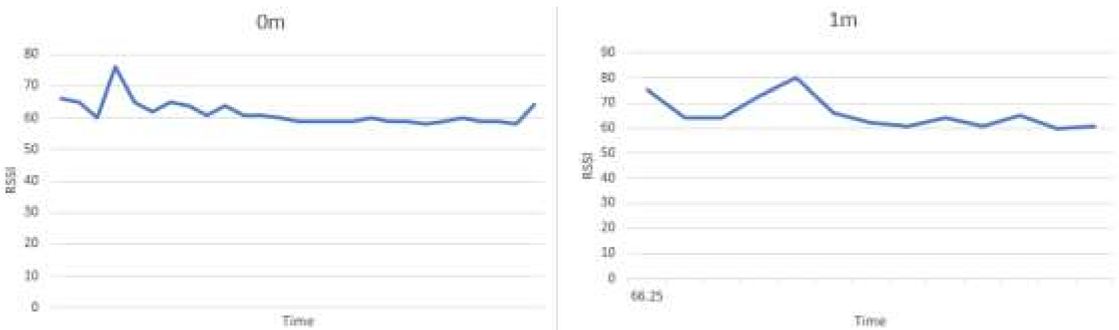
Case 1.

<Vestella 앞 복도 RSSI Data>

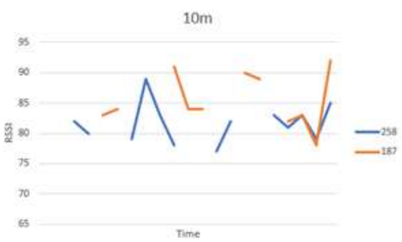


<KINTEX RSSI Data>

- 이웃한 두 신호를 이용하여 베이지안 사용
  Flat한 부분이 신호를 받지 못하는 경우. 이 때 이전 신호를 최신 신호로 업데이트



<가까운 거리에서의 RSSI 수신 그래프>



<먼 거리에서의 RSSI 수신 그래프>

- 가까운 거리에는 RSSI신호 수신이 양호.
- Beacon과 Beacon 사이의 거리가 20m일 때 중간 지점에 갈수록 신호를 받지 못하는 경우가 多.
- Test 폰에 따라서 수집되는 Data의 양의 차이 大.
- 수신이 좋지 못한 핸드폰은 신호를 받지 못하거나 한 개의 신호만 계속 받아와 Data를 사용할 수 없음.
- 수신이 좋은(최신 폰) 핸드폰을 사용할 경우 중복되는 Data 多

Case 2.

152610	[8.749105729401052, False, False, False]	153
152610	[False, -7.521517979604354, False, False]	153
152610	[9.373050468158027, False, False, False]	153
152610	(-10.0, False, False, False)	153
152610	[False, -8.749105729401052, False, False]	155
152610	[9.373050468158027, False, False, False]	153
152610	[9.373050468158027, False, False, False]	153
152610	[False, -8.131048082198735, False, False]	153
152610	(10.0, False, False, False)	153
152610	[9.373050468158027, False, False, False]	151
152610	[False, -8.131048082198735, False, False]	155
152610	[False, -8.131048082198735, False, False]	153
152610	(-10.0, False, False, False)	155
152610	[9.373050468158027, False, False, False]	151
152610	[False, -8.131048082198735, False, False]	155

<Figure 1>

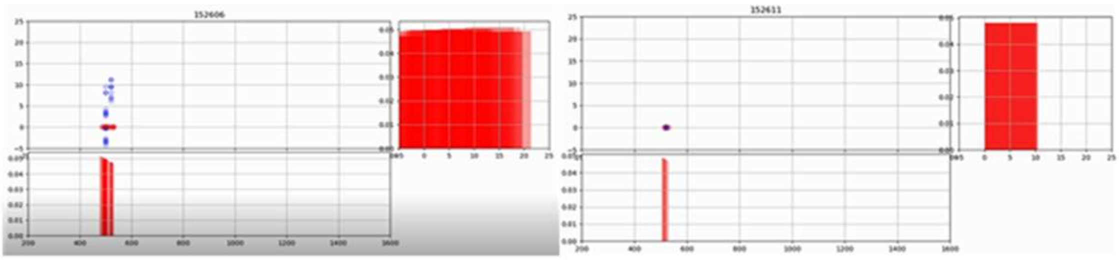
- 아폴로니우스 원 절편값이 <Figure 1>과 같을 때

152610, [(528.7491057294011, 0)],
152610, [(512.4784820203956, 0)],
152610, [(529.373050468158, 0)],
...
152610, [(530.0, 0)],
152610, [(509.373050468158, 0)],
152610, [(531.8689519178013, 0)]
y값이 모두 0일 때 avg와 std가 모두 0.

$$\frac{1}{\sigma_0'^2} = \frac{1}{\sigma_0^2} + \frac{N}{\sigma^2}$$

ZeroDivisionError 발생.

Case 3.



- Data가 얼마나 응집해있나 보여주기 위한 Histogram plot이 필요.
- 또한, Bayesian의 결과에 신뢰성을 높이기 위해 필요.

참고차료.

아폴로니우스 원 구하는 공식
https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=brian0409&logNo=60212380661
https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=10baba&logNo=220727889661
https://jwmath.tistory.com/98

베이지안 이론
Choi Yung Ji. "RnD_Report_PositioningCalibration." 210521. Bayesian Estimation.