IPIN Track 7 report

2020/12/09

NienTing Lee

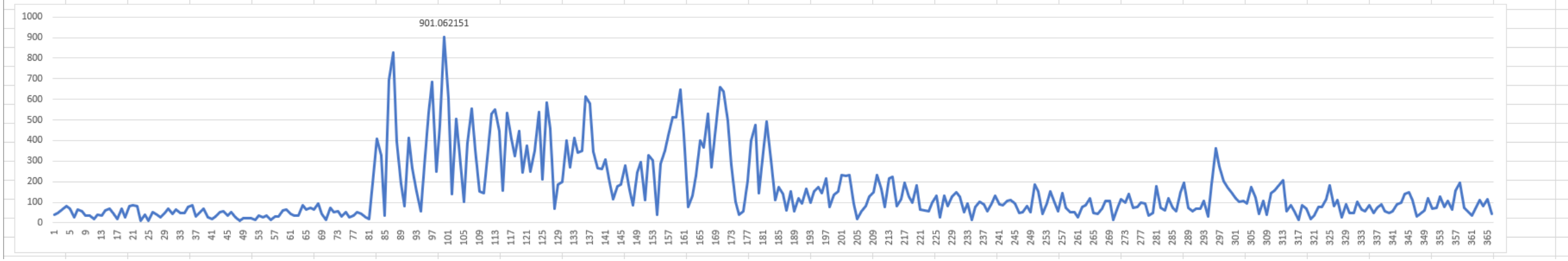
Outline

- CIR Signal processing
- Labeling
- Model building and training
- 6 receiver model training results
- Each receiver model predicts location voting
- The accuracy of the answer after voting
- Output prediction results as answer format

For CIR real 、 imaginary processing

Root the square sum of the CIR signal

cir_real	36	-6	16	79	30	-18	-12	-36	-20	19	2	-31	-30	-55	-12	20	-5	-64	-15	23	18	37	-6	20
cir_imag	15	46	62	23	61	21	65	46	30	29	-20	-24	-17	24	67	44	20	-20	22	80	84	72	8	32
	39	46.38965	64.03124	82.28001	67.97794	27.65863	66.09841	58.41233	36.05551	34.66987	20.09975	39.20459	34.48188	60.00833	68.06614	48.33218	20.61553	67.05222	26.62705	83.24062	85.90693	80.9506	10	37.73592



For labeling

- ref_x + (ref_y * 15) - 80
- ref_x : 5-19 (15)
- ref_y : 5-23 (19)
- cir_label total classes 285 (from 0)

For labeling

ref_y ref_x	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
5	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
7	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44
8	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
9	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74
10	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89
11	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104
12	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119
13	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134
14	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149
15	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164
16	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179
17	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194
18	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209
19	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224
20	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
21	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254
22	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269
23	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284

Model building and training

```
cir_train, cir_test, cir_label, cir_target = train_test_split(cir_train, cir_label, test_size=0.2)

# build the model
model = models.Sequential()
model.add(Dense(512, input_shape=(366,)))
model.add(Dense(256))
model.add(Dense(256))
model.add(Dense(128))
model.add(Dense(128))
model.add(Dense(64))
model.add(Dense(64))
model.add(Dense(285, activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

result = model.fit(cir_train, cir_label, epochs=300, batch_size=128)

model_name = str(id) + '_model_v4.h5'
models.save_model(model, model_name)
```

79:95:4D:56:05:84, Test accuracy : 0.34429853075022754

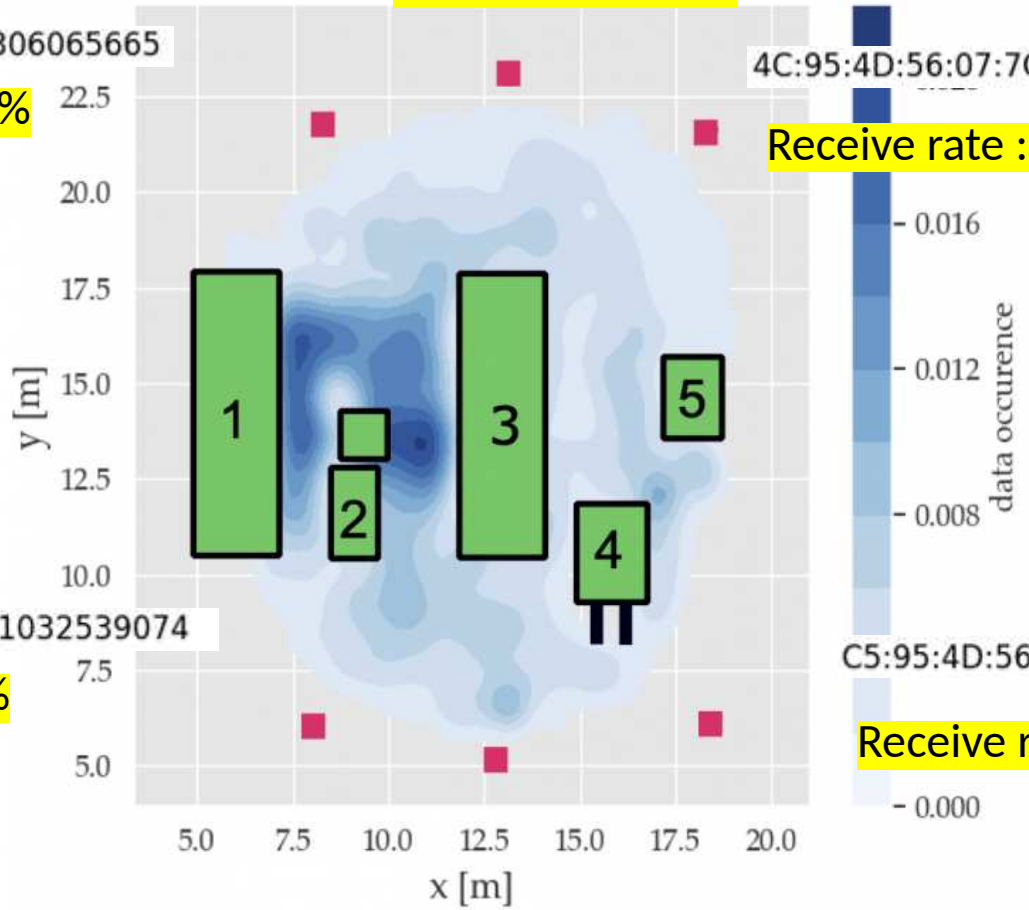
Receive rate: 84%

11:96:4D:56:02:D6, Test accuracy : 0.276850306065665

Receive rate : 80%

4C:95:4D:56:07:7C, Test accuracy : 0.2935425778235143

Receive rate : 85%



A7:95:4D:56:08:81, Test accuracy : 0.34844991032539074

Receive rate : 89%

C5:95:4D:56:02:D6, Test accuracy : 0.3273615635179153

Receive rate : 90%

Receive rate : 88%

D5:95:4D:56:01:67, Test accuracy : 0.3133727512894704

Each receiver model predicts location voting

```
for i in range(df.shape[0]):
    if df.burst_id.values[i] == df.burst_id.values[i - 1] or i == 0:  # 當它為 相同burst_id 或 第0個burst_id
        buffer.append(df.predict_result.values[i])
        if df.cir.values[i].max() > cir_max:  # 當投票無結果，就「哪一個接收器cir_max最大，就取誰的預測答案。」
            vote_err_ans = result_to_xy(df.predict_result.values[i])
            cir_max = df.cir.values[i].max()
    else:  # 換下一個burst_id之前
        vote_result = Counter(buffer)
        top2 = vote_result.most_common(2)
        if len(top2) != 1:
            if top2[0][1] == top2[1][1] or top2[0][1] == 1:  # 如果投票結果兩個平手 or 全部平手
                x_est.append(vote_err_ans[0])
                y_est.append(vote_err_ans[1])
            else:  # 取得最多model猜的位置存入
                x, y = result_to_xy(top2[0][0])
                x_est.append(x)
                y_est.append(y)
        else:  # 全部model猜一樣
            x, y = result_to_xy(top2[0][0])
            x_est.append(x)
            y_est.append(y)
```

Tie situation in voting

260820	778907799	9.78594	16.33470	4C:95:4D:56:07:7C	22824	156	6654.99579
361297	778907920	9.78594	16.33470	A7:95:4D:56:08:81	22824	184	4296.70758
464402	778907994	9.78594	16.33470	D5:95:4D:56:01:67	22824	168	6413.46895
213570	778922373	9.78563	16.32122	11:96:4D:56:02:D6	22824	170	4087.00783
412551	778928427	9.78525	16.30783	C5:95:4D:56:02:D6	22824	97	5914.53937

rec_time	ref_x	ref_y	anch_id	burst_id	predict_result
0	13.13915	6.51588	A7:95:4D:56:08:81	14264	38
13147	13.13932	6.51596	D5:95:4D:56:01:67	14264	38
13356	13.13932	6.51596	4C:95:4D:56:07:7C	14264	38
13433	13.13932	6.51596	79:95:4D:56:05:84	14264	38
14882	13.13932	6.51596	11:96:4D:56:02:D6	14264	38
15774	13.13950	6.51600	C5:95:4D:56:02:D6	14264	38

Results in voting

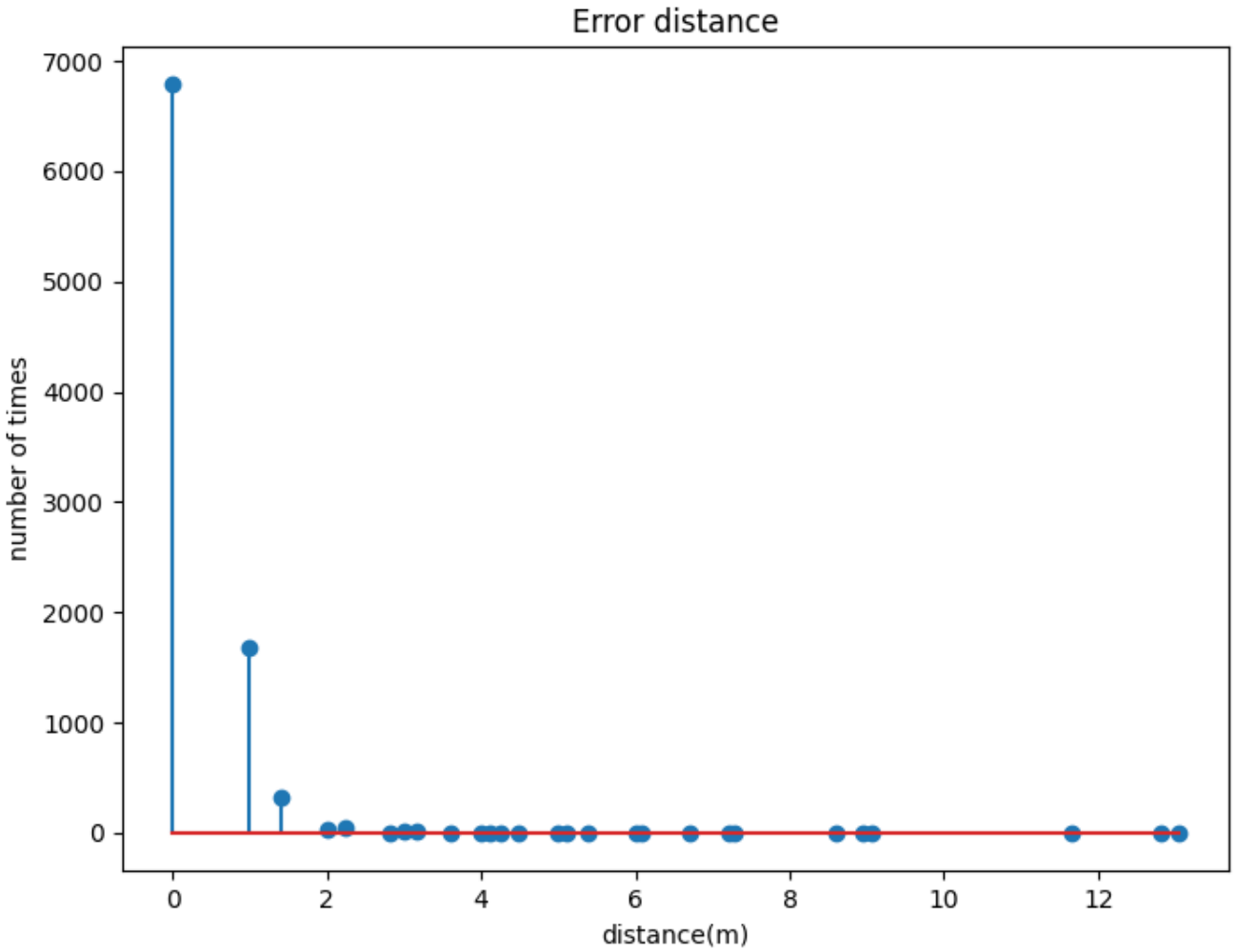
159471813	10.44962	6.69915	79:95:4D:56:05:84	16017	36
159471935	10.44962	6.69915	4C:95:4D:56:07:7C	16017	56
159472009	10.44962	6.69915	A7:95:4D:56:08:81	16017	36
159474394	10.44962	6.69915	D5:95:4D:56:01:67	16017	20
159503485	10.45352	6.67473	11:96:4D:56:02:D6	16017	47
159536181	10.45498	6.66227	C5:95:4D:56:02:D6	16017	38

6 receivers vote after training

Accuracy : 0.7579241071428572

Missing point : 2169

Total estimate : 8960



Output prediction results as answer format

- the timestamps `t_est` [`int`] of the position estimates in μs (starting from 10,000).
- the corresponding estimated positions `x_est` and `y_est` as float

```
        x_fill = np.linspace(x_f, df.x_est.values[i], range_fill)
        y_fill = np.linspace(y_f, df.y_est.values[i], range_fill)
        fill_index = 0
        for j in range(range_start, range_end):
            t_est.append(j * 100)
            x_est.append(float(x_fill[fill_index]))
            y_est.append(float(y_fill[fill_index]))
            fill_index += 1
        start = df.t_est.values[i]
        x_f = df.x_est.values[i]
        y_f = df.y_est.values[i]
        print('Data processing...', i, ', fill', range_end-range_start, 'points')
    else:
        print('Data processing...', i, ', time error !')
```

Output prediction results as answer format

- the timestamps `t_est` [
  10,000).
- the corresponding estir

```
    x_fill = np.linspace(x
    y_fill = np.linspace(y
    fill_index = 0
    for j in range(range_s
        t_est.append(j * 1
        x_est.append(float
        y_est.append(float
        fill_index += 1
    start = df.t_est.value
    x_f = df.x_est.values[
    y_f = df.y_est.values[
    print('Data processing
else:
    print('Data processing
```

1565237	156533700	10.00000	13.00000
1565238	156533800	10.00107	13.00214
1565239	156533900	10.00214	13.00428
1565240	156534000	10.00321	13.00642
1565241	156534100	10.00428	13.00857
1565242	156534200	10.00535	13.01071
1565243	156534300	10.00642	13.01285
1565244	156534400	10.00749	13.01499
1565245	156534500	10.00857	13.01713
1565246	156534600	10.00964	13.01927
1565247	156534700	10.01071	13.02141
1565248	156534800	10.01178	13.02355
1565249	156534900	10.01285	13.02570
1565250	156535000	10.01392	13.02784
1565251	156535100	10.01499	13.02998
1565252	156535200	10.01606	13.03212