高精度測位チャレンジ発表

東京海洋大学　情報通信工学研究室
岡田悠聖　安藤大貴　福原安滋

大学授業での取り組み

- 2023年のデータを利用してRTKLIBとRTKLIBexplorerで解析をした経験がある。
- その際にマスク角や信号レベルその他のパラメータを調整して最適な設定を見つける課題に取り組んだ。
- 2023年の東京データと名古屋データである程度のパラメータは把握していた。
- 後藤先輩がそのパラメータで提出したが、スコアが40前半だった。
- 我々のチーム（学部生3名）は研究室のソフトで取り組んだ。

目次

RTKコアの紹介

高精度測位チャレンジ内での工夫点

総括・今後の展望

解析に用いたRTKコアの紹介

対応する衛星：GPS/GLONASS/Galileo/BeiDou/QZSS
開発環境：Microsoft Visual Studio C/C++（基本C）

内部演算：一部カルマンフィルタを用いているが、基本は最小
2乗法で測位演算を実施

利用した測位手法：RTKとDGNSSのみ（一部速度ベクトル利用）

2018年頃のCQ出版社の雑誌で一度紹介されており、基本的な部分は全く同じ。BDSの衛星数が増えたりで読み込み部分を改修してきた。最近の改良ポイントは以下の論文で発表（詳細はこちら）

Fredeluces, E.; Ozeki, T.; Kubo, N.; El-Mowafy, A. Modified RTK-GNSS for Challenging Environments. *Sensors* **2024**, *24*, 2712. https://doi.org/10.3390/s24092712

RTKの概要

- 整数アンビギュイティ決定はRTKLIBのlambda.cをそのまま利用している
- LAMBDA法で得た整数解を受け入れるかどうかはRatioテストを利用
- Ratioテストは最終的に整数アンビギュイティを求める段階での利用衛星数に応じて閾値を変化させた（以下の通り）

```
if (SATndual[rcvn] >= 20)
Ratio_limit = 1.5;
if (SATndual[rcvn] >= 15 && SATndual[rcvn] < 20)
Ratio_limit = 2.0;
if (SATndual[rcvn] >= 10 && SATndual[rcvn] < 15)
Ratio_limit = 2.5;
if (SATndual[rcvn] < 10)
Ratio_limit = 3.0;
```

高精度測位チャレンジにおける解析手法の工夫点

0．利用した測位衛星と周波数

1．各コースの最適なマスク角、信号強度を調整

2．　擬似距離とドップラー周波数の残差を確認

3．Velocity積分によるfloat解の調整

4．高度情報の活用

途中から名古屋データを中心に解析をすることが多かった。
東京よりも名古屋データの結果が良くなかった。

利用した測位衛星と周波数

- GPS,GLONASS,GALILEO,BDS,QZSS
- 2周波観測値を同時利用のみ（rover:x5受信機）

  GPS,QZSS：C1CとC2L

  GLONASS：C1CとC2P

  GALILEO：C1CとC5Q

  BDS：C2IとC6I
- 基準局はトリンブル受信機なのでGPSとQZSSを念のため分けた（二重位相差の基準衛星を別とした）

2．各コースの最適なマスク角、信号強度へ調整

```
#RTK coreをinitial settingを変更して繰り返し実行する
#変更する項目はMask_angleとThreshold_cn

import numpy as np
import subprocess
```

[9] ✓ 0.0s

```
# ファイルを開く
with open('initial_setting.txt', 'r') as file:
    # ファイルの各行をリストに格納
    ini_setting_lines = file.readlines()
print(ini_setting_lines)
Mask_angle=np.arange(15,32.5,2.5)#仰角マスクの設定　開始仰角、終了仰角、step
Threshold_cn=np.arange(30,40.5,0.5)#CNマスクの設定　開始CN、終了CN、step
```

[10] ✓ 0.0s

```
['Mask_angle    15.0\n', 'Rov_obs       tokyo3\\\\rover.obs\n', 'Ref_obs       tokyo3\
```

```
num=1#繰り返し回数
logfile=open("autorun_log.txt",'w')#実行設定ログファイル
logfile.write("Num,Mask_angle,Threshold_cn\n")
print("total_num="+str(len(Mask_angle)*len(Threshold_cn)))
for ele in Mask_angle:#仰角ループ
    ini_setting_lines[0]="Mask_angle    "+str(ele)+"\n"
    for cn in Threshold_cn:#cnループ
        ini_setting_lines[10]="Threshold_cn  "+str(cn)+"\n"
        ini_setting_lines[18]="Num           "+str(num)+"\n"
        with open('initial_setting.txt', 'w') as file: #initial_settingを上書き
            for line in ini_setting_lines:
                file.write(line)
        runlog=str(num)+","+str(ele)+","+str(cn)
        print(runlog)
        subprocess.run("rtk.exe")#RTKプログラム実行
        logfile.write(runlog+"\n")
        num=num+1

logfile.close()
```

- Pythonを用いてマスク角と信号強度を変化させながらrtkコアをを実行するプログラムを作成
- fixの回数を計測するプログラムをつくりどの組み合わせが最適なのか求める

マスク角
15　17.5　20　22.5　25　27.5　30
7パターン

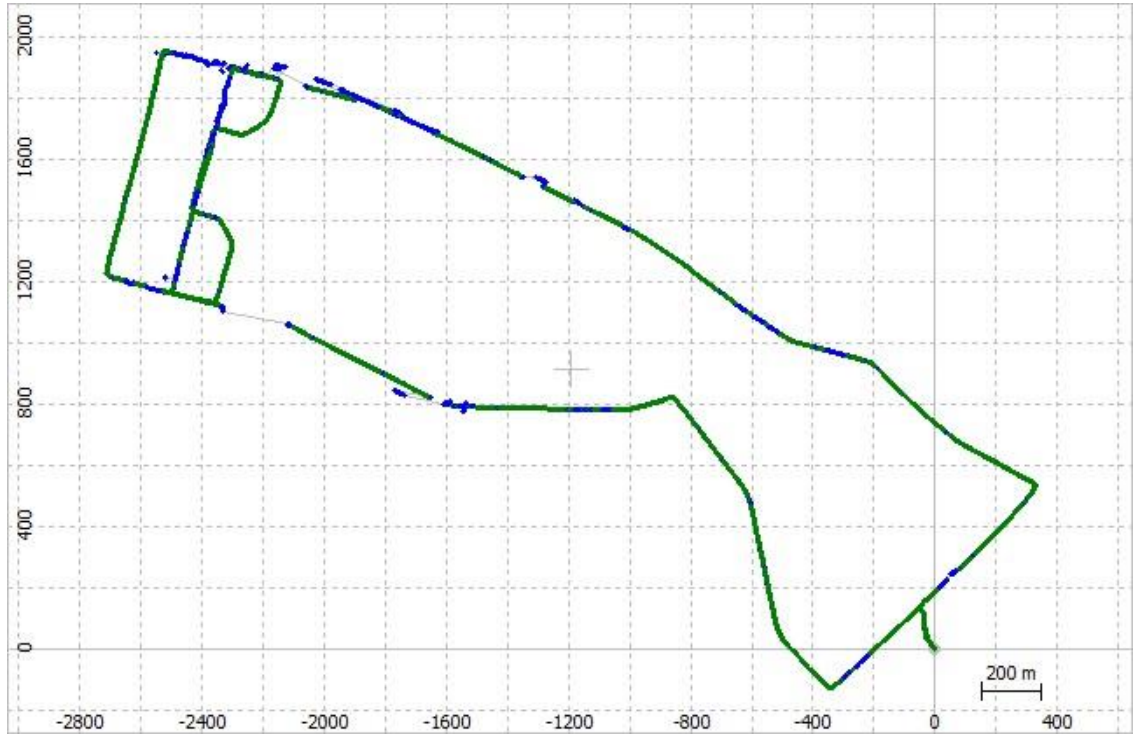
信号強度
30 30.5 31…40
21パターン

合計147通りを試行する。

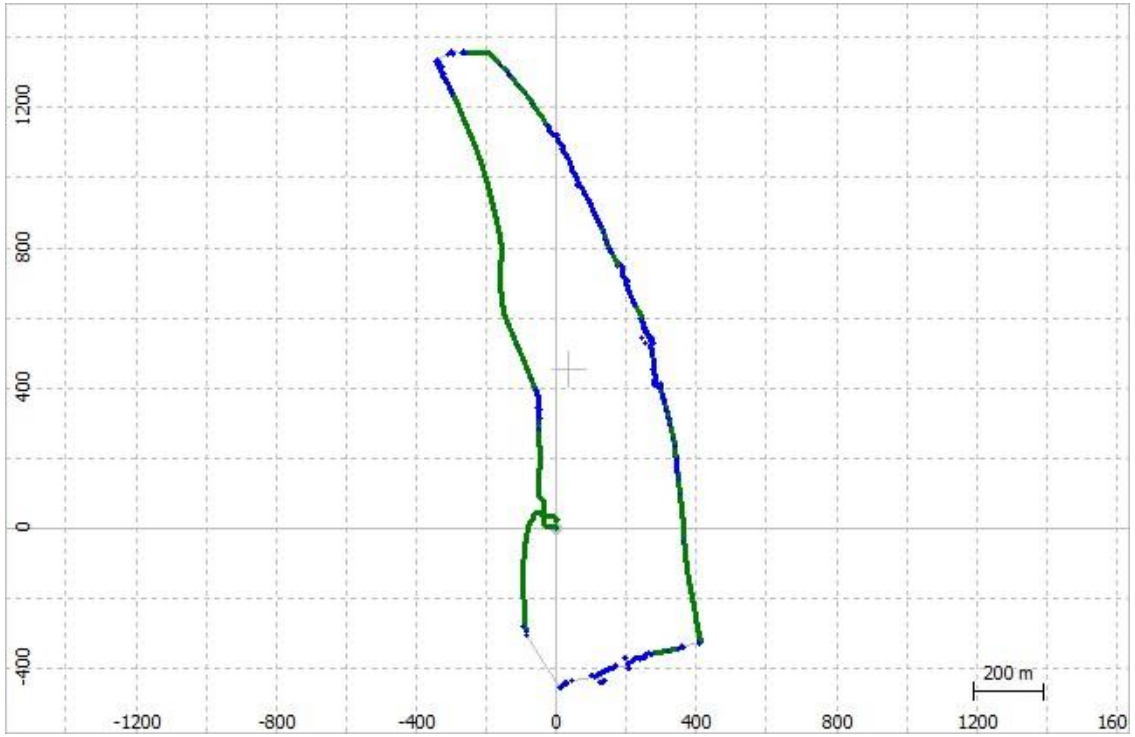
求めた最適なマスク角・信号強度

	順位	番号	mask角	信号強度			順位	番号	mask角	信号強度
名古屋1	1	68	22.5	32		東京1	1	1	15	30
	2	48	20	32.5			2	22	17.5	30
	3	28	17.5	33			3	132	30	32.5
	4	69	22.5	32.5			4	43	17.5	40
	5	49	20	33			5	127	30	30
名古屋2	1	93	25	34		東京2	1	3	15	31
	2	51	20	34			2	2	15	30.5
	3	72	22.5	34			3	22	17.5	30
	4	9	15	34			4	1	15	30
	5	30	17.5	34			5	4	15	31.5
名古屋3	1	96	25	35.5		東京3	1	45	20	31
	2	97	25	36			2	22	17.5	30
	3	117	27.5	35.5			3	46	20	31.5
	4	95	25	35			4	66	22.5	31
	5	138	30	35.5			5	23	17.5	30.5

マスク角・信号強度調整後のfix率の変化



東京1
マスク角15.0信号強度30.0
fix率 79.8%



名古屋1
マスク角22.5信号強度32.0
fix率 80.4%

・fix率を基準に調整し、fix率は向上したが**スコアへの寄与は少しだった**。
・ミスfixが多かったと考えられる。

３．擬似距離とドップラー周波数の残差を確認。測位衛星の組み合わせ

(1) 異常データの排除

・**擬似距離とドップラー周波数の残差**が大きい場合、それに対応する観測データにノイズや誤差を含んでいる可能性が大きいため、残差が一定以上の観測データを除外することで、RTK解が安定しFix率が向上する部分があった

(2) アンビギュイティ解決の支援

・動的測位環境では速度情報を利用することで、都市部での測位精度を向上させることができる。**擬似距離と速度情報による絶対位置をRTKに入力**することで、RTKが改善した

(3) 測位衛星の組み合わせ

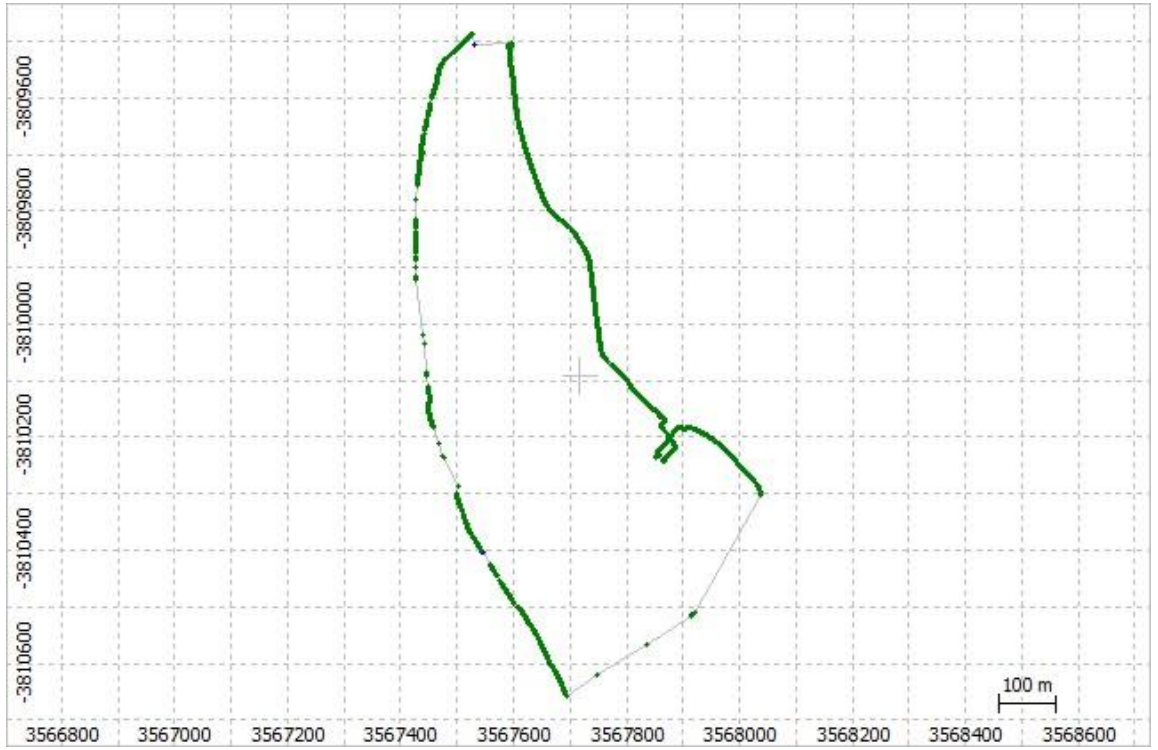
・都市部でも利用できる衛星が多いため、一つずつ衛星を排除してRTKを確認する（Partialアンビギュイティ決定）ことが可能。我々は測位衛星の組み合わせ変更で対応した。

GQEBR→GQEB→GQER→GQB→GQR→GQ

擬似距離とドップラー周波数の残差確認によるスコアの変化

東京1　　Fix率　80.8%

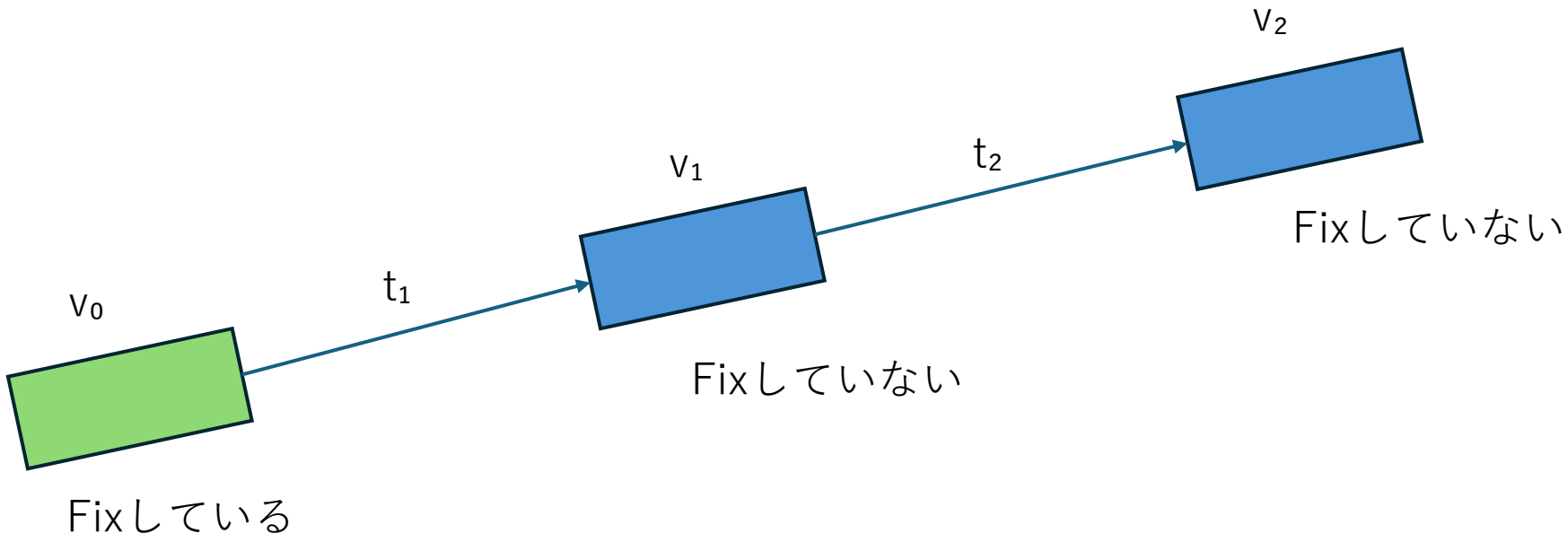
約1%改善



名古屋1　　Fix率　85.1%

約5%改善

1．Velocity積分による次エポックの位置の調整

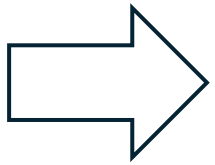


通常はfixしない場合は擬似距離によって位置を割り出している。

これを最後にfixしたときの位置と速度情報を利用することで、次のエポックの位置を予想している。

Velocity積分によるスコアの変化

今回のコンテスト内容はfix率を高めるよりも正解のルート上で高精度で測位することが重要になる。



速度による積分を何エポック分行うのか、
6コースそれぞれスコアが高くなるように調整した。

最終的に採用したもの

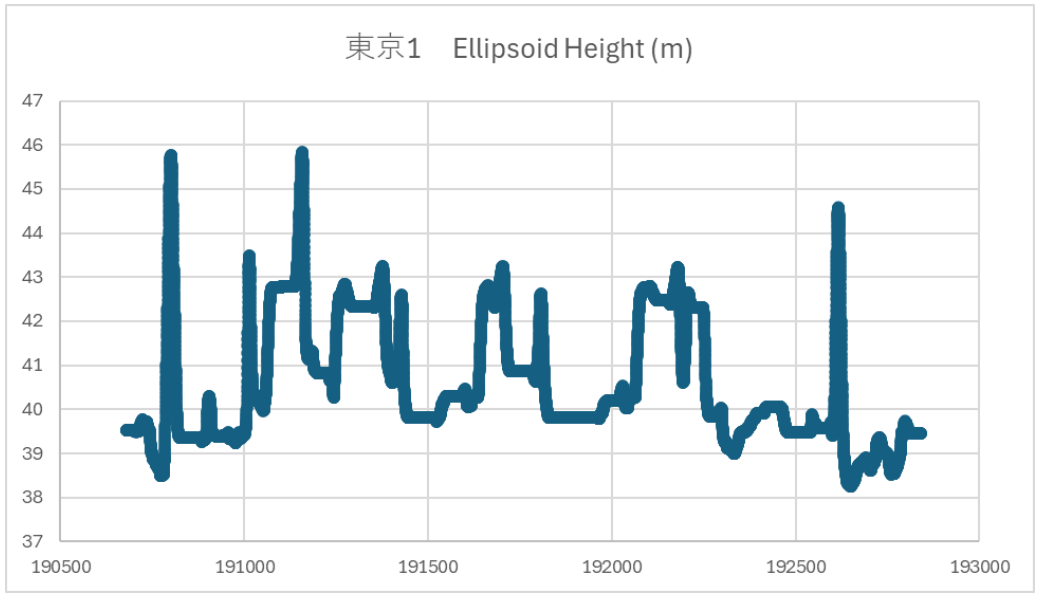
東京1　3秒（15エポック）	名古屋1　6秒（30エポック）
東京2　4秒（25エポック）	名古屋2　6秒（30エポック）
東京3　7秒（35エポック）	名古屋3　6秒（30エポック）

高度情報の活用　（今回は試すことができなかった）

(1)事前に高度の範囲（最高値と最低値）が分かっている場合、この範囲を制約条件として設定できる。これにより、誤ったアンビギュイティ決定を防ぎ、正しいFix解を向上させる効果が期待される→今回のスコアにはそれほど直結しないかもしれないが

(2) 元々、縦方向（高度）の精度が水平方向よりも劣る傾向がある。

(3)左下の図が東京のトレーニングデータの楕円体高度である。
右下の図が国土地理院より入手した今回の経路の高度情報である。
国土交通省のプラトーにも10cm程度の精度の高度情報がある。
これらを利用することで概略の水平位置が分かれば高度の範囲を設定できる。



今後の展望

先ほど紹介した高度情報と提供されていたIMUの情報を今回は使用できていないため、次はこれらのデータもプログラムに組み込むことでより測位精度を高められるのではないかと考えている。また、利用していない観測データもあり改善の余地はあると考える。

学生、社会人関係なく競い合うことができた高精度測位チャレンジアは非常に良い経験になりました。もし来年度もこのコンテストが開催されるのであれば参加したい。