CS284 HW1 Report: 2D ICP algorithm

Hongchen Cao
2019533114
caohch1@shanghaitech.edu.cn

***Abstract*—This report covers a basic description of the implementation, performance, and Q&A of the ICP algorithm.**

I. Project

A. Description of the ICP algorithm

The ICP algorithm with point-to-point error can be summarized as follow:

ALGORITHM 1: ICP

p is reference point clouds, q is the other point clouds
while not termination criteria **do**
  nearestPairs = *match*(p, q)
  nearestPairs = *reject*(p, q)
  $\hat{p} = \frac{1}{n}\sum_{i=1}^{n} p_i$, $\hat{q} = \frac{1}{n}\sum_{i=1}^{n} q_i$ # n=*len*(nearestPairs)
  $Q = \sum_{i=1} n(q_i - \hat{q})(p_i - \hat{p})^t$
  $R = VU^t$ where $SVD(Q) = U\Sigma V^*$
  $t = \hat{p} - R\hat{q}$
  q = *update*(R, t)
end while
return R, t

I use *python* to implement the algorithm and visualize the result.

B. Structure of the project

seq_frame_pics and *seq_frame_pics_noisy* contains the visualization of the results for both raw data and noisy data.

TransformMatrix and *TransformMatrix_noisy* contains the calculated transformation matrix for both raw data and noisy data.

hw1_data contains the raw and noisy data.

main.py is the implementation and visualization of the ICP algorithm.

getMatrix.py can read the transformation matrix and visualize the results for every frame.

```
d-----        2021/10/8     13:37          1   .idea
d-----        2021/9/22     13:08          1   hw1_data
d-----        2021/9/27     23:41          1   seq_frame_pics
d-----        2021/10/7      3:35          1   seq_frame_pics_noisy
d-----        2021/10/8      0:06          1   TransformMatrix
d-----        2021/10/7      3:35          1   TransformMatrix_noisy
d-----        2021/10/8     13:20          1   __pycache__
-a----        2021/10/8     13:12       2.48KB getMatrix.py
-a----        2021/10/8     13:12       8.00KB main.py
```

Fig. 1: Structure of the project

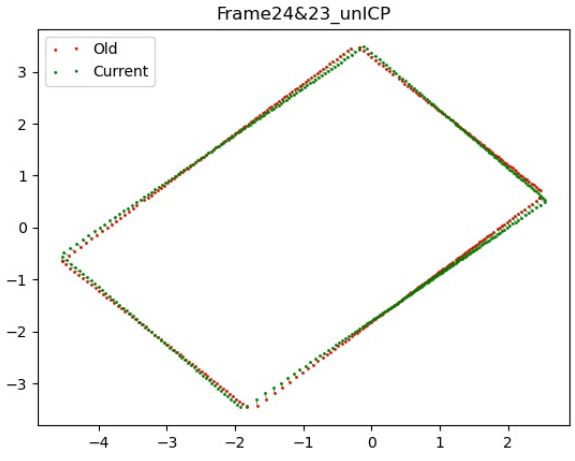
C. Dependencies

joblib $\geq$ 1.0.1
matplotlib $\geq$ 3.4.2
numpy $\geq$ 1.20.3

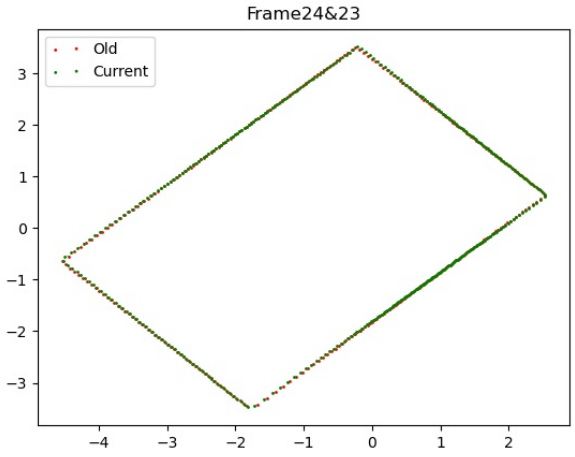
II. Performance

All the results are stored in the folder *seq_frame_pics* and *TransformMatrix*.

Following is an example for frame 24&23, the initial loss is 19.306 and finial loss is 5.680 after using ICP:



(a) UnICP



(b) ICPed

Fig. 2: Results for frame24&23

III. Q&A

Q1: What could be a straightforward way to speed up the correspondence search given that the data is coming in a very structed form?

A1: For i^{th} point in k^{th} frame, we can search t points close to it (i.e., $i-t/2^{th}$ to $i+t/2^{th}$ point) in $k-1^{th}$ frame. When $t = 360$, it becomes search all the points in previous frame.

Q2: Do you have to perform outlier filtering in our case?

A2: I tested Median, Trimmed, Fix, and None 4 outlier filtering method, their effect is almost the same. Since the environment is just a 4×6 rectangle, outlier filtering is unnecessary. However, performing outlier filtering can slightly increase the speed.

Median: $d_i < 3\times$ median
Trimmed: Remove $x\%$ worst matches
Fix: $d_i < d$, d is custom

Q3: Suitable termination criteria?

A3: I calculate loss by summing up the Euclidean distance between 2 matched points in 2 adjacent frames. Then I record last 3 loss and if $|loss[0]-loss[1|+|loss[1]-loss[2]| < 0.01$ or $loss[2] \geq loss[1] \geq loss[0]$ then terminate. The former condition represent loss decrease little and the latter condition represent loss is increasing.

I also record the iteration number, if it is greater than 30 then terminate.

Q4: Bonus questions: Compare the influence of noisy data on the ICP algorithm, what other factors will affect the results of ICP?

A4: All the results are stored in the folder *seq_frame_pics_noisy* and *TransformMatrix_noisy*. Compared to the raw data, the results for noisy data is worse and loss is larger.

I think the method of preprocessing data can affect the results of ICP, use some filters we can make the noisy data get cleaner so that the results can be more accurate.