Project 5 Report

Name: Anthony Wong
GT Email: awong307@gatech.edu
GT Username: awong307

Naive vs Numpy Transforms

1.1) Record the time per loop for your naive implementation and the time per loop for your numpy implementation. How many times faster does the numpy implementation run compared to the naive implementation?

Naive: ~211ms
Numpy: ~21.6ms

The numpy implementation is about 10 times faster than the naive implementation

```
211 ms ± 7.42 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
21.6 ms ± 1.28 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

Map with Identity Transforms (pt. 1)

1.2) Uncomment the `get_identity_results()` method to replace the transforms with identity transforms, and observe the map. Does this make sense? Why is it necessary to perform a transform to all the point clouds in order to create a map?

It does make sense that this is what we would observe with identity transforms. It is necessary to perform transforms to the point clouds because the point clouds themselves are oriented with respect to the sensor position. As the car starts turning the point cloud orientations start turning along with it. If we want a map we need to corresponding objects or points across point clouds to align with each other, thus making these objects or points static. Or else we will get a very cluttered and unreadable map.

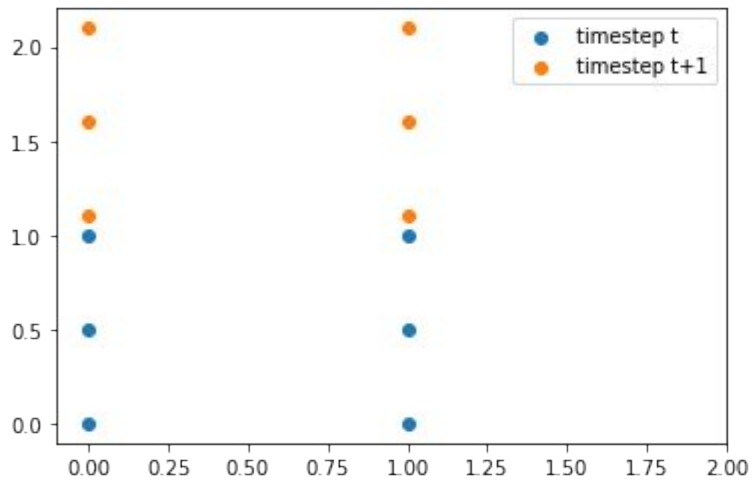
Aligning Closest Pairs Rearrangement

2.1) Why does `align_closest_pairs` return a "rearranged" cloud?

This returns a rearranged cloud because align_closest_pairs takes all the points in some cloud and aligns it with the closest point in some other cloud. This is a rearranged cloud of the latter cloud because depending on the shape, some points may have mapped to it a point, multiple points, or none at all. However, it will keep its general shape, thus rearranged.

Scan Correspondences Issue

2.2) Observe the given example scans (2D clouds). What happens when you make correspondences based on shortest Euclidean distance like in `assign_closest_pairs`? Why might this be an issue?



In this case, when we use shortest Euclidean distance we end up flattening a point cloud. This is because there are two points that are closest to all the other points in the other cloud. This means we do not maintain the proper shape of the cloud when we run assign_closest_pairs and we fail to represent the points correctly in the final map.

ICP Transform Conventions

3.1) Why is the output of ICP denoted as `bTa` and not `aTb`? Refer to Section 6.1 in the textbook for transformation conventions.

This is because our ICP implementation is trying to find the closest point in cloud a for each point in cloud b, thus finding a transformation from some pose b to some pose a. This is why it is denoted as bTa, since this follows the intended order of ICP. aTb would be the reverse.

Triangle Initial Estimate

3.2) Change the initial estimate for the ICP triangle example to be \`gtsam.Pose3(gtsam.Rot3(), gtsam.Point3(1, 1, 1)\`and evaluate the result. Play around with changing the initial estimate. What happens when your initial estimate is initialized far away from the ground truth? Why is it important to have a good initial estimate?

If our estimate is too far away from the ground truth, you increase the number of iterations needed to converge on a final/correct transformation. It is important to have a good initial estimate because the likelihood that you converge to a correct transformation within the defined maximum number of iterations. This is important both for accuracy and runtime, as you will also generally find correct transformations faster.

Implementing GTSAM

3.3) What information does a `gtsam.NonlinearFactorGraph()` hold compared to a `gtsam.Values()` object? What method do you use to access Pose3 values from a `gtsam.Values` object?

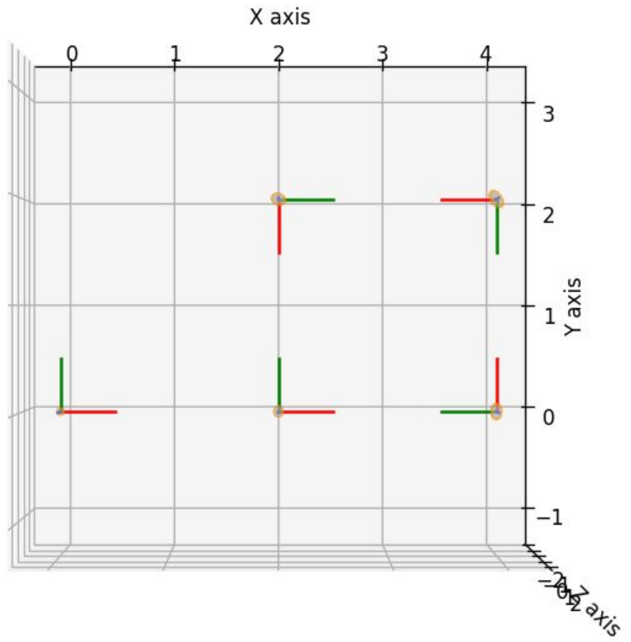
A NonlinearFactorGraph holds both variable and factor nodes in a traditional factor graph. It also holds the relative transform between any two connected poses. A Values object is more similar to a dictionary such that it holds a collections of name and value pairs. The method used to access Pose3 values from a Values object is the .atPose3() method.

Loop Closure

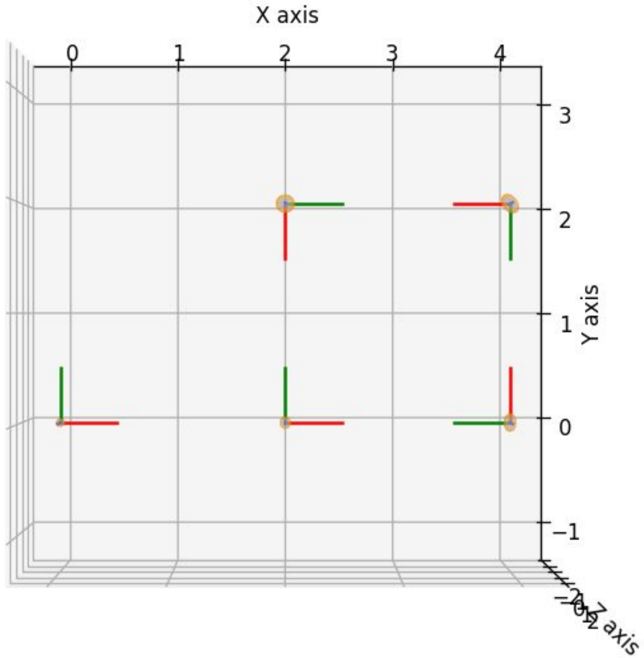
- 3.4) Comment out the loop closure constraint and upload a screenshot of the covariance plot with and without loop closure. What does covariance represent? What happens to the covariance when loop closure is performed?

Covariance represents uncertainty in the pose in any direction on the 2D plane. When loop closure is performed, the pose at (2, 2) has a slightly smaller covariance.

With:

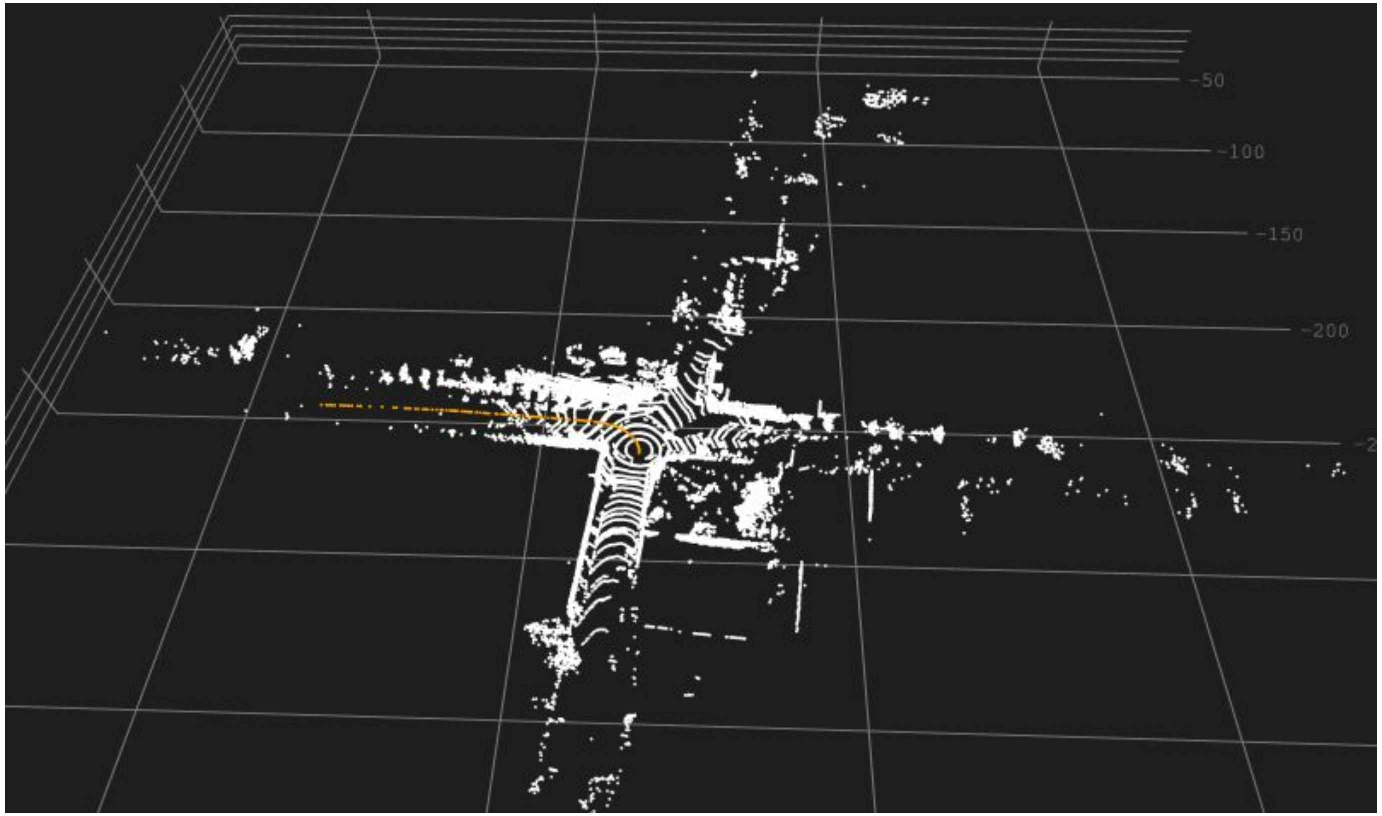


Without:



Final Vehicle Trajectory

- 4.1) Upload a screenshot of your final vehicle trajectory.



Using gtsam.Pose3.inverse()

4.2) What is the purpose of using `gtsam.Pose3.inverse()`? What happens if we do not include this?

The purpose of the inverse method is to obtain the transformation matrix that maps timestep t to timestep t + 1. If we do not include this we will have a transformation that is in reverse chronological order. This will cause our vehicle trajectory to be incorrect.

Transformation Composition (pt. 1)

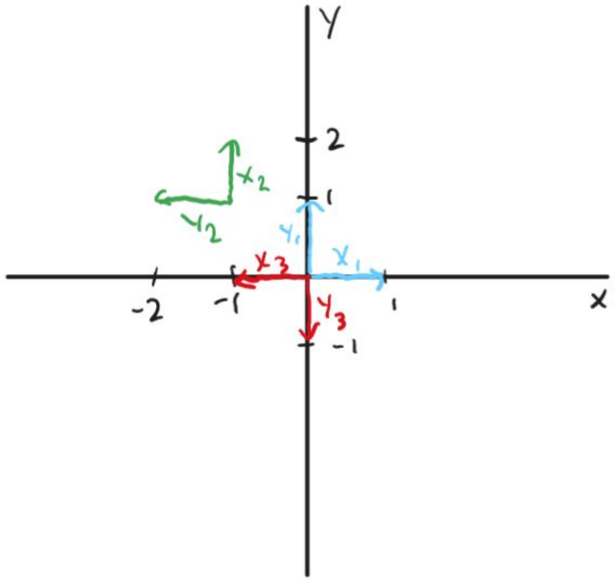
4.3) Suppose we have a robot that lives in 2D space and starts with an initial pose `gtsam.Pose2(0, 0, 0)`. The robot first performs the transform `gtsam.Pose2(1, 1, np.pi/2)` followed by a transform `gtsam.Pose2(-1, 1, np.pi/2)`.

- Draw the robot's initial pose, second pose, and final pose on a 2D coordinate grid.
- Calculate the robot's final pose using matrix multiplication. Show your work.
- Include a screenshot of calculating the robot's final pose using `gtsam.Pose2.compose()`. Does this make sense? Why?

(Answers on next slide)

Transformation Composition (pt. 2)

Part 1:



Part 2:

$$P_3 = T_1^0 T_2^1 \overbrace{P_1}^{=I^3} = \begin{bmatrix} 0 & -1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} I^3 = T_2^0 I^3 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} I^3$$

$$= \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \rightarrow \text{This is the same as Pose2}(0, 0, \pi)$$

Part 3:

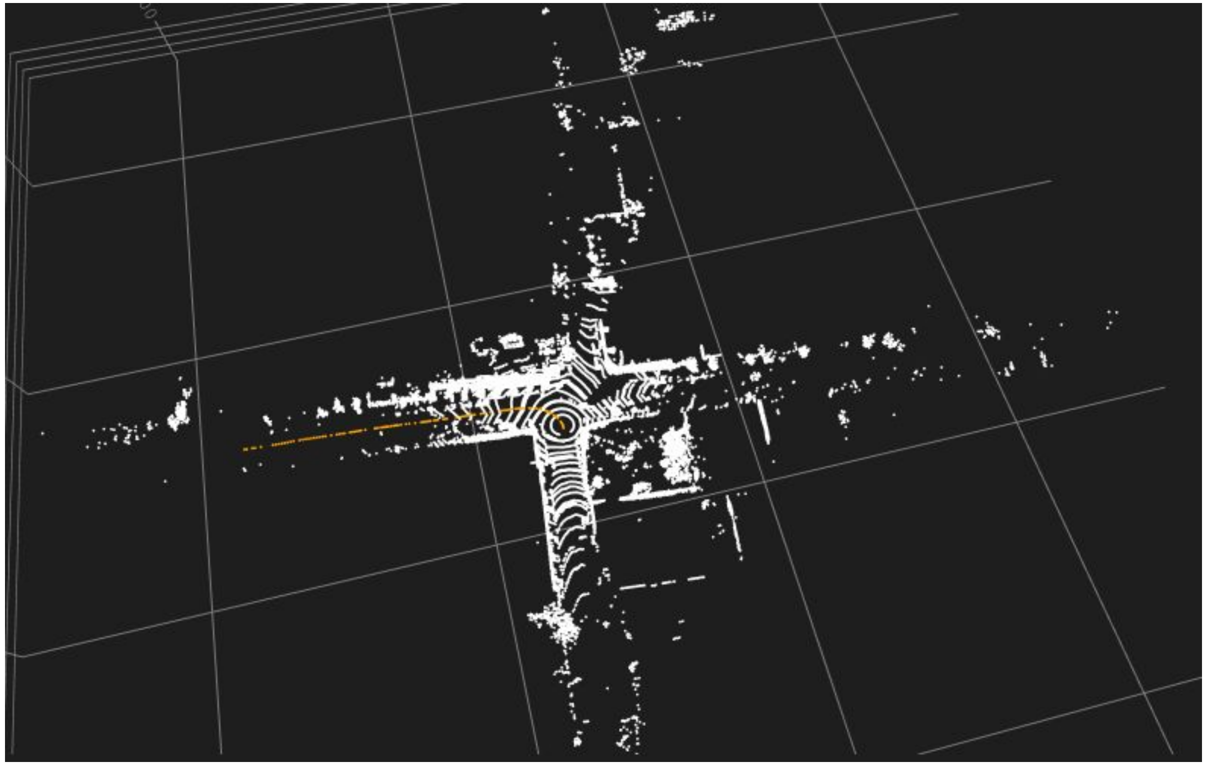
This does make sense because compose internally does matrix multiplication. If we structure the code as the picture states, we are doing the same work as in part 2. The final pose matches visually what is done in part 1 and mathematically what is done in part 2.

```
init = gtsam.Pose2(0,0,0)
t01 = gtsam.Pose2(1,1,np.pi/2)
t12 = gtsam.Pose2(-1,1,np.pi/2)
t01.compose(t12.compose(init))

(0, 1.11022302e-16, 3.14159265)
```

Skip connections (pt. 1)

5.1) Upload a screenshot of your final vehicle trajectory with skip connections. If you didn't implement this, explain the expected trajectory with skip connections.



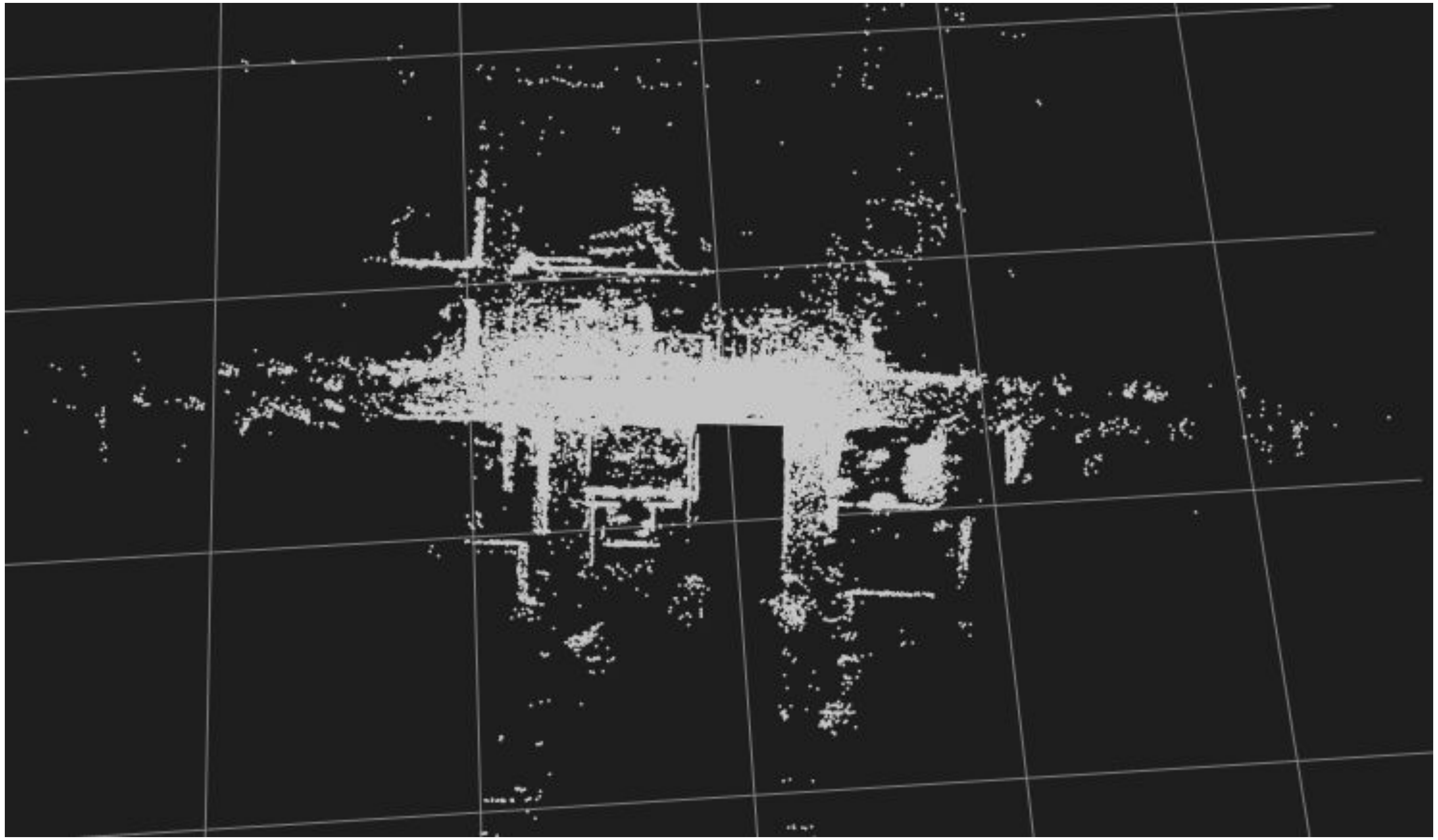
Skip connections (pt. 2)

5.2) We have access to the first 60 ground-truth poses for the car with us. Find the sum of squared differences between your pose graph output (with and without skip connections) and the ground-truth poses for the first 60 frames. Explain why you observe, what you observe (or expect to observe, if you didn't implement) in these two cases.

Without skip connections = 2.40
With skip connections = 7.05e-10
With skip connections, squared error is much lower. This occurs because we have additional connections on our factor graph. This means that there is more information during optimization and we can predict positions much more accurately.

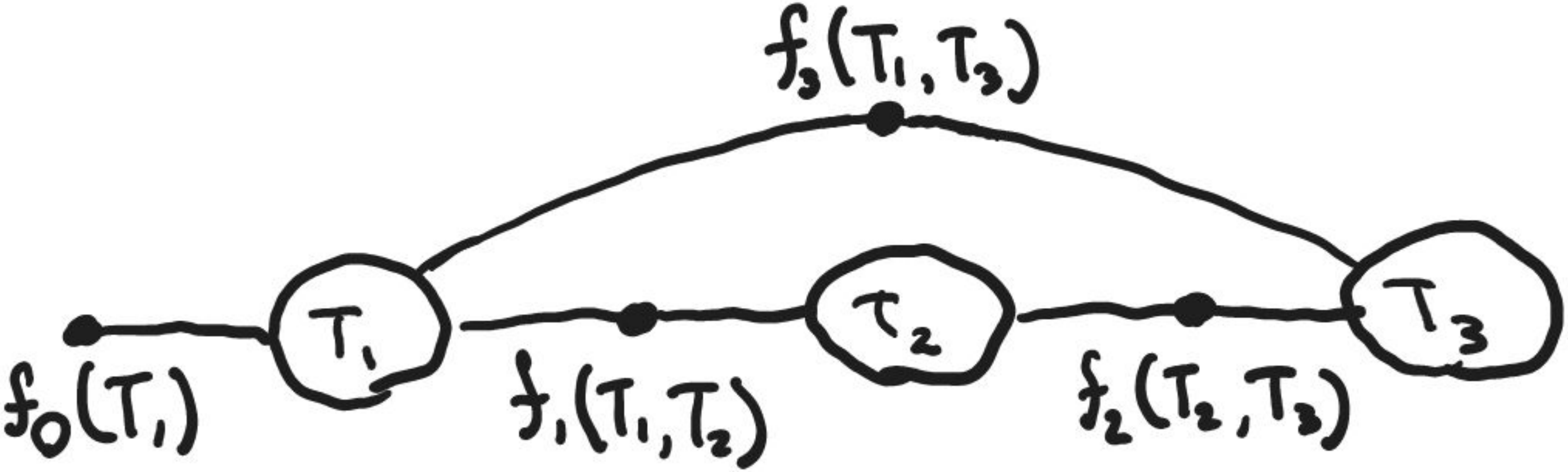
Final Map

5.3) Upload a screenshot of your final map.



Factor Graph

5.4) Draw a factor graph between three point clouds, including skip connections.



Feedback

Please provide feedback on the coding portion of the project. How did it help your understanding of the material? Is there anything that you think could’ve been made more clear?

I think the step by step parts of the project were very rewarding and simplified what could have been very confusing algorithms. I would’ve liked more explanation/an example of what every operation was doing but I understand it is also part of the report.