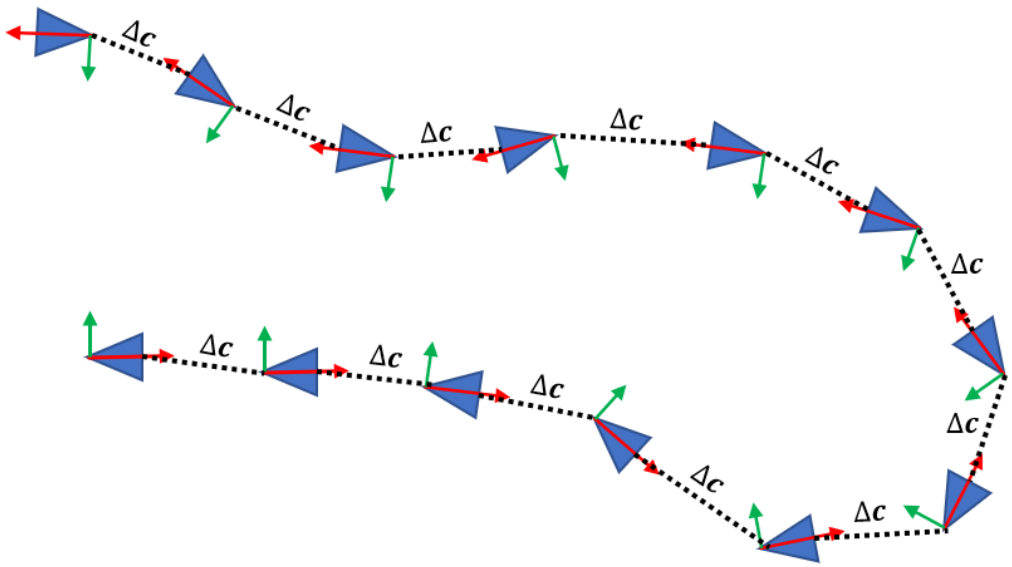
Vision Aided Navigation - Exercise 4

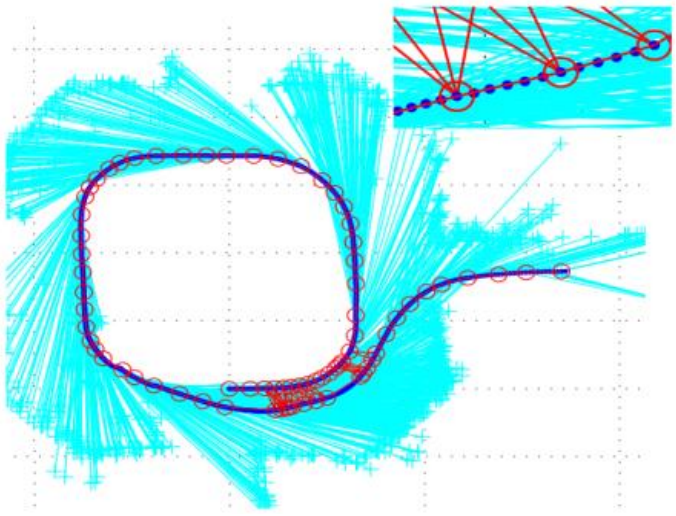
In **exercise 3** we calculated the relative poses of a subset of the trajectory frames we called keyframes.

We solved small Bundle Adjustment problems and used the results to calculate relative transformations between consecutive keyframes.
These relative poses can be used as constraints for a factor graph optimization problem we call **Pose Graph**.



Pose Graph: Relative poses Δc are used to constrain the factor graph

The pose graph is used to keep a concise summary of the trajectory. It consists of the poses of some of the frames (keyframes) and the relative nonlinear pose constraints between them. Since it contains no reprojection constraints (only pose-pose constraints) and only a subset of the frame poses it is a very concise representation of the entire trajectory.



Pose graph reduction example: A Bundle Adjustment problem with 700 frames and 100k landmarks becomes a pose graph with 133 keyframes.

Landmarks in cyan, frames in blue and keyframes circled in red.
Red edges are relative pose constraints.

4.1

Extract relative pose constraint from Bundle optimization

Use the result of the first Bundle Adjustment we got in the last exercise to estimate the relative motion between the first two keyframes and the covariance (uncertainty) of that motion.
Use marginalization and conditioning to calculate the correct covariance matrix of the relative motion (synthetic) measurement.
The covariance should be the **relative** covariance between the frame poses. i.e. the covariance associated with the distribution $P(c_k|c_0)$ with k the index of the 2nd keyframe.

- Extract the marginal covariance of the solution.
- Plot the resulting frame locations as a 3D graph including the covariance of the locations. (all the frames in the bundle, not just the 1st and last)
- Print the resulting relative pose between the first two keyframes and the covariance associated with it.

Use the results of all the Bundle Adjustment optimizations we got in the last exercise to estimate the relative motion between every two consecutive keyframes and the **relative** covariance of that motion.

4.2

Pose Graph

Build a factor graph that represents the pose graph of the keyframes. Add the relative motion estimated in the previous section as constraints to that graph with the correct uncertainty (covariance matrix) for each constraint.
Construct poses for the initial solution of the pose graph and solve it.

- What would be a reasonable initialization for the poses?
- Plot the initial poses you supplied the optimization.
- For the keyframe locations resulting from the optimization:
    - Plot the locations **without** covariances.
        - What effect did the optimization have on the locations? What is the error of the factor graph? Explain.
    - Plot the locations **with** the marginal covariances.

GTSAM

- The covariances resulting from an optimization can be extracted using:
```
marginals = gtsam.Marginals(graph, result)
```

- The marginal covariance matrix of a specific subset of the parameters can be extracted using:
```
keys = gtsam.KeyVector()
keys.append(c1)
keys.append(c2)
marginals.jointMarginalCovariance(keys).fullMatrix()
```

- Given two pose3, the relative pose between them can be calculated using:
  `relative_pose = pose_c0.between(pose_c1)`

- Relative pose factor:
  `BetweenFactorPose3(c0, c1, relative_pose, noiseCov)`

- Getting the error from the optimizer:
  `optimizer.error()`
  Can be called before or after the optimization, returns the error of the initial solution or the final result respectively.

- In order to present a trajectory with location covariances:
  ```
  from gtsam.utils import plot
  plot.plot_trajectory(1, result, marginals=marginals, scale=1)
  ```
  see documentation for the different parameters:
  http://docs.ros.org/en/kinetic/api/gtsam/html/namespacegtsam_1_1utils_1_1plot.html#a7601fd5bf0877fd02fb60270693b0b6a