ROB 530 Mobile Robotics
Winter 2023 – Homework SLAM

Yulun Zhuang, yulunz

January 30, 2023

1 2D Graph SLAM

A.

```
def load_g2o(filename):
    """
    Load a G2O file.
    Each VERTEX will be stored in "poses" ndarray and each EDGE will be stored in "
    edges" ndarray.

    Return: a data dict contain poses and edges.
    """
    poses = []
    edges = []
    path_to_file = os.path.join(DATA_PATH, filename)
    with open(path_to_file, 'r') as f:
        for line in f.readlines():
            temp = line.split()
            if temp[0][0] == "V":
                poses.append(temp[1:])
            elif temp[0][0] == "E":
                edges.append(temp[1:])
            else:
                raise NotImplementedError()

    data = {}
    data["poses"] = np.array(poses, dtype=DTYPE)
    data["edges"] = np.array(edges, dtype=DTYPE)

    return data
```

B. **Batch Solution:**

To solve a 2D pose SLAM problem in batch

1. Construct a factor graph from data. A G2O file in this case, where edges are odometry measurements and verteies are initial guess of robot poses. Use $readG2o()$ will be fast and efficient.
2. Add a prior to the pose having index zero. A diagonal noise model with zero mean and small variances can be chosen as a naive prior, a better way is using the first pose guess as the prior.
3. Create the Gauss-Newton optimizer with proper parameters.
4. Perform graph optimization and plot results.

To tune a Gauss-Newton optimizer

1. $Verbosity$ The printing verbosity during optimization (default SILENT), and set it to TERMINATION will show information about stopping conditions.
2. $MaxIterations$ The maximum iterations to stop iterating (default 100).
3. $RelativeErrorTol$ The maximum relative error decrease to stop iterating (default 1e-5).

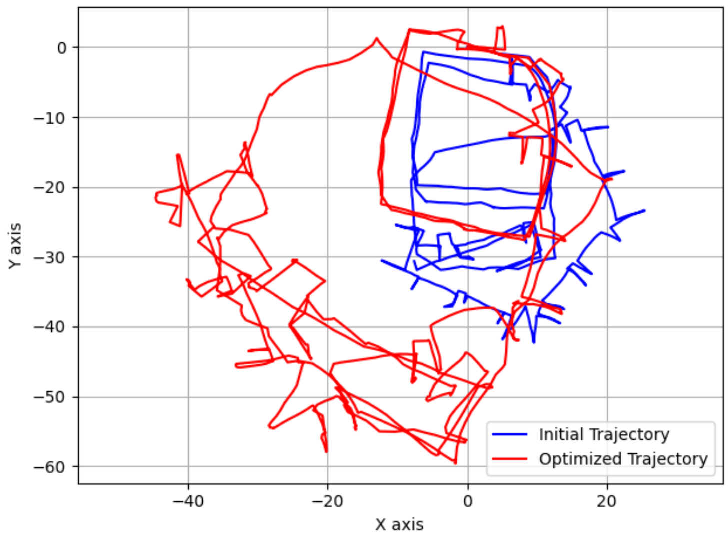


Figure 1: Batch Solution for 2D Pose SLAM on INTEL Dataset

Figure 1 shows the 2D pose SLAM result by the batch solution. It worth noting that the Gauss-Newton solver fell into a local minimum without perturbation and the optimization results are even worse than the initial trajectory. The initial error[1] is 2.575×10^6 and the final error is 1.547×10^9.

C. **Incremental Solution:**

To solve a 2D pose SLAM problem incrementally, a slightly modified version of the provided algorithm is proposed in Algorithm 1.

[1]The error is computed by the unnormalized error $0.5\Sigma_i (h_i(X_i) - z)^2/\sigma^2$ between a nonlinear factor graph and a trajectory of value nodes.

Algorithm 1 incremental_solution (poses, edges)

```
Require: poses, edges
  priorNoiseModel ← diagonal noise model
  isam ← ISAM2()                                  ▷ Initialize iSAM2 solver with proper parameters
  for each pose in poses do
      graph ← NonlinearFactorGraph()
      initialEstimate ← Values()
      (id_p, currPose) ← pose
      if id_p == 0 then
          graph.add(PriorFactorPose(0, currPose, priorNoiseModel))
          initialEstimate.insert(id_p, currPose)
      else
          prevPose ← result.atPose(id_p − 1)
          initialEstimate.insert(id_p, currPose)
          for each edge in edges do
              (id_e1, id_e2, poseBetween, infoVec) ← edge
              if id_e2 == id_p then
                  infoMat ← constructInfoMat(infoVec)
                  noiseModel ← noiseModel.Gaussian.Information(infoMat)
                  graph.add(BetweenFactorPose(id_e1, id_e2, poseBetween, noiseModel))
              end if
          end for
      end if
  end for
  isam.update(graph, initialEstimate)    ▷ Perform incremental update to iSAM2's internal Bayes tree
  result ← isam.calculateEstimate()
```

To reconstruct a information matrix from a list of its upper triangular entries

(a) Create a zero matrix A with corresponding dimensions.

(b) Loop over and assign values to its upper triangular elements.

(c) Return $A + A.T - diag(A.diagonal)$

To tune a iSAM2 solver

1. $RelinearizeThreshold$ Only relinearize variables whose linear delta magnitude is greater than this threshold (default: 0.1).
2. $MaxIterations$ Only relinearize any variables every relinearizeSkip calls to $ISAM2.update$ (default: 10).

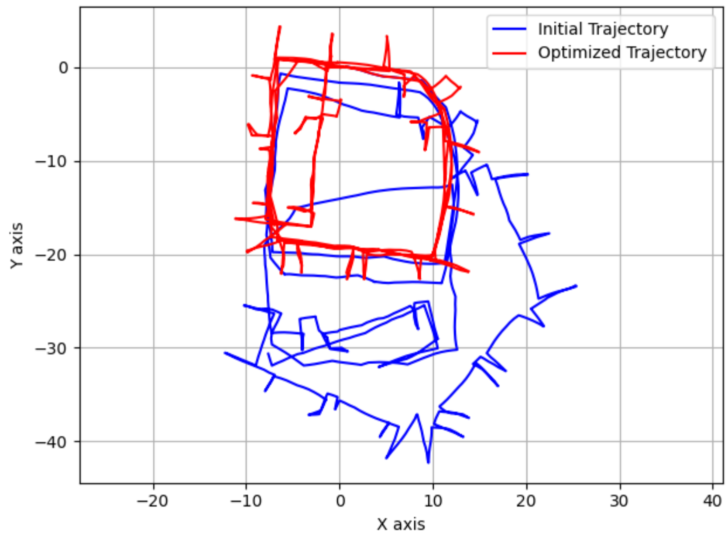


Figure 2: Incremental Solution for 2D Pose SLAM on INTEL Dataset

Figure 2 shows the 2D pose SLAM result by the incremental solution. The initial error is 2.575×10^6 and the final error is 1.319×10^6.

2 3D Graph SLAM

A. This part is the same as Section 1 A.

B. **Batch Solution:**

To solve a 3D pose SLAM problem in batch

1. Construct a factor graph from data. A G2O file in this case, where edges are odometry measurements and verteies are initial guess of robot poses. Use $readG2o()$ will be fast and efficient.
2. Add a prior to the pose having index zero. A diagonal noise model with zero mean and small variances can be chosen as a naive prior, a better way is using the first pose guess as the prior.
3. Create the Gauss-Newton optimizer with proper parameters.
4. Perform graph optimization and plot results.

To tune a Gauss-Newton optimizer

1. $Verbosity$ The printing verbosity during optimization (default SILENT), and set it to TERMINATION will show information about stopping conditions.
2. $MaxIterations$ The maximum iterations to stop iterating (default 100).
3. $RelativeErrorTol$ The maximum relative error decrease to stop iterating (default 1e-5).

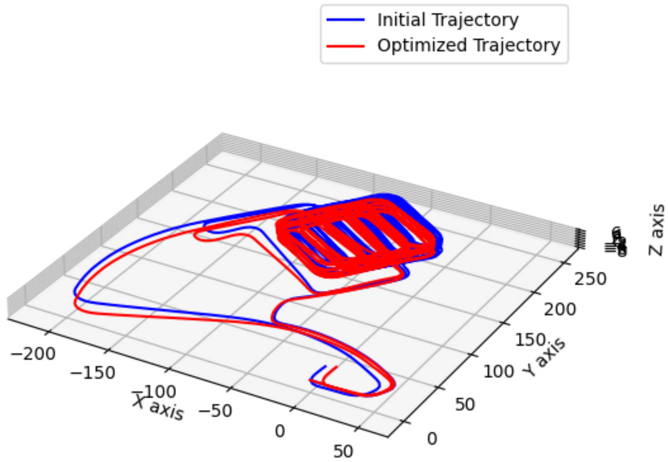
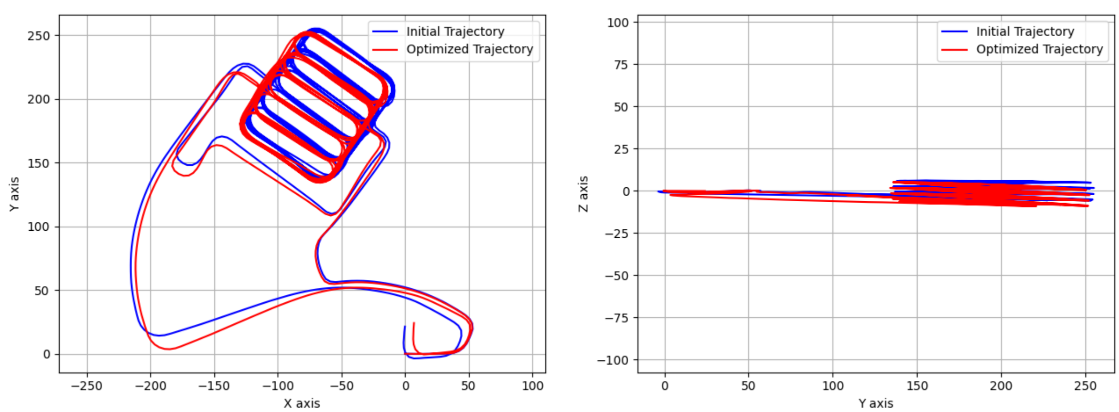


Figure 3: Batch Solution for 3D Pose SLAM on GARAGE Dataset

Figure 3 shows the 3D pose SLAM result by the batch solution. Two side views in X-Y plane and Y-Z plane of trajectories are plotted in Figure 4. The initial error is 8.364×10^3 and the final error is 0.634.



(a) X-Y View

(b) Y-Z View

Figure 4: Side Views for 3D Batch Solution

C. **Incremental Solution:**

To solve a 3D pose SLAM problem incrementally, a modified version of the provided algorithm is proposed in Algorithm 1.

To tune a iSAM2 solver

1. $RelinearizeThreshold$ Only relinearize variables whose linear delta magnitude is greater than this threshold (default: 0.1).
2. $MaxIterations$ Only relinearize any variables every relinearizeSkip calls to $ISAM2.update$ (default: 10).

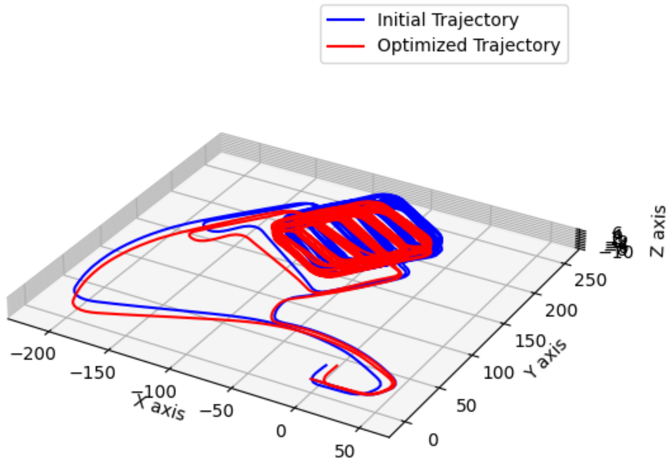
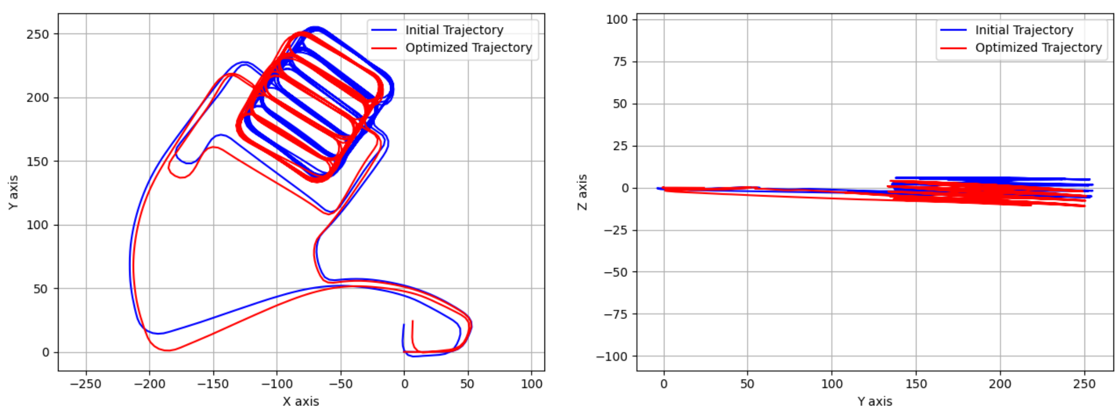


Figure 5: Incremental Solution for 3D Pose SLAM on GARAGE Dataset

Figure 5 shows the 3D pose SLAM result by the incremental solution. Two side views in X-Y plane and Y-Z plane of trajectories are plotted in Figure 6. The initial error is 8.364×10^3 and the final error is 0.756.



(a) X-Y View

(b) Y-Z View

Figure 6: Side Views for 3D Incremental Solution