Large-Scale Mapping with Loop Closure

Thomas Cohn and John Rosner

***Abstract*—We present a robotic mapping system which can produce accurate and consistent maps of large environments. Basic systems for simultaneous localization and mapping (SLAM) can estimate the motion of a mobile robot to a high degree of accuracy, allowing sensor readings from different positions to be fused into a map. But even small amounts of error can accumulate over time, especially when the robot can only observe its immediate surroundings. Our approach enables the robot to recognize that it has returned to a location it has seen before, even if the accumulated error means the map is no longer consistent. The map is then rectified with a pose graph optimization technique, producing a consistent map.**
Project video: https://youtu.be/uV2OP7v6lY0
Codebase and dataset:
https://github.com/cohnt/ICP-SLAM-with-Loop-Closure

***Keywords*—Mapping, Sensor Fusion, Optimization**

I. Introduction

ROBOTIC mapping is an essential area of robotics research. Besides the practicality of creating maps without the need for human cartographers, maps are an essential building block for many higher-level robotic functions, including localization, autonomous navigation, and mobile manipulation. For wheeled robots traveling in an indoor environment, a common task is the production of 2D maps, roughly akin to a building floor plan. Such a map is produced using a horizontally-scanning LIDAR sensor; as the robot explores the building, its sensor readings are incrementally joined into a global map of its environment.

Accurately combining sensor readings requires knowledge of the robot's motion, which is a challenging problem. Methods for directly estimating a robot's motion, including measuring wheel rotation or using inertial measurement units, quickly lose accuracy. Thus, robotic mapping systems have to estimate robot motion by aligning successive sensor readings (usually in the form of images or point clouds).

But even the most reliable scan matching techniques can still accumulate small amounts of error over time. This can lead to the map becoming inconsistent when the robot returns to a location in which it has been before. In order to build an accurate and globally-consistent map of an environment, it is necessary to detect that the robot has returned to a location it has visited before, and use that information to correct for the accumulated error and rectify the map. This process is called loop closure [1].

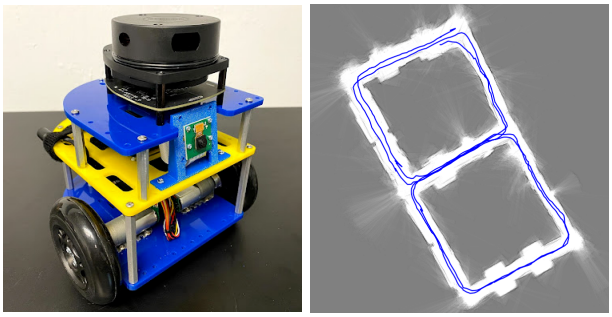


Fig. 1: The "MBot-Mini" robot used to collect the data for our mapping system (left), and an example map produced by our mapping system (right).

In this report, we present our final project for *EECS 467 Autonomous Robotics*: a complete robot mapping system. Our robot collects sensor data from a 2D scanning LIDAR and a fisheye camera. The LIDAR scans are matched together with Iterative Closest Point (ICP) [2], [3], to produce an initial map. We extract ORB keypoints [4] from each of the images, and then match keypoints between images to detect potential loop closures. We then use ICP again to determine the precise transformation of the loop closure, or discard it if the error is too high. We collect all this information into a pose graph: a data structure where each vertex is a robot pose (with its associated camera image and LIDAR scan), and edges are constraints between the poses where scans were taken. Loop closures over-constrain the graph, so we run a pose graph optimization algorithm [5] to find the solution which minimizes the error. The output from this is a globally consistent map, which can be used for downstream tasks, such as navigation.

II. Related Work

The SLAM system produced in the "Botlab" portion of EECS 467 is an excellent example of a baseline robotic mapping system. The map is structured as an occupancy grid: a probabilistic representation of the world, where each grid cell is associated with the probability that an obstacle is present. As LIDAR data is obtained by the robot, the cell probabilities which are passed over by a LIDAR beam are updated – if the beam passes through the cell, it is less likely there is an obstacle, and if the beam ends at a cell, then it must have hit an obstacle. As the map is produced, the robot is localized

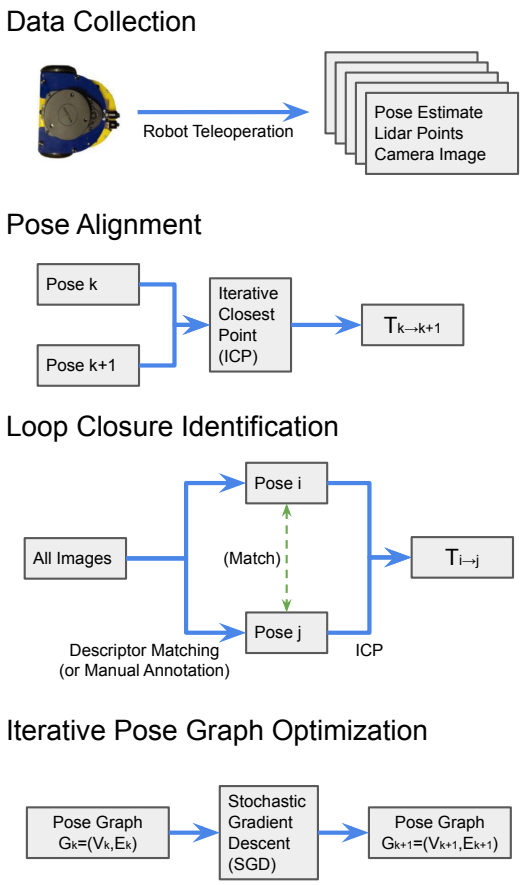


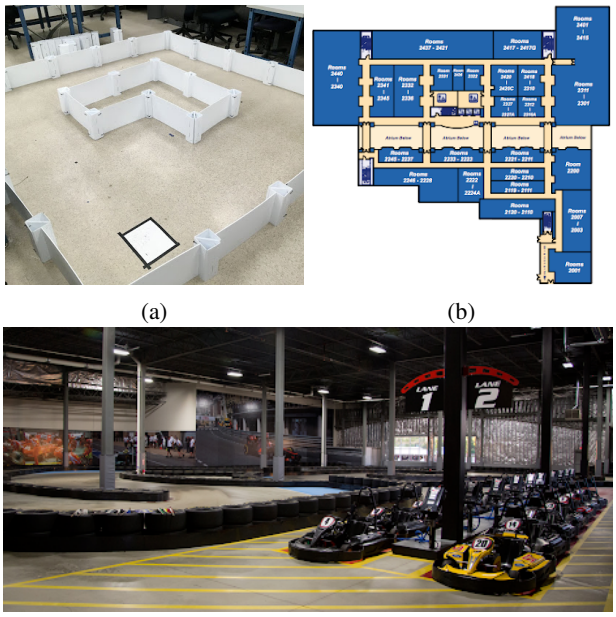
Fig. 2: A diagram of our complete robotic mapping system.

within the map using a particle filter [6]. Although this system was produced in a classroom setting, occupancy grid mapping approaches are a modern technique [7].

In contrast to occupancy grid mapping approaches, SLAM algorithms designed to work with loop closure usually represent the robot's trajectory as a pose graph [8], [3]. This graph can then be optimized to produce an accurate map [5]. Automatically identifying loop closures is a challenging problem, but can be solved by combining visual information from a camera, and world geometry from a LIDAR sensor [1].

III. Methodology

In this section, we present an exposition of our dataset, followed by a complete description of the three primary components of our mapping software: scan alignment, loop closure identification, and pose graph optimization. An overall description of our methodology is shown as a system diagram in Figure 2. At any stage of the program, the output can be stored as a pose graph, and the map



(a) (b)

(c)

Fig. 3: Three of the environments included in our dataset: A small maze environment (a), the second floor of the University of Michigan EECS Building (b), and a "go-kart" track (c).

can be exported as an occupancy grid or dense point cloud.

A. Data Collection

We obtained the data from our project by teleoperating the robot and recording all of the robot's observations for later playback. We used an "MBot-Mini" robot to collect our data. This robot is equipped with wheel encoders and an interial measurement unit; we combined information from these sensors using gyrodometry [9] to estimate robot odometry. The robot is also equipped with an RPLidar A1 2D scanning LIDAR sensor, which has a range of 0.15-12 m, a 0.5-2 mm resolution, a 2000-8000 Hz sample rate, and a 5-10 Hz scanning rate. The robot's camera is 5MP, and is equipped with a wide-angle fisheye lens.

Internal robot communication between software components is handled using Lightweight Communications and Marshalling (LCM) [10], so we record odometry and LIDAR data by storing all relevant LCM messages into an LCM log file. During the teleoperation, we also separately take pictures with the robot's camera at a rate of 1 Hz, and record the exact time every picture was taken in a separate lookup table. This allows us to associate each picture with the odometry and LIDAR data that was recorded concurrently.

We collected data from 3 different environments: small, handcrafted mazes similar to those previously used to test our "Botlab" SLAM system, the University of Michigan EECS building[1], and a local "go-kart" track. Photos of these environments are shown in Figure 3. We recorded three runs from the lab mazes, six runs from the EECS building, and two runs from the "go-kart" track. The first four runs of the EECS building were recorded on the second floor, and the final two runs were recorded on the fourth floor. Every run took a different path around the building. The two runs through the "go-kart" track were taken in opposite directions around the track's circuit.

B. Pose Alignment

We align subsequent poses using Iterative Closest Point (ICP) [2]. ICP is an iterative algorithm used to estimate the rigid-body transformation between two point clouds. Given two points clouds P and Q, in a single iteration, ICP computes, for each point in P a corresponding point in Q. Using these correspondences, the best rigid-body transformation is estimated, and then applied to P. This process is repeated, with correspondences being recomputed at every iteration.

Finding the best rigid-body transformation given correspondences is explicitly formulated as follows: suppose $\{(p_i, q_i) \subseteq \mathbb{R}^2 \times \mathbb{R}^2 : p_i \in P, q_i \in Q\}_{i=1}^n$ are the correspondences. Then point set registration finds the optimal rotation and translation to align the point clouds by solving

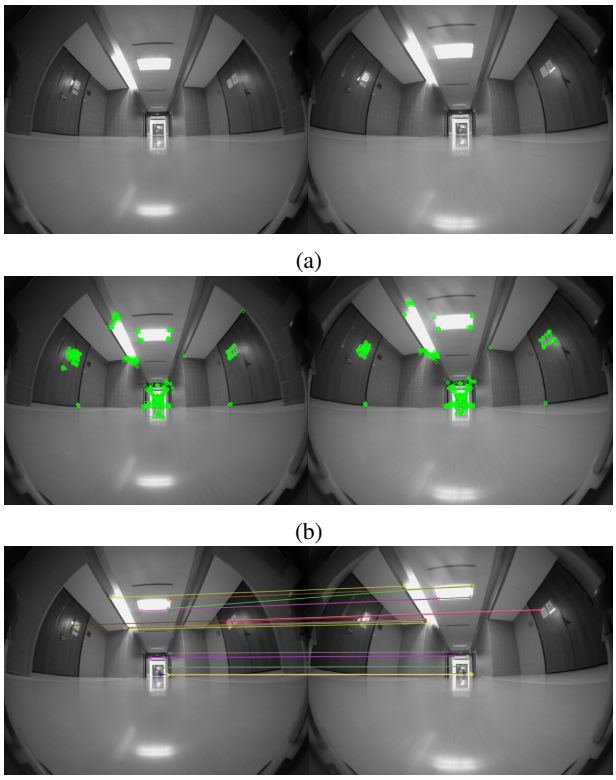
$$\underset{R \in \mathrm{SO}(d),\ t \in \mathbb{R}^d}{\operatorname{argmin}} \sum_{i=1}^{n} ||(Rp_i + t) - q_i||_2^2 \qquad (1)$$

This optimization problem has a closed form solution which can be efficiently computed [11].

ICP halts its iterations if the sum of the Euclidean error for each point is less than some small ε or if a maximum number of iterations is reached. For pose alignment, we use $\varepsilon = 0.05$, and cap ICP at 100 iterations.

C. Loop Closure Identification

Our process for loop closure identification uses images taken by the robot to identify candidate loop closures[2], and then determines the exact pose constraint using ICP. This method is based on the assumption that what the robot can see is unique to its location (at least to some extent). Fortunately, this turns out to be a reasonable assumption: even feature-poor environments like building hallways (which may seem rather indistinguishable to a person) will have small differences that can be recognized and leveraged with computer vision.



(a)

(b)

(c)

Fig. 4: A visualization of the loop closure detection process for a pair of images from the EECS building. The original images are shown in (a), and the detected ORB keypoints are shown in (b). The 20 best matches between the images are shown in (c). It is clear that the matched keypoints reliably represent the correspondences between the same objects across different images.

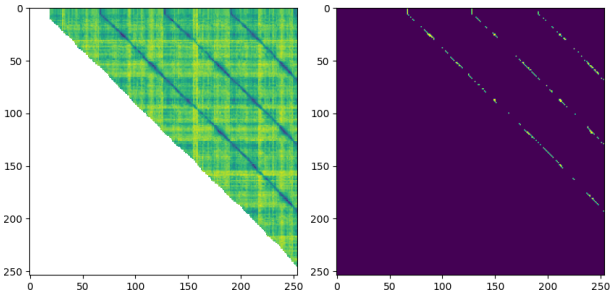
For each image, we extract ORB keypoints [4]. This is a technique for feature extraction: each keypoint is associated to a descriptor of the world at that keypoint. These features are rotation invariant, making them well-suited to matching between images taken from different angles. The ORB features were also robust to large amounts of lens distortion, and could be matched accurately between images, despite our usage of a fisheye lens. Although our system is reliant on the environment being largely unchanging, people walking in front of the robot and occluding portions of the image did not cause the keypoint matching to fail. An example of this is shown in Figure 6.

For each pair of images[3], we matched keypoints

[1] 1301 Beal Ave, Ann Arbor, MI 48109

[2] We also include a method for manually specifying candidate loop closures, but this was not used in any of our experiments.

[3] We do not consider pairs of images where the robot has traveled less than 5 meters between them, to avoid wasting time on loop closures that would be useless for pose graph optimization.



(a) (b)

Fig. 5: The image pairwise dissimilarity matrix, before (a) and after (b) thresholding, from the "EECS 3" run. The long, diagonal strips, where two series of images have been matched with each other in order, are caused by the robot traversing the same path at different times.

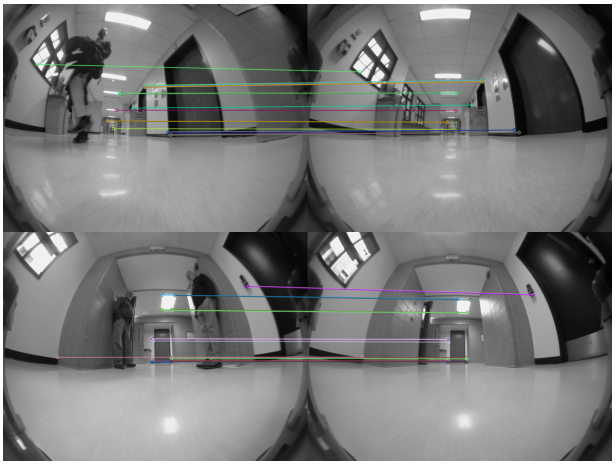


Fig. 6: Instances where people were visible to the robot during data collection. The keypoint matching technique is still able to identify a loop closure, even though the people were not present in both images.

between images using the FLANN algorithm [12] for approximate nearest neighbors. Each match has an associated distance value, describing how good the match is. We defined the image dissimilarity to be the sum of keypoint match distances for the k best keypoints, or infinity if fewer than k matches were found. (We used $k = 20$ in our experiments.) This process is visualized for a sample loop closure in Figure 4. Once the pairwise image dissimilarities have been found, we take the best match for each image, and then discard any matches with a dissimilarity above a certain threshold, which we set to be 2500. We include the dissimilarity matrix (before and after thresholding) in Figure 5.

After selecting a set of candidate loop closures, we use ICP to align the LIDAR scans associated to the images. If the ICP error is above a certain threshold (in our experiments, 30), we discard the loop closure. Otherwise, the transformation from ICP is added to the pose graph as a constraint.

D. Iterative Pose Graph Optimization

A pose graph is a directed graph $G = (V, E)$, where each vertex $v \in V$ identifies a robot pose and each edge $(v_i, v_j) \in E$ is the transformation $T_{i \to j} \in \mathrm{SE}(2)$ from pose i to pose j. Edges of the form (v_i, v_{i+1}) are obtained from the ICP scan matching in the pose alignment process, and other edges are obtained from the loop closure identification.

A pose graph with no identified loop closures is already fully-constrained, and the loop closures over-constrain the graph. By associating an error to each edge in the graph, depending on how well the transformation between the two poses matches the constraint in the edge, we can quantify the overall validity of the graph. Pose graph optimization seeks to find robot poses for every vertex, so as to minimize the total error. In our system, we use the fast iterative alignment approach of Olson et. al. [5]. Their approach uses stochastic gradient descent to iteratively improve the pose graph, and converges to the optimal solution even with very poor initial estimates.

After optimization, the positions are very accurate, but the orientations may be inaccurate. So we re-estimate the orientation of every pose in the pose graph: we take the tangent line to the trajectory at each pose, and treat that as the robot's orientation. This is a reasonable assumption, because the robot has a differential drive, although it can potentially cause problems if the robot turns in place.

IV. Results

In this section, we include results from three of our robot's trips around the EECS building. First, we include results from the "EECS 3" run, where the robot was driven in a square several times through the building, in Figure 7. Next, we include results from the "EECS 4" run, where the robot was driven in a figure-eight path, in Figure 8. Finally, we include results from the "EECS 6" run, where the robot was driven in a long, complex path throughout the entire building, in Figure 9. In all three cases, the initial pose graph is very noisy, but our system identifies a large number of loop closures, and utilizes them to successfully optimize the pose graph into an accurate and consistent map.

V. Discussion and Future Work

We have created a robotic mapping system, which is capable of producing accurate and consistent maps of large environments. Our system could produce accurate

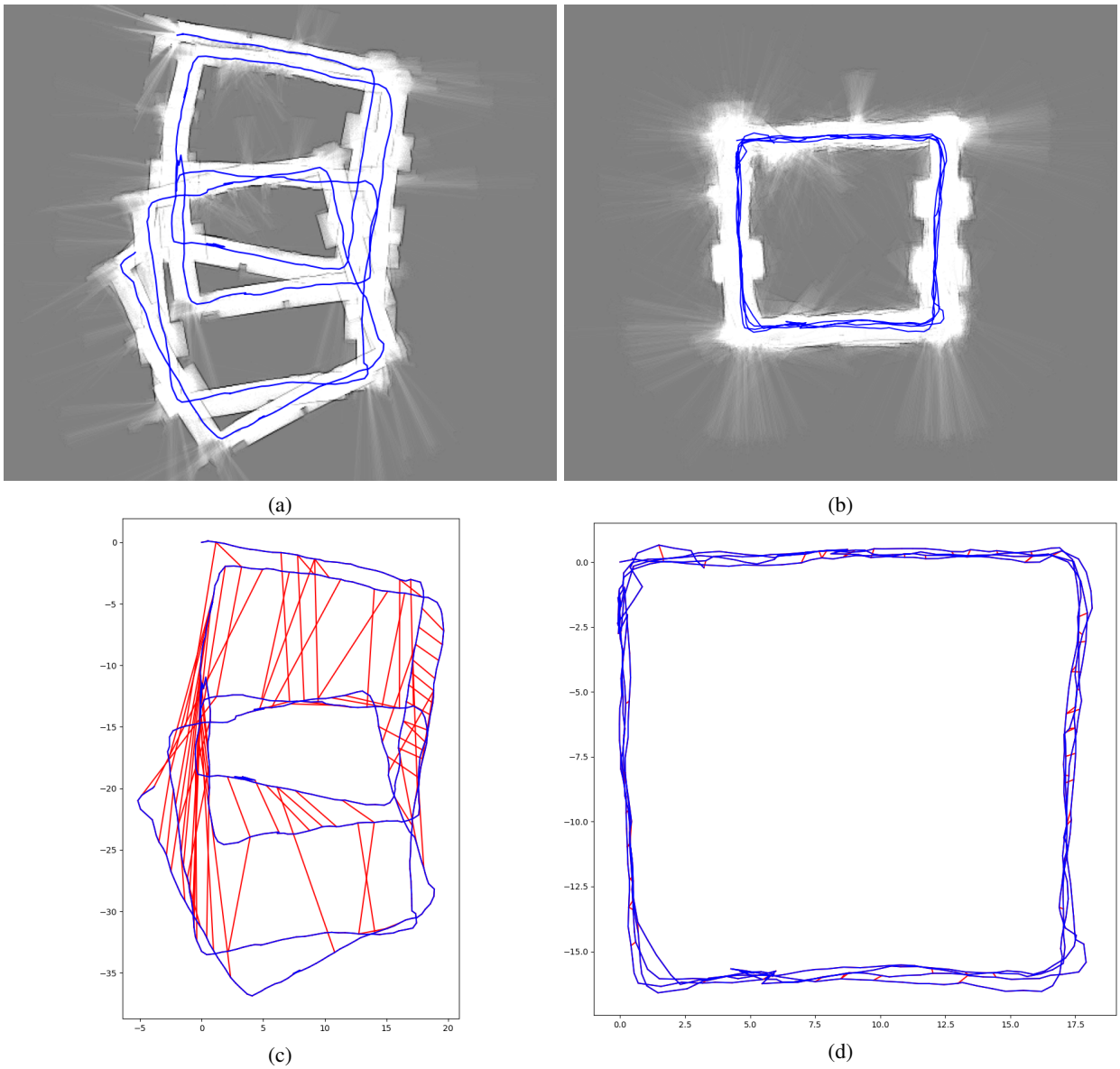


Fig. 7: For the "EECS 3" run, the map produced using ICP scan matching (a) and the final map after pose graph optimization (b). White areas are free space, black areas are obstacles, and grey areas are unknown. The robot path is annotated in blue. Also, the initial pose graph after detecting loop closures (c), and the optimized pose graph (d) that was used to produce the final map. Loop closure constraints are the red edges.

maps of the lab mazes using only the robot's odometry – an example is shown in Figure 10. The "go-kart" track proved to be a much more challenging environment to map, as the assumptions behind our loop closure system do not hold. Specifically, the robot can see over the short walls in the track, so the camera will be able to see the same features from different locations. Additionally, the walls of the track were almost completely featureless, causing ICP to struggle

There are a variety of ways that our system could be improved in the future. Our system does not fully model the uncertainty of the pose graph, instead using uniform weights for edges. By computing the covariance of the ICP pose alignment and loop closure identification, the pose graph optimization process could become more efficient. The system could also be made incremental, with optimization beginning as soon as the first loop closure is detected. This would hopefully prevent the pose graph from ever becoming too corrupted, leading to faster convergence.

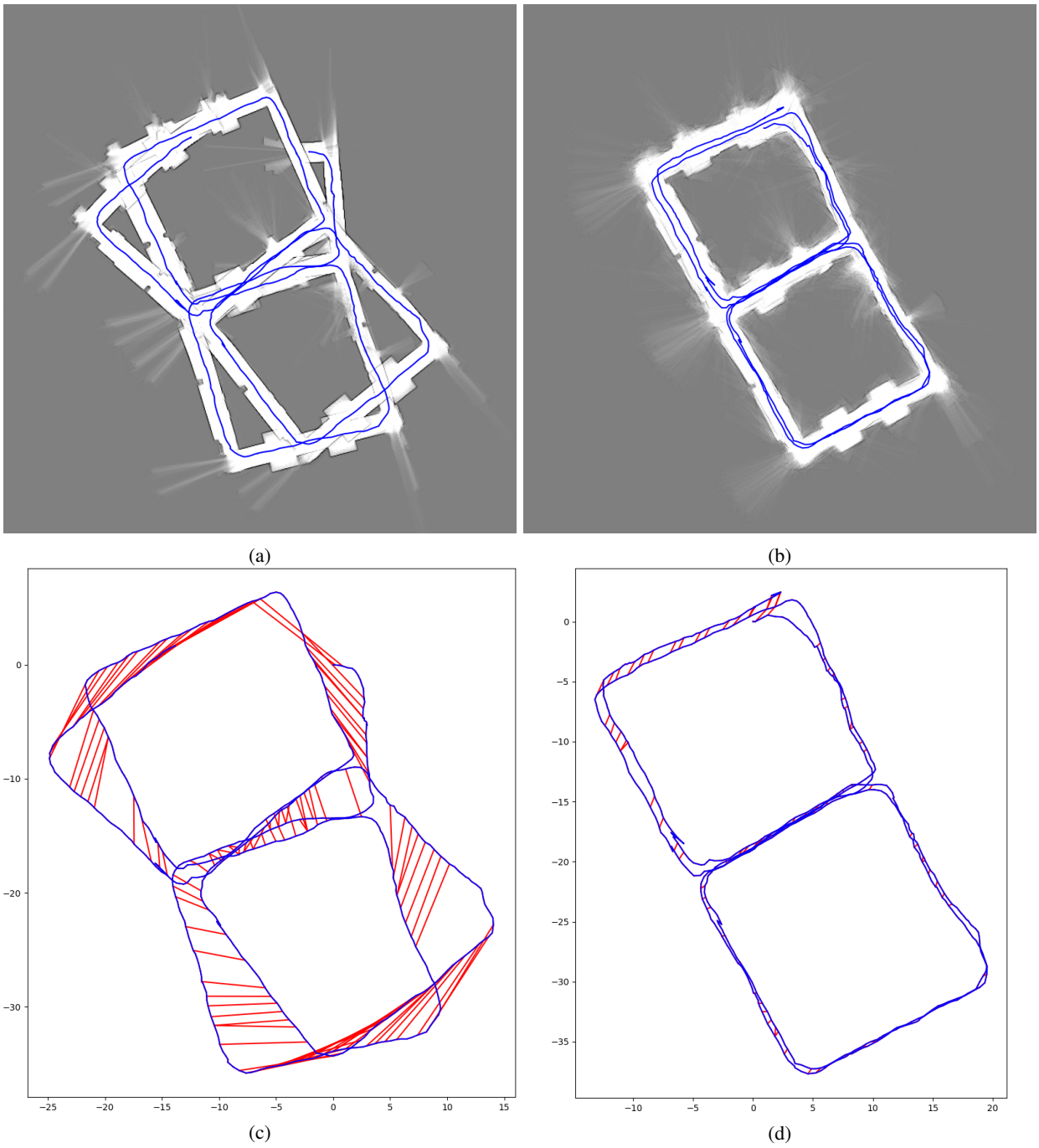


Fig. 8: For the “EECS 4” run, the map produced using ICP scan matching (a) and the final map after pose graph optimization (b). White areas are free space, black areas are obstacles, and grey areas are unknown. The robot path is annotated in blue. Also, the initial pose graph after detecting loop closures (c), and the optimized pose graph (d) that was used to produce the final map. Loop closure constraints are the red edges.

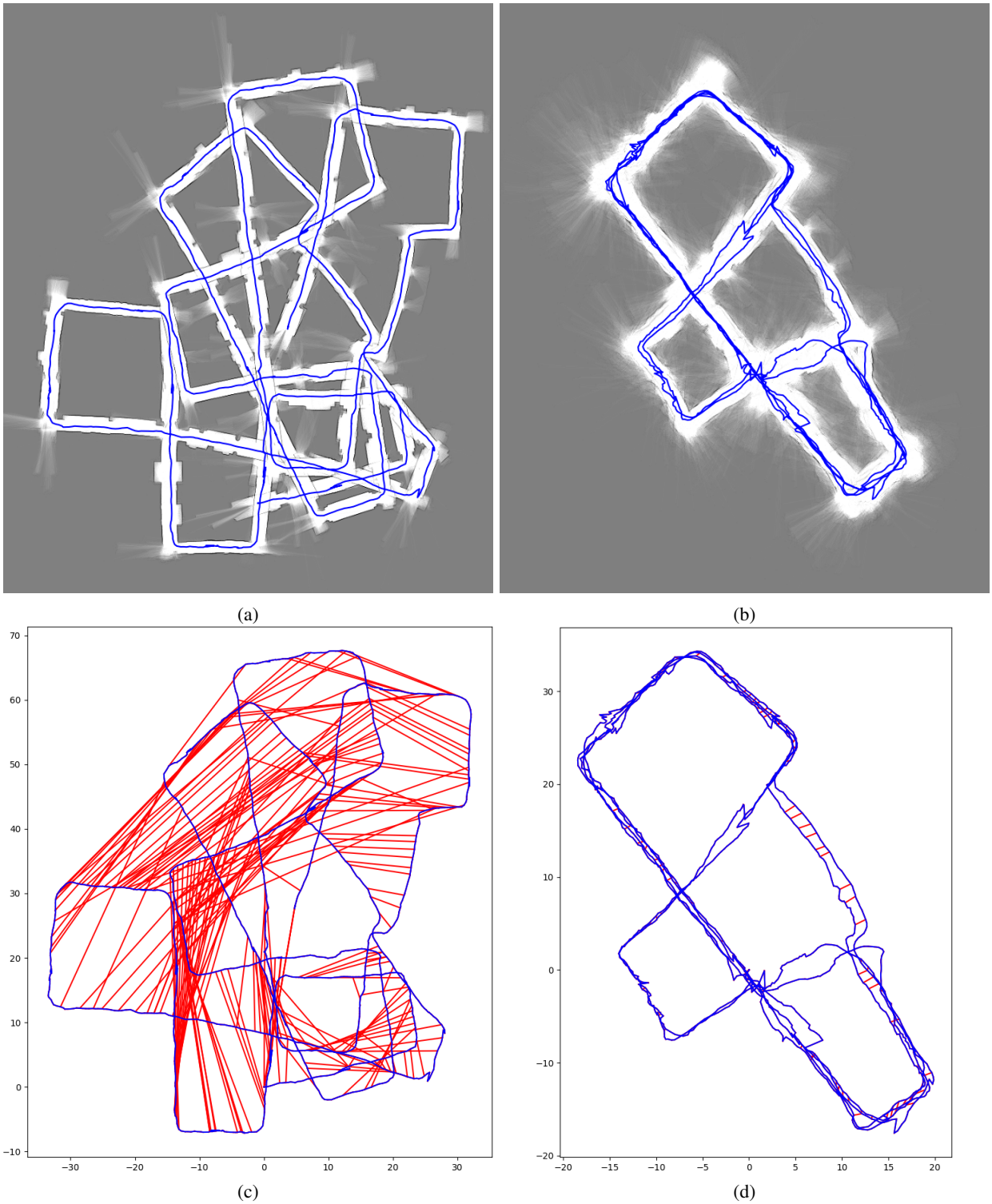


Fig. 9: For the “EECS 6” run, the map produced using ICP scan matching (a) and the final map after pose graph optimization (b). White areas are free space, black areas are obstacles, and grey areas are unknown. The robot path is annotated in blue. Also, the initial pose graph after detecting loop closures (c), and the optimized pose graph (d) that was used to produce the final map. Loop closure constraints are the red edges.

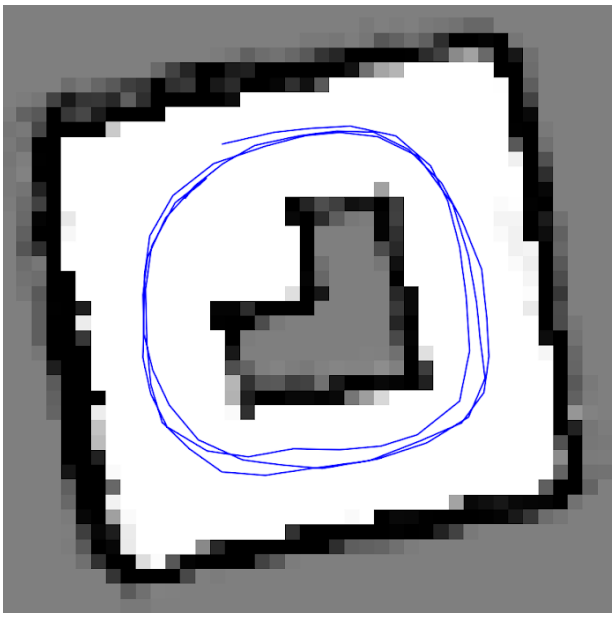


Fig. 10: A map of the lab maze, with robot pose estimated using just odometry.

Many directions for future work relate to the loop closure detection process. Because the robot's camera is forward-facing, it is very hard to detect loop closures if the robot is driving in opposite directions. This could potentially be improved by mounting the camera pointing straight up, or adding additional cameras. We could also estimate the motion of the camera using image feature matching, fully integrating the visual data into the mapping framework [13]. These keypoints could even be reconstructed in 3D via triangulation [14], allowing us to extend our SLAM system into 3D without any additional sensors. All of these features would likely require undistorting the camera images [15].

Finally, the brute-force image matching used to detect loop closures has quadratic time complexity in the number of images. To handle much larger amounts of data, representing image features as a "bag of visual words" can achieve similar rates of accuracy in a fraction of the computational runtime [16].

References

[1] K. L. Ho and P. Newman, "Loop closure detection in slam by combining visual and spatial appearance," *Robotics and Autonomous Systems*, vol. 54, no. 9, pp. 740–749, 2006.

[2] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor fusion IV: control paradigms and data structures*, vol. 1611. Spie, 1992, pp. 586–606.

[3] E. Mendes, P. Koch, and S. Lacroix, "Icp-based pose-graph slam," in *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2016, pp. 195–200.

[4] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International conference on computer vision*. Ieee, 2011, pp. 2564–2571.

[5] E. Olson, J. Leonard, and S. Teller, "Fast iterative alignment of pose graphs with poor initial estimates," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 2262–2269.

[6] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT Press, 2006. [Online]. Available: http://www.probabilistic-robotics.org/

[7] O. Ozisik and S. Yavuz, "An occupancy grid based slam method," in *2008 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications*. IEEE, 2008, pp. 117–119.

[8] S. Thrun and M. Montemerlo, "The graph slam algorithm with applications to large-scale mapping of urban structures," *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 403–429, 2006.

[9] J. Borenstein and L. Feng, "Gyrodometry: A new method for combining data from gyros and odometry in mobile robots," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 1. IEEE, 1996, pp. 423–428.

[10] A. S. Huang, E. Olson, and D. C. Moore, "Lcm: Lightweight communications and marshalling," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 4057–4062.

[11] O. Sorkine-Hornung and M. Rabinovich, "Least-squares rigid motion using svd," *Computing*, vol. 1, no. 1, pp. 1–5, 2017.

[12] D. A. Suju and H. Jose, "Flann: Fast approximate nearest neighbour search algorithm for elucidating human-wildlife conflicts in forest areas," in *2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN)*. IEEE, 2017, pp. 1–6.

[13] D. Nistér, "An efficient solution to the five-point relative pose problem," *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 6, pp. 756–770, 2004.

[14] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[15] D. Scaramuzza, A. Martinelli, and R. Siegwart, "A toolbox for easily calibrating omnidirectional cameras," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 5695–5701.

[16] A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer, "Fast and incremental method for loop-closure detection using bags of visual words," *IEEE transactions on robotics*, vol. 24, no. 5, pp. 1027–1037, 2008.

[17] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.