2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

OmniMapper: A Modular Multimodal Mapping Framework

Alexander J. B. Trevor[1], John G. Rogers III[2], Henrik I. Christensen[1]

***Abstract*— Simultaneous Localization and Mapping (SLAM) is not a problem with a one-size-fits-all solution. The literature includes a variety of SLAM approaches targeted at different environments, platforms, sensors, CPU budgets, and applications. We propose OmniMapper, a modular multimodal framework and toolbox for solving SLAM problems. The system can be used to generate pose graphs, do feature-based SLAM, and also includes tools for semantic mapping. Multiple measurement types from different sensors can be combined for multimodal mapping. It is open with standard interfaces to allow easy integration of new sensors and feature types. We present a detailed description of the mapping approach, as well as a software framework that implements this, and present detailed descriptions of its applications to several domains including mapping with a service robot in an indoor environment, large-scale mapping on a PackBot, and mapping with a handheld RGBD camera.**

I. Introduction

The literature includes a variety of Simultaneous Localization and Mapping (SLAM) approaches targeted at platforms with different sensors, environments, CPU budgets, and applications. It is clear from this diverse set of solutions that SLAM is not a problem with a one-size-fits-all solution. Different applications have varying requirements, and so require different solutions. Many of the existing solutions have limited applicability, and are not open. This implies that doing comparative experiments using different types of sensors, features or data association methods becomes a challenge. Concurrently utilizing data from multiple sensors, such as laser scanners and RGB-D sensors, is usually not supported.

Many factors can determine what type of mapping system is suitable for a given application. Environments vary in their appearance, structure, and texture; this can be seen in Figure 1. Due to this variety, a given type of feature may be better suited to some environments than others. Robotic platforms may also be equipped with different types of sensors and have varying computational capabilities, which impose requirements on what algorithms and feature types can be used. Different mapping applications also require differing information to be stored in a map, ranging from basic occupancy information to semantic maps that include place names and object information.

Most current approaches to SLAM support only one or two particular types of measurements. For example, the

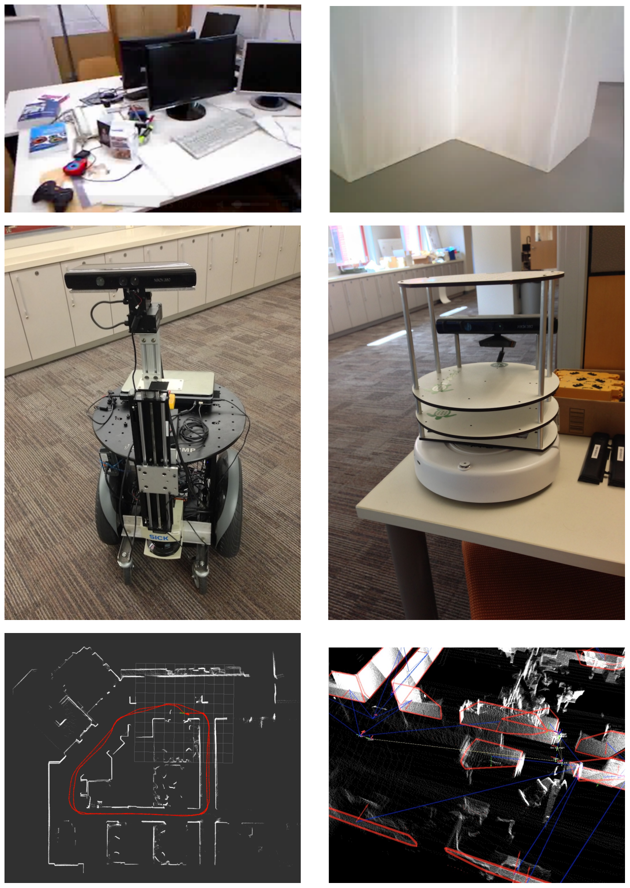

Fig. 1: Examples of environments with different properties (top), robot platforms with different sensors and computational capabilities (center), and different map representations (bottom). Top Left: an environment with a lot of visual texture. Top Right: an environment with little visual texture. Images are from the TUM RGB-D Dataset.

Iterative Closest Point (ICP) algorithm is popular for pose-graph solutions to the SLAM problem. Kinect Fusion [26] uses 3D Point-to-Plane ICP matching against a Truncated Signed Distance Function (TSDF) map representation. RGB-D SLAM [14] uses SURF features back-projected to a depth image from an RGB-D Camera. For some environments and applications, it can be advantageous to use multiple sensors concurrently. For example, may service robots are equipped with both laser scanners and RGB-D sensors, and these complement each other well for mapping purposes. Lasers typically have long range and wide field of view, while providing only 2D range measurements. RGB-D sensors

*This work was financially supported by the Boeing Corporation and the US Army Research Lab.

[1]Alexander Trevor and Henrik Christensen are affiliated with Robotics & Intelligent Machines, Georgia Institute of Technology, Atlanta, GA {atrevor, hic}@cc.gatech.edu

[2]John G. Rogers is affiliated with the United States Army Research Lab, Adelphi, MD john.g.rogers59.civ@mail.mil

have limited field of view and range, but provide rich sensor information. Combining these can lead to improved maps. In previous work, [35] we showed the benefits of using 2D line measurements from laser range finder concurrently with 3D planar landmarks extracted from an RGB-D sensor to take advantage of the different qualities of these two sensing modalities.

To better address the diverse range of SLAM problems and to enable easy multimodal mapping, we propose the OmniMapper, a modular multimodal mapping framework. Rather than providing a single mapping system with a fixed choice of features, we have developed a modular plugin-based software framework that can be easily configured to use different feature types, sensors, and produce different types of maps. Much as toolkits such as the Point Cloud Library (PCL) [30] and OpenCV [4] provide perceptual tools for addressing a wide range of 3D perception and computer vision problems, and "MoveIt!" [33] provides a framework for motion planning, we aim to provide tools for addressing the set of SLAM problems. By providing a *framework* rather than a library, we additionally impose some structure on what a solution should look like, while maintaining generality. The plugin architecture allows plugins to operate in one or more threads, so as to take advantage of modern multi-core CPUs. The key contribution is an approach to integrating data from multiple sensors and measurement types into a single factor graph, so all measurements can be considered jointly.

The remainder of this paper is structured as follows: Section II describes some related works. Section III describes the overall approach to the SLAM problem, and how multiple sensor types are used concurrently. Section IV provides details on the OmniMapper software framework, which implements this approach. Mapping results are provided in Section V for four different platforms, each with different sensors, operating in different environments and different configurations of the tools provided by our framework. We then conclude in Section VI.

II. Related Work

The Simultaneous Localization and Mapping (SLAM) problem was introduced by Smith and Cheeseman in [32], where an Extended Kalman Filter (EKF) on landmark positions and the robot position was used. Other formulations for the SLAM problem include particle filtering approaches such as FastSLAM [25], and graph based approaches such as Graphical SLAM [15], Graph SLAM [34], and Square Root SAM [11]. An introduction to SLAM techniques is provided in Durrant-Whyte's two-part SLAM article [13] [2].

Several libraries and toolkits for SLAM are currently available. The GTSAM library [12] provides tools for constructing factor graph based maps, and performing optimization and inference on these models, and is used in our work. Other recent graph optimization techniques include Toro [17], g^2o [21], and the Ceres Solver [1]. These tools are SLAM "back-ends", focusing on performing the optimization required for the SLAM problem, but providing limited or no support for feature extraction, data association, or interfacing with sensors.

Another related area of research is *feature-based* SLAM techniques, which use landmarks to solve the SLAM problem, in contrast to techniques which create pose-graphs. Some examples of frameworks for handling this type of technique include: the M-Space model [16] and the SP-Model [5].

Mapping approaches with multiple sensors and multiple feature types have also been proposed, some of which are highlighted here. Henry *et. al* [18] introduced RGB-D Mapping, which combined visual features extracted from the RGB camera with Iterative Closest Point (ICP) on the dense point cloud from the depth camera. Visual information and laser information have also been combined, as in the work of Newman *et. al* [27] where visual information was used for loop closure detection, and laser was used for building a geometric map. Visual measurements have also been combined with inertial measurement unit (IMU) data in previous work, such as the work of Leutenegger *et. al* [22]. These approaches were designed with specific platforms, sensors, or environments in mind. In contrast to these approaches, we provide a mapping framework in which different sensors and feature types can be changed or combined freely depending on the application.

The approach presented here builds on our previous work on SLAM and multisensory mapping. We demonstrated the usage of laser scanners, robot odometry, and visually detected door signs in [29]. Planar landmarks observed from both RGB-D cameras and 2D laser range finders were combined in [35], along with robot odometry. In this work, we build upon and extend these mapping approaches by making a more modular framework from mapping, allowing these approaches to be used interchangeably or combined with other techniques.

III. Mapping Approach

This section provides a brief overview of some of the key concepts used in our mapping system. The underlying factor graph representation and optimization tools are described, followed by our approach to building such factor graphs from a variety of different measurement sources concurrently.

A. Factor Graphs & Optimization

To optimize the robot trajectory and landmark positions, we employ a factor graph-based approach called Square Root Smoothing And Mapping ($\sqrt{SAM}$) [11]. In $\sqrt{SAM}$, the full robot trajectory is optimized, as opposed to a filtering approach in which only the most recent pose is present in the solution, and past poses are marginalized out. More specifically, we use the iSAM2 approach, which enables efficient incremental updates to the SLAM problem thereby enabling online operation. The details of iSAM2 are available in [19].

A key advantage of factor graph based SLAM is that a variety of different factor types can be used jointly in one optimization problem. Factor graphs are composed of variables, which for the SLAM problem represent entities in the world such as robot poses or landmarks, and factors, which provide probabilistic measurements that relate to one or more variables. Using this formulation enables us to jointly optimize robot trajectories and a variety of landmark types, and to use many different measurement types that relate these entities. Our framework provides a means to construct such factor graphs using a variety of measurement and sensor types.

B. Common Measurement Types

Many different types of measurements can be used for SLAM. We will highlight several of the important measurement types used by our system here.

1) Relative Pose Measurements: Relative Pose Measurements are measurements between two poses, and so are represented in the factor graph as binary factors that impose some constraint between these two poses. In our approach, there is a 1:1 correspondence between poses and timestamps, so this means each relative pose measurement has a corresponding time interval. A relative pose measurement consists of a spatial displacement $T \in SE(3)$, two time points that define the time interval, $time\_start \in \mathbb{R}$ and $time\_end \in \mathbb{R}$, and a noise model C that represents the uncertainty associated with this measurement.

Many sensors can produce this type of measurement. For example, scan matching approaches can provide relative pose measurements between sequential scans, or match to a scan from a previous pose to trigger a loop closure. Robot odometry and visual odometry also produce this type of measurement. A motion model such as a velocity motion model (or even a null motion model representing the expected absence of motion) can also be used to produce such measurements. Another example would be place-recognition measurements such as FABMap [8], which would add measurements indicating that two poses from different times correspond to the same place.

Our approach makes an additional distinction between types of relative pose measurements. Some measurement sources, such as robot odometry and inertial sensors, operate at a high rate, and are suitable for generating relative pose measurements for arbitrary time intervals by using interpolation if the exact requested timestamps are not available. We call this type of measurement source a *continuous pose measurement source*. On the other hand, measurement sources such as ICP or other scan matching approaches can only produce such measurements for certain time intervals where sensor measurements are available. As we will see in Section IV-D, we will treat these types of measurements sources differently, as continuous pose measurement sources will be used to link together all sequential poses.

2) Absolute Pose Measurements: Some sensors, such as GPS, external tracking systems such as Vicon systems, and celestial navigation approaches provide absolute position measurements in some fixed coordinate frame. If this external coordinate frame is used as the map frame, or a known transform between the two frames exists, such measurements can be used to directly measure the robot pose at a given timestamp. In contrast to relative pose measurements, these measurements are unary measurement factors, in that they only impose a constraint on a single pose in the graph.

3) Landmark Measurements: Another common type of measurement is a landmark measurement, in which the robot's sensors observe an external entity, usually called a feature or a landmark. The observed landmark observation then becomes part of the SLAM problem, and the landmark's pose is the optimized jointly with the trajectory. Multiple observations of the same landmark from different poses provide information about the robot pose, in addition to updated the landmark pose.

Several landmark types are supported including 2D and 3D points, 6DOF poses, and visual measurements from cameras. We have also introduced some new types, such as measurements of planes (e.g. walls), as described in detail in our previous work [35]. Such measurements typically occur in individual sensor frames, and as such will occur at a single point in time. These measurements are typically realized through binary factors between a pose and a landmark.

4) Virtual Measurements: When performing multimodal mapping, features detected by the robot can often be related to each other. For example, if we detect a wall using a laser scanner, and detect some visual features at a depth quite near to this wall, it is likely that these features are actually attached to the wall. Another example of a constraint between features is two walls that intersect at a corner and are perpendicular to each other. Introducing these types of constraints into our SLAM problem is a way of injecting knowledge about our environment into our map estimation, and can improve the mapping result. This type of relationship between landmarks can be thought of as a *virtual measurement* between the landmarks. In previous work [38], we introduced this type of measurement and described their use. The current OmniMapper framework does not include any implementations of virtual measurements, though support may be added as future work.

C. Trajectory Management

OmniMapper is designed to track the movement of a single entity such as a sensor or robot, as it moves through an environment. Note that this could be a robot with multiple sensors, but we choose one point on the robot to be the *base frame*. It is the trajectory of this *base frame* pose that will be tracked, and all measurements are transformed to this point.

While the mapper is operating, it adds poses to the trajectory periodically as it moves through space and time. A pose is added for each timestamp at which a measurement is provided, so there is a $1:1$ mapping between pose symbols and timestamps. We require that our trajectory is fully connected (multiple disjoint trajectories are not supported), so there must be at least one relative pose measurement between each sequential pose. Multiple measurements from different sources between sequential poses are supported – for example, we may have both odometric information and a relative pose measurement from ICP between the same pair of poses.

We make a distinction between continuous pose sources (high-rate measurements sources such as odometry, motion models, or inertial sensors) and other measurement types. It is generally not necessary or beneficial to add a factor between each odometric measurement, but instead to aggregate the odometric estimate into single factors for a longer time duration. Instead, the other measurement sources used in the system determine the timestamps at which poses should be created by requesting a pose to be created to correspond to a given timestamp. Given a sequence of timestamps (and therefore poses), these can be linked together by invoking all available continuous poses sources to create a relative pose measurement between these timestamps / poses. This ensures that our trajectory remains connected, and that the information from each continuous pose source is used for the entire trajectory.

Many sensors operate at different rates, and features may take different amounts of time to compute. This can lead to measurements arrive with out-of-order timestamps. This can be a challenge to avoid double-counting information from continuous pose measurement sources such as robot odometry. Two possible solutions to this issue include measurement queueing and pose splicing.

Measurement queueing involves delaying the generation of relative pose measurements and the addition of measurements to the map for some time window, to allow for lower rate sensors and features to be computed. After the time window has passed, the measurements are added to the factor graph, and relative pose measurements from continuous pose sources are generated. This approach is currently implemented in the OmniMapper framework.

An alternative approach would be to perform pose-splicing. In this approach, measurements are added immediately, and if an out-of-order measurement arrives in a time interval that has already been added to the factor graph, the previously generated factors from continuous pose sources are removed, and new ones are added to include the new timestamp, without double-counting information. This approach is not currently implemented in the OmniMapper, but may be added as future work.

IV. Software Architecture

The mapping approach described in Section III has been implemented in C++, and will be available as open source under the BSD license at ICRA 2014, from `http://www.omnimapper.org`. This section provides some details on this implementation.

Our framework integrates with and builds upon several other open-source software projects, including GTSAM [10], PCL [30], and ROS [28]. Detailed dependency information is available on our web site.

A. Optimization & SLAM Backend

SLAM systems are often regarded as being composed of two parts: a front-end and a back-end. The front-end provides feature extraction, interfaces to sensors, data association, and more, while the back-end performs the optimization to actually solve the SLAM problem. Our contribution is primarily on the front-end, though we go a bit further than traditional SLAM front-ends in that we also include support for addition of semantic information, as well as interactive semantic mapping applications as in [36]. We make use of the Georgia Tech Smoothing and Mapping Library (GTSAM) [10] as our SLAM back-end. While alternate SLAM back-ends such as g2o or Google Ceres could in theory be used in our framework, only a GTSAM factor-graph based back-end has been implemented as of this writing.

B. OmniMapperBase

OmniMapperBase is the core of the mapping system. This class is responsible for building the actual factor graph used to solve our Smoothing and Mapping (SAM) problem, and interacting with GTSAM and its iSAM implementation which performs the optimization. OmniMapperBase does not interact directly with any sensors; measurements are provided to OmniMapperBase via a measurement plugin architecture. The main functionality of this class is managing the trajectory and measurements as described in Section III-C.

Figure 2 provides a visual overview of how plugins interact with the mapper. Plugins process timestamped sensor data to generate measurements. These could be landmark measurements, relative pose measurements, or any of the types described in Section III-B. The plugins are responsible for generating measurement factors, which internally provide measurements that involve one or more symbols in the map. Given a timestamp, a plugin can request the pose symbol associated with that time using the $getSymbolAtTime$ function, which returns a $gtsam :: Symbol$. This symbol is then used to construct any factors that reference the pose at this time. Once the plugin has constructed a factor, it can add this to the SLAM problem by calling the $addFactor$ function, which will add it to the SLAM problem. Plugins can operate in parallel, executing in one or more threads each, allowing the system to take advantage of modern multi-core CPUs.

As described in Section III-C, the OmniMapperBase interacts with timestamped data. Different applications may require different methods of getting the current time. For example, when performing real-time SLAM, the system clock is generally the source of time, but processing logged data such as a ROS bag file may require using a different clock for playback at a different rate, or to enable pausing of the logged file. Both use-cases are supported in the current implementation.

C. Measurement Plugins

This section highlights a selection of measurement plugins currently available for use with our mapping system. Relative pose measurements are supported for building pose-graphs, and several landmark measurements are supported for feature-based SLAM. Both types can be used

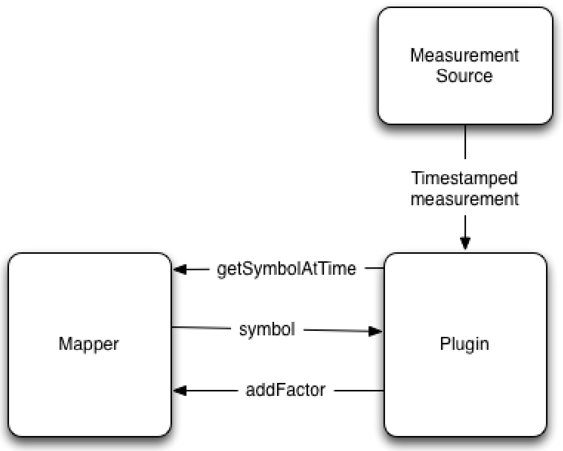

Fig. 2: An overview of the plugin-based architecture.

simultaneously. As described in Section III-C, the mapping system tracks the trajectory of a single entity, so we require all sensor measurements to be transformed to the *base frame*. Measurement plugins can be provided with a $getSensorToBaseFunctor$, which can return either a static or dynamic transform.

1) 3D Iterative Closest Point: Since its introduction in [3], the Iterative Closest Point algorithm and various other versions and adaptations have been popular for registration of point clouds. Our mapping framework includes a plugin for registration of point clouds using either the standard Iterative Closest Point algorithm [3], or the Generalized Iterative Closest Point (GICP) [31] algorithm. Both implementations used are from the Point Cloud Library (PCL) [30].

The plugin can be used to register full point clouds, uniformly downsampled point clouds, or other point sets. We have additionally used this plugin on 3D edge features, as demonstrated in our previous work [7] on edge-based registration and SLAM.

Loop closure with the 3D ICP plugin is handled by a thread which finds the closest place along the robots trajectory which was observed far enough in the past so as to favor large loops. If this closest place is sufficiently close to the robot's current location, then the ICP or GICP routine is used to find the transformation between the two keyframe clouds from these locations. If the ICP match is successful, then the resulting relative pose is added to the factor graph as an additional constraint. This additional constraint corrects any error which has accumulated since the robot visited this place before, thereby *closing the loop*.

2) Laser Scan Matching: Laser range finders such as the SICK LMS series and Hokuyo UTM-30LX are popular sensors for use on mobile robots. We developed a 2D scan matching plugin built upon Censi's Canonical Scan Matching approach [6], which can provide accurate pose registration between laser scans.

As with the 3D ICP plugin described above, in addition to being used for matching sequential scans, the scan matching plugin can also produce loop closures by attempting to match scans from nearby robot poses. If a match is successful, loop closure constraints are added.

3) Planar Surface Landmarks: Large planar surfaces can serve as excellent landmarks for indoor mapping systems. In our previous work [35], we described how such landmarks can be used in SLAM. To fully constrain platform motion, at least three planes in general position (not co-planar) must be matched between frames, which is not always possible. As such, these types of measurements are best used in multimodal mapping, in conjunction with other measurement types. The planar landmarks used in this work are extracted from RGB-D data using the organized connected component approach described in our previous work [37], but planes extracted from unorganized point clouds are also supported.

4) Object-Based Landmarks: Object-level mapping can also be useful both for SLAM and semantic mapping purposes. In previous work, we demonstrated the usage of recognized objects such as door-signs in SLAM [29]. The framework supports this type of measurement as an object plugin, which adds landmark measurements between a robot pose and either a 6D pose or a 3D point representing the object location, depending on whether or not orientation information is measured.

D. Pose Plugins

1) Robot Odometry: Many robotic platforms can report relative position changes using odometric information computed from wheel encoders. ROS [28] is often used on such mobile robots, and publish their position in an odometric coordinate frame using the TF library available with ROS. We have developed an odometry pose plugin for using odometric information published via TF. Such measurements are treated as a continuous pose source, so odometric information will be summarized in factors between each sequential pose / timestamp as described in Section III-C.

2) Visual Odometry: Visual odometry systems operate by generating relative pose measurements between sequential images. Many such systems operate at relatively high framerates (30 Hz), and are suitable for using in the same way as robot odometry. The odometry plugin described above can be employed with visual odometry systems that publish such results using the TF system in ROS. We have tested this using the TUM Dense Visual Odometry library by Kerl *et. al* [20].

3) Motion Models: In the absence of a continuous pose source, it may be helpful to use a motion-model as a pose plugin, to provide a prior on the type of motion that may occur. The toolkit provides a *null motion model*, which adds a weak prior indicating that no motion occurred between two timestamps. Such factors can serve to keep the trajectory connected in the absence of other measurements, for example if a plugin such as ICP did not converge, and was thus unable to provide a relative pose measurement for a given time interval.

E. Output Plugins

Output plugins are called after each map optimization, and are used to produce various types of outputs from the mapping system, including visualizations, and publishing or saving various kinds of map data.

1) Visualization Plugins: Visualizations are supported for ROS's RViz Visualization tool, as well as PCL based visualizations. As with the other plugins, these plugins are configurable, and can publish appropriate visualizations based on the content of the map and the user's requirements. Visualizations such as the robot / sensor trajectory, loop closure constraints, planar landmark locations and boundaries, and object landmark locations are supported.

2) Map Output Plugins: Many types of maps may be useful for different applications. One supported type of map output is an aggregate point cloud, in which many point clouds taken at different poses are aggregated in the map frame, using the optimized robot trajectory. Another type of useful map output is a mesh model of the environment. This is supported via a plugin that creates a Truncated Signed Distance Function (TSDF) volume [9] using a set of organized point clouds, and generates a mesh using the Marching Cubes algorithm [23]. Stephen Miller's CPU_TSDF library is used in this plugin [24].

F. Adding User Defined Plugins

The framework has been designed with extensibility in mind, so users can easily add new plugins of any of the types described above. To create a measurement plugin, all that is required is for a user to create a measurement of one of the types described in Section III-B and provides these to the mapper base as described in Section IV-B. Output plugins need only to install a callback to receive the resulting optimized factor graph.

V. Mapping Results

Our mapping system has been applied to a variety of different platforms, environments, and feature types. Several such applications are highlighted here, to demonstrate both the flexibility of the framework and its applicability to multimodal mapping.

A. 3D Semantic Mapping for Service Robots

Service robots can be equipped with a wide variety of sensors. We have employed our system on the Jeeves service robot, which is a Segway RMP base, equipped with a SICK LMS-291 Laser Range Finder and a Microsoft Kinect RGBD Camera mounted on a pan-tilt unit. Several different feature types have been used with this robot, including 2D scan matching, 3D ICP, and planar landmarks. The mapping system has additionally been used to support semantic mapping applications as in [36]. Figure 3 shows the platform used, as well as a system diagram and an example map.

B. Large Scale 3D Mapping

We have also used OmniMapper in larger scale operation on an iRobot PackBot platform in a number of test facilities. The experimental platform, system diagram, and example map output can be seen in Figure 4. In this configuration, the robot collects 3D point clouds from a Velodyne 32E laser scanner and uses the 3D Iterative Closest Point plugin described in Section IV-C.1 to build a map of the environment. In this example, the robot maps a large loop (0.3 km) around the corridors of an office building.

C. Mapping with a handheld RGB-D Camera

Creating maps using hand-held RGB-D Cameras has recently become popular, with approaches such as RGB-D Mapping [18], RGB-D SLAM [14], and Kinect Fusion [26]. Our framework has also been applied to this domain using several feature types. The example map shown here uses planar landmarks, edge ICP, downsampled cloud ICP, and dense visual odometry. The sensor, mapper configuration, and example map are shown in Figure 5.

D. Scan Matching for Mobile Robots in Industrial Environments

SLAM systems based on matching scans from laser range finders are very popular, and have been widely reported on in the literature. An example of such a system is the gmapping package distributed with the ROS navigation stack, and used on the PR2 robot. While several SLAM approaches can be applied to these types of measurements, such as Extended Kalman Filters and Particle Filters, graphical approaches can also be used.

We have applied the OmniMapper system to a holonomic industrial robot equipped with laser scanners and wheel odometry, shown in Figure 6. The system was used to map and localize the robot for navigational purposes in an industrial environment.

VI. Conclusions & Future Work

We introduced a multi-modal SLAM framework flexible enough to operate on a variety of different platforms and sensors, and in several different environments. The system was demonstrated on several real-world platforms and datasets. The software framework is available as open source at `http://www.omnimapper.org`.

In addition to enabling mapping for a variety of different platforms and environments, the OmniMapper framework is also suitable for comparative studies between feature types, or sets of feature types. As future work, we plan to examine the accuracy and efficiency of various feature types and mapping approaches.

References

[1] S. Agarwal, K. Mierle, and Others. Ceres solver. `https://code.google.com/p/ceres-solver/`.

[2] T. Bailey and H. Durrant-Whyte. Simultaneous localisation and mapping (SLAM): Part II state of the art. *Robotics and Automation Magazine*, September 2006.

[3] P. Besl and N. McKay. A method for registration of 3-D shapes. *IEEE Transactions on pattern analysis and machine intelligence*, pages 239–256, 1992.

[4] G. Bradski and A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library.* O'reilly, 2008.

[5] J. Castellanos, J. Montiel, J. Neira, and J. Tardós. The SPmap: A probabilistic framework for simultaneous localization and map building. *Robotics and Automation, IEEE Transactions on*, 15(5):948–952, 1999.

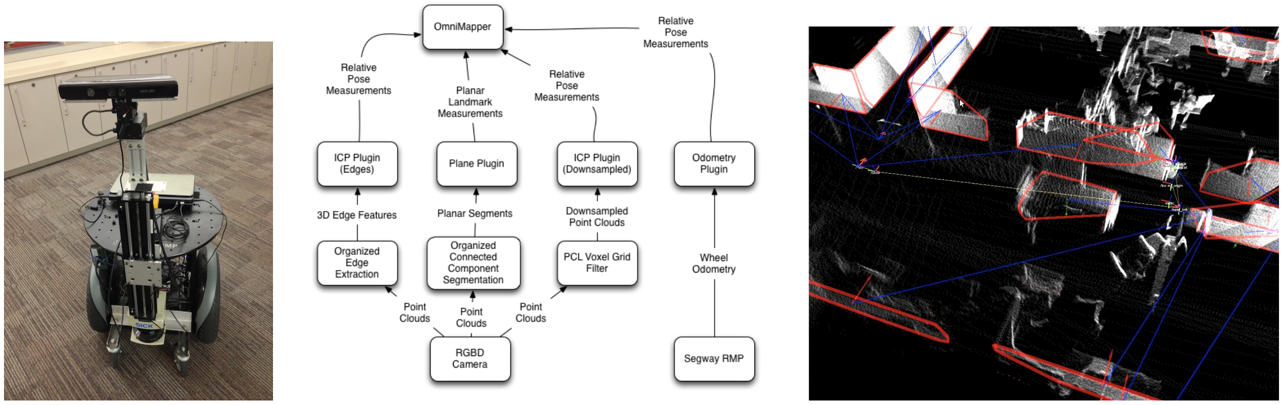

Fig. 3: Left: The Jeeves service robot platform. Center: A system diagram representing the configuration and plugins used. Right: An example map produced by our system with this platform and configuration.

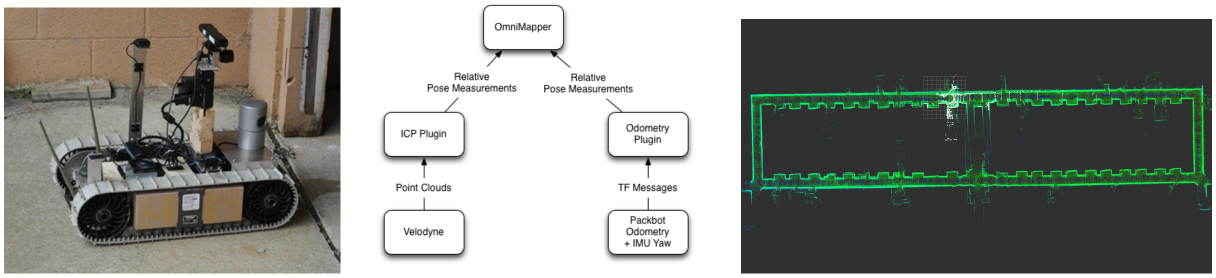
OmniMapper
Relative Pose Measurements
Relative Pose Measurements
ICP Plugin
Odometry Plugin
Point Clouds
TF Messages
Velodyne
Packbot Odometry + IMU Yaw

Fig. 4: Left: An iRobot PackBot equipped with a Velodyne 32E 3D laser scanner, a MicroStrain GX2 IMU, and an onboard computer. Center: A system diagram representing the configuration and plugins used. Right: An example map produced by our system with this platform and configuration.

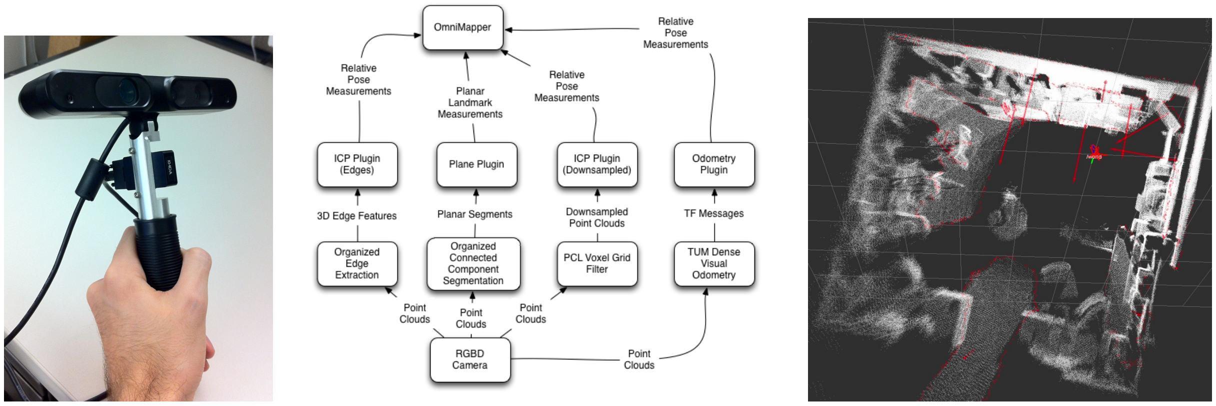

Fig. 5: Left: A handheld RGB-D camera. The pictured IMU is not used in this work. Center: A system diagram representing the configuration and plugins used. Right: An example map produced by our system with this platform and configuration.

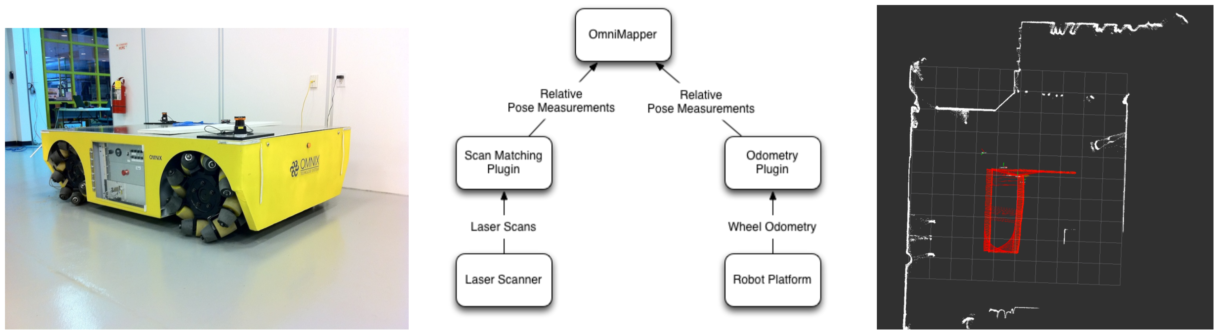

Fig. 6: Left: A holonomic mobile robot equipped with laser scanners and wheel odometry. Center: A system diagram representing the configuration and plugins used. Right: An example map produced by our system with this platform and configuration.

[6] A. Censi. An ICP variant using a point-to-line metric. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Pasadena, CA, May 2008.

[7] C. Choi, A. J. B. Trevor, and H. I. Christensen. RGB-D Edge Detection and Edge-Based Registration. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013.

[8] M. Cummins and P. Newman. Appearance-only slam at large scale with fab-map 2.0. *The International Journal of Robotics Research*, 30(9):1100–1123, 2011.

[9] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312. ACM, 1996.

[10] F. Dellaert. Factor Graphs and GTSAM: A Hands-on Introduction. 2012.

[11] F. Dellaert and M. Kaess. Square Root SAM: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25(12):1181–1203, 2006.

[12] F. Dellaert and M. Kaess. Square root SAM: Simultaneous localization and mapping via square root information smoothing. *International Journal of Robotics Research*, 25(12):1181–1204, 2006.

[13] H. Durrant-Whyte and T. Bailey. Simultaneous localisation and mapping (SLAM): Part I the essential algorithms. *Robotics and Automation Magazine*, June 2006.

[14] N. Engelhard, F. Endres, J. Hess, J. Sturm, and W. Burgard. Real-time 3d visual slam with a hand-held rgb-d camera. In *Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum, Vasteras, Sweden*, volume 2011, 2011.

[15] J. Folkesson and H. Christensen. Graphical slam-a self-correcting map. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 1, pages 383–390. IEEE, 2004.

[16] J. Folkesson, P. Jensfelt, and H. I. Christensen. The M-Space Feature Representation for SLAM. *IEEE Transactions on Robotics*, 23(5):1024–1035, 2007.

[17] G. Grisetti, C. Stachniss, and W. Burgard. Non-linear constraint network optimization for efficient map learning. In *IEEE Transactions on Intelligent Transportation Systems*, volume 10, pages 428–439, 2009.

[18] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments. In *the 12th International Symposium on Experimental Robotics (ISER)*, volume 20, pages 22–25, 2010.

[19] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *The International Journal of Robotics Research*, 31(2):216–235, 2012.

[20] C. Kerl, J. Sturm, and D. Cremers. Robust odometry estimation for rgb-d cameras. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, May 2013.

[21] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g^2o: A General Framework for Graph Optimization: A General Framework for Graph Optimization. In *International Conference on Robotics and Automation*, 2011.

[22] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart. Keyframe-based visual-inertial slam using nonlinear optimization. *Robotics, Science and Systems*, 2013.

[23] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM Siggraph Computer Graphics*, volume 21, pages 163–169. ACM, 1987.

[24] S. D. Miller. CPU TSDF Library. [Online] Available: `https://github.com/sdmiller/cpu_tsdf`.

[25] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *AAAI/IAAI*, pages 593–598, 2002.

[26] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011.

[27] P. Newman, D. Cole, and K. Ho. Outdoor slam using visual appearance and laser ranging. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1180–1187. IEEE, 2006.

[28] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, 2009.

[29] J. G. Rogers III, A. J. B. Trevor, C. Nieto-Granda, and H. Christensen. Simultaneous localization and mapping with learned object recognition and semantic data association. In *IEEE Conference on Intelligent Robots and Systems*, 2011.

[30] R. B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[31] A. Segal, D. Haehnel, and S. Thrun. Generalized-icp. In *Robotics: Science and Systems*, volume 2, page 4, 2009.

[32] R. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *International Journal of Robotics Research*, 5(4):56–68, Winter 1987.

[33] I. A. Sucan and S. Chitta. MoveIt! [Online] Available: `http://moveit.ros.org`.

[34] S. Thrun and M. Montemerlo. The graph slam algorithm with applications to large-scale mapping of urban structures. *The International Journal of Robotics Research*, 25(5-6):403–429, 2006.

[35] A. J. Trevor, J. Rogers, and H. I. Christensen. Planar surface slam with 3d and 2d sensors. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3041–3048. IEEE, 2012.

[36] A. J. Trevor, J. G. Rogers III, A. Cosgun, and H. I. Christensen. Interactive object modeling & labeling for service robots. In *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction*, pages 421–422. IEEE Press, 2013.

[37] A. J. B. Trevor, S. Gedikli, R. B. Rusu, and H. I. Christensen. Efficient organized point cloud segmentation with connected components. In *3rd Workshop on Semantic Perception, Mapping and Exploration (SPME)*, Karlsruhe, Germany, May 2013.

[38] A. J. B. Trevor, J. G. Rogers III, C. Nieto-Granda, and H. Christensen. Applying domain knowledge to SLAM using virtual measurements. *International Conference on Robotics and Automation*, 2010.