Mapping and Control with Telepresence Robots

Lingzhu Xiang
Department of Computer Science
Brown University
Providence, RI 02912
xlz@cs.brown.edu

Abstract

We create a mapping system and a control system on a telepresence robot to further facilitate indoor social navigation. The implementation is based on a commercial telepresence robot and its custom operating system. A mapping framework based on SLAM methods is employed to obtain real-time results. Navigation controls are implemented with inspection of the internal interfaces of the telepresence robot.

1 Introduction

Despite development of modern technologies of video conferencing, the presence of users is still limited to stationarily placed cameras and screens. Mobile telepresence robots liberate users from such constraints by enabling them to navigate around the environment and physically interact with their audiences. Recent products of telepresence robots include Beam by Suitable Technologies and Double by Double Robotics.

However, the same products also provide great opportunities for research in other areas. PR2 robots are the previous common platform used in robotics research communities. Due to the nature of PR2 being a high-end research robot equipped with an extensive set of expensive hardware, and its low volume of production, it encountered difficulties in accessibility, maintenance, and transportation. Meanwhile telepresence robots are usually designed as commercial products with focus on usability, connectivity, and battery life. With the help of economics of scale, they can be a great low-cost mobile platform with add-on functionalities for different specialized interests of research.

In this project, we investigate the possibilities of building navigation systems based on existing operating system of a commercial telepresence robot. We present the internal mechanism of such robot and methods of utilizing it, and also introduce frameworks to map the environment in real-time.

2 Related Work

Mapping with mobile robots is a well researched topic in the robotics literature as one part of the SLAM problem. Several frameworks have been developed for autonomous navigation of PR2 robots. GMapping [6] uses Rao-Blackwellized particle filters on a grid to learn the map by taking into account both the movement of the robot and the most recent observation. This method uses laser range data from depth sensors which are present on PR2 robots but unfortunately not present in most telepresence robots.

There also exists visual SLAM methods for mobile robots using only cameras or RGB-D cameras. RGBDSLAM [1] uses a Kinect RGB-D camera to capture visual features using SIFT or SURF descriptors and uses RANSAC to robustly model the 3D motion of the camera, i.e. the visual odometry, then optimizes the obtained camera pose graph with its SLAM back-end. This approach

was later enhanced [2] with a more efficient representation of the map using OctoMap [7] and GPU implementation of SIFT descriptors [13].

RTAB-Map [8] uses similar approaches in obtaining visual odometry with extensive choices of visual descriptors, and detect loop closures with a Bayesian bag-of-words detector to connect the current location to the previously visited location, and the minimizes the errors in the map with graph optimization methods. Due to the modular design of the RTAB-Map implementation, we base our system on RTAB-Map to realize real-time mapping on the telepresence robot.

3 Platforms

In this project, Beam, a telepresence robot is used as the robot platform, and Robot Operating System (ROS) [9] is used as the software frameworks connecting Beam to our software components. We also attach a Asus Xtion Pro Live RGB-D camera and a laptop onto Beam for additional computation. Microsoft Kinect is possible but it requires external power which is not feasible as an attachment to the telepresence robot, while the Asus camera can be powered with USB only.

3.1 Beam

Beam is a telepresence robot by Suitable Technologies. It was originally developed by Willow Garage as Texai Remote Presence System, and spun out with most of Willow Garage team to build it as a product. Beam's specification claims 8 hours of active use and 24 hours for standby which is much longer than the battery life of PR2 of 2 hours, and much more user-friendly for research and social purposes.

From our inspection, Beam is equipped with two Logitech C910 cameras augmented with wide-angle lenses, an LCD screen, a microphone array, a built-in speaker. It has three casters for movement and two Wi-Fi radios for connectivity. Its cameras are one front-facing and one downward-facing, which output up to 1920x1080 JPEG for 30 frames per second. And this is able to be transmitted over its Gigabit Ethernet connection. It includes a dual core Intel Core i3 processor and 4GB RAM which are far from being as powerful as the ones of PR2, but much better than those ARM processors on most embedded mobile robot platforms.

Beam runs on a Linux distribution modified from Ubuntu with no source code provided for the modifications. However, the Linux environment is still helpful for deploying our own code and inspecting the internal mechanism of Beam.

Figure 1: Beam telepresence robot, the camera and the wheels can be seen.

3.2 ROS

Robot Operating System (ROS) is a middleware platform for robot software development. It provides standardized message-passing interfaces between modularized components which make integration straightforward. Its ecosystem of hardware abstraction, device drivers, commonly used libraries, and package management reduces duplicated effort in implementation.

A running ROS instance consists of a master node which negotiates with other nodes to connect them together, and server nodes and client nodes requesting and replying messages according to the ROS remote procedure call protocol. This standard protocol provides a way to integrate heterogeneous software from different developers and different communities, which is the reason for the booming ROS ecosystem.

In this project, we utilize ROS to provide a bridge between Beam and our laptop to relay the camera output from Beam and exposing Beam's motor controls. Thanks to the modularized nature of ROS, we are able to accomplish the tasks without reimplementating everything.

4 Navigation Systems

Following the goals illustrated in the introduction, we investigate the potentials of building systems for navigation based on the telepresence robot. Here we focus on the mapping system and the control system.

4.1 Mapping

We build the mapping system using RTAB-Map [8] which provides modular components based on ROS so that some of them can be easily replaced. As illustrated in figure 2, the mapping system is one part of the classic design of SLAM system. However, here we do not focus on localization and its integration with planning and control considering the scope of this project.

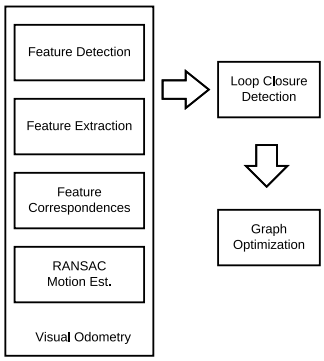

Figure 2: SLAM pipeline

We utilize the RGB-D camera with RTAB-Map. It is attached to the top of the telepresence robot and connected to the laptop which is placed on the bottom of the telepresence robot. The mapping system collects point cloud data from the RGB-D camera along estimation of visual odometry, then detects potentials loop closures by appearance based bag-of-words detectors using visual descriptors, and then optimizes the map with a graph optimization framework.

Given visual feature descriptors with depth data, visual odometry is more straightforward than with monocular cameras where the depth of particular keypoints still need to be estimated. The odometry node will detect features and extract them from the RGB frames, and then associate them with the depth data from the depth frames. Then between frames, feature correspondences are established by similarity metrics. After that, a RANSAC technique computes the relative motion between the frames which is the odometry. By integrating relative motion, a trajectory can be estimated, albeit almost always distorted over time. But the globally inaccurate trajectory still provides strong local consistency for map optimization given loop closures.

4.2 Control

We expose the controls by inspecting the internal mechanism of the telepresence robot and apply our modifications. The telepresence robot includes three wheels, of which the rear two are powered by electric motors and the front one controls the driving direction. The interfaces of controlling the motors and direction are not public, therefore they require inspection of Beam's internal mechanism.

Inspection of Beam's internal hardware confirms that Beam carries a separate control board at its bottom for controlling its motors. And the main computer communicate with the motor board through a USB serial port. Further inspection of the serial wire transmissions within its Linux operating system shows that the control program on the main computer sends drive command packets to and receive driving state packets from the motor board with fixed frequencies. When the telepresence robot is in active operation, the frequency is 100Hz; when it is idle, the frequency is 10Hz. The drive command packets have a fixed format which is determined by extensive testing. The driving state packets also have a fixed format but the format is much more complex therefore it is part of the future work.

Offset	0-3	4-7	8-11	12-15	16-19	20-23
Field	Header	Mode	Linear vel.	Angular vel.	Type	Tag
Offset	24-27	28-31	32-35	36-39	40-43	44-47
Field	Tag	Linear limit	Angular limit	Linear acc.	Angular acc.	CRC

Table 1: Drive Command Packet Format

As shown in table 1, the drive command packet format has fixed length of 48 bytes with multiple fields specifying the parameters of the drive command. All numeric linear and angular parameters are encoded using 32-bit fixed point number with scaling factor of 1/65536. The header is a constant

0x5555aaaa for drive command packet. The linear and angular limit parameters, and linear and angular acceleration parameters remain unchanged for normal operation with the defaults being 1.5 for the limits, and 2.0 and 0.5 for the acceleration parameters. The "tag" field is a timestamp tag that specify the expiry time for the particular drive command. After initial time synchronization between the main computer and the motor board, any future drive command with older than current timestamp would be ignored by the motor board.

With the rest of the parameters fixed, the effective parameters are linear velocity v_l and angular velocity v_r. A simplified kinematics model of Ackermann steered vehicle provides simple geometric relationship which can be used to determine the path of the vehicle given these parameters. [11] The geometric relationship can be written as:

$$\tan(\delta) = \frac{L}{R}, \tag{1}$$

where δ is the steering angle of the front wheel, L is the length of the wheelbase, and R is the radius of the circle at the current position of the trajectory. The radius R also satisfies $v_l = Rv_r$. Given a physically fixed constant L, and the linear velocity v_l, the steering angle δ can be determined by $\delta = \arctan(\frac{Lv_r}{v_l})$. However, we do not have to specify the steering angle of the front wheel as the parameter is derived within the motor board by given only the linear velocity and angular velocity parameters.

5 Implementation

The Linux distribution running on Beam is a modified version of Ubuntu. It follows typical design of embedded system by bootstrapping with a read-only SquashFS image, and then creating the root filesystem by extracting files from several tar archives and then performing chroot operation to change to the new root directory and start applications. By modifying the static images and archives we can obtain user credentials and be able to log in to remote shell running on the telepresence robot and execute our commands. By leveraging similar techniques of chroot, we can deploy newer version of Ubuntu filesystem and updated applications with full privileges.

Several sensors for mapping purposes require calibration to estimate their intrinsic parameters. The RGB-D camera is provided with pre-calibrated parameters. The cameras on the telepresence robot are augmented with wide-angle lenses therefore careful calibration is required. There are several calibration tools. ROS provides a package called `camera_calibration` which estimates a pinhole model. However this package was very difficult to set up. A modified version of PTAM [12] also provides a calibration tool which is also unmaintained and outdated for the latest ROS distribution Hydro. AprilCal [10] also provides a calibration tool with state-of-the-art accuracy which also focuses on user-friendly interface by provided an interactive process of suggesting optimal calibration poses. By using this tool the calibration process was able to obtain the mean reprojection error of 0.307 pixel with 20 optimal frames. AprilCal still faced several challenges which were revealed during the actual calibration process. It does not provide an interface to connect with ROS making it difficult to operate with image frames transmitted over network. The only solution was to run it on the telepresence robot's main computer which is relatively slow. The result was that each suggestion for calibration poses would take several seconds to more than ten seconds to appear, defeating the goal of usability.

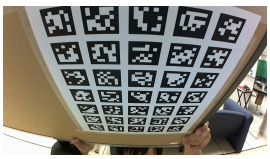

Figure 3: An optimal calibration pose suggested by AprilCal showing the tangential distortion.

The telepresence robot includes two Logitech C910 cameras which are well supported by UVC video drivers and Video4Linux video capture framework in Linux kernel. The `uvc_camera` package from ROS provides interfaces to export image frames via Video4Linux framework. We contributed a patch to the project of the package for enabling it to extract image frames in JPEG encoding. Before that the only available format is uncompressed which makes high resolution frames exceed the bandwidth capacity. After enabling its JPEG format, the framework is able to capture and transmit JPEG encoded 1920x1080 images 30 frames per second over Gigabit Ethernet to the laptop for further computation.

We expose the drive command interface with instrumentation of the official driving program. Due to the commercial nature of the telepresence robot, its software is being constantly updated and maintained. To avoid unnecessary breakage over time, we want to minimize the code footprint of our modifications. Therefore we utilize the LD_PRELOAD and ptrace mechanism of Linux to intercept a minimal set of system calls with which the official driving program sends drive command to the motor board, and replace the parameters of the system calls which are discussed previously as the drive command format with our desired values. This process is essentially to tap the drive command communication between the main computer and the motor board, and intercept those which we want to modify to realize the effective control. There are differences for LD_PRELOAD based approach and ptrace based approach. The code loaded via LD_PRELOAD is run as part of the drive command program therefore in order to send our desired parameters it needs to utilize inter-process communication mechanism. Here we implement the IPC with POSIX semaphores and shared memory objects. The approach based on ptrace is more flexible and works with static binaries which is not the case for LD_PRELOAD approach. The drawbacks of ptrace based approach is that it introduces overhead and its failure can cause interruption in communication to the motor board which can be dangerous for a moving mobile robot.

The telepresence robot does not include a low-level emergency stop mechanism. We employ a dead man's switch method by implementing a watchdog timer within the drive command interface. A remote operator should set up a ping program to send ICMP ping packet containing a particular string in very short intervals. The telepresence robot will then monitor all ICMP ping packets with the Netfilter interface and set up a watchdog timer to count down from the last time when it receives a ping packet containing the particular string. When it does not receive such packets for a certain time, for example 500 millisecond, the timer expires and the drive command parameters are reset to prevent any further movement. For the operator to actively stop the robot, they can just terminate the ping program and the robot will stop moving within the given deadline.

6 Experiments

We performed mapping experiments using previously discussed navigation systems on two different places. The RGB-D camera was attached to the top of the telepresence robot to collect point cloud data. Two datasets were obtained. The first dataset was obtained in a lab on the first floor by driving the telepresence robot within a small circular area shown as a black circle in the middle of figure 6. The second dataset was obtained in a lounge on the fourth floor by driving the telepresence robot through chairs. In both datasets the robot has come back to the starting point to verify the effectiveness of loop closure. The first dataset includes 339 frames and the second dataset includes 466 frames.

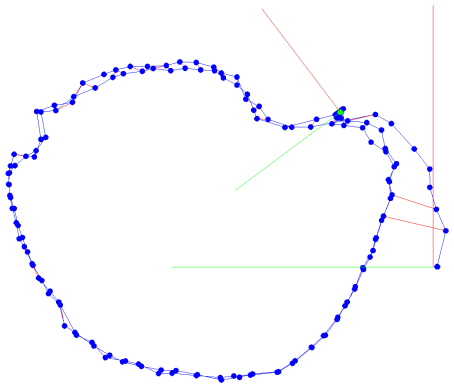

Figure 4: Trajectory estimated for the first floor dataset

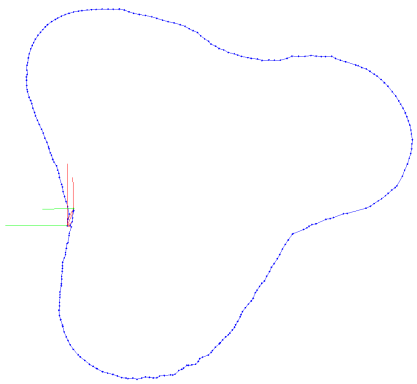

Figure 5: Trajectory estimated for the fourth floor dataset

We utilize SIFT features with depth information to estimate visual odometry with RANSAC technique. As shown in figure 4 and figure 5, the blue points are robot poses, the blue edges are motion between poses, and the red edges are loop closure constraints recognized by the appearance based detector between poses. The red-green axes illustrate the starting pose and the ending pose. In both

datasets, it is able to obtain visual odometry of significant accuracy and correctly establish loop closures.

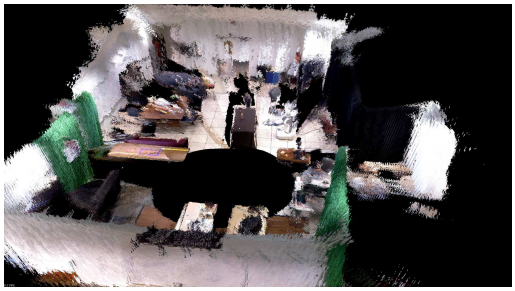

Figure 6: 3D map estimated for the first floor dataset

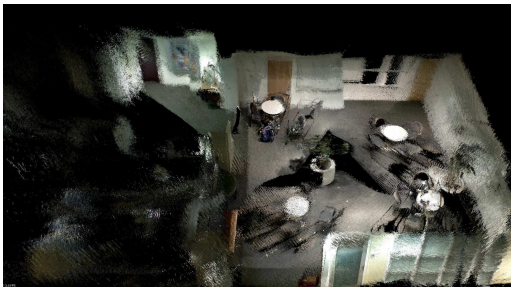

Figure 7: 3D map estimated for the fourth floor dataset

The 3D point cloud map created using the estimated poses are shown in figure 6 and 7. For the dataset on the first floor, there is a black circular area in the middle because the RGB-D camera on the top of the telepresence robot did not observe the area underneath it during the whole collection process due to the circular trajectory. The resulting map for the first floor is fairly accurate with black grid on the floor correctly connected together, and the colored lines on the table correctly connected together. The map is less accurate for areas further from the camera, because of the limit of accuracy of the depth sensor over a large distance. The second dataset on the fourth floor shows similar accuracy. Near the end of data collection there was a person in blue moving across, which can be seen in the middle of the map. The SLAM framework assumes static environment therefore some distortion in this map can be attributed to the moving person.

Overall the mapping system is able to obtain accurate results using the RGB-D camera in real-time. Moving objects during mapping process can still result in geometric distortion. Note that results obtained by driving through the interface provided by the motor board are much more accurate than those by manually pushing the telepresence robot. This is probably because manual operation results in non-uniform motion which causes inaccuracy in visual odometry. Drive command set a constant velocity for regularized motion help to improve the accuracy of visual odometry.

7 Future Work

We have demonstrated the efficacy of our systems. Several challenges still remain to be solved. The primary challenge for the RGB-D SLAM approach is that the RGB-D camera has limited depth range and it relies on visual features. If the scene is featureless, for example, white walls and uniform texture, there will not be enough features detected for estimation of visual odometry. The other situation when this can happen is in a large environment where only a small area near the robot has depth readings and the other area is too deep, for example, corridors and hallways.

SLAM based on direct image alignment instead of feature correspondences may be useful for solving the problem with difficult scenes. However given the limited computation resources of a mobile telepresence robot, full dense approaches might exceed its computation capacity. Semi-dense monocular visual SLAM approaches including SVO [5] and LSD-SLAM [3] [4] is applicable to the problem.

The calibration process with AprilCal is slow running on the telepresence robot. To speed up the process, it is possible to modify the AprilCal program to receive remote image frames via ROS interfaces and conduct calibration.

The current implementation of mapping relies on visual odometry which can be inaccurate under adverse circumstances. The official driving program on the telepresence robot receives certain kind of driving state update packets which is worthy for study for the purpose of extracting wheel odometry. Wheel odometry is helpful for improving the results of SLAM framework.

8 Conclusion

This report shows methods to build mapping framework and control framework for a telepresence robot as a platform for broader research topics. The mapping framework and the control framework are implemented and deployed to the telepresence robot for operation. The feasibility of the mapping framework is shown with results from experiments on collected real world datasets. The implementation is discussed in detail to provide internal details and guidance on different ways of utilizing commercially available robots for research purposes.

References

[1] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An evaluation of the rgb-d slam system. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012.

[2] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard. 3d mapping with an rgb-d camera. *IEEE Transactions on Robotics*, 2014.

[3] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision (ECCV)*, September 2014.

[4] J. Engel, J. Sturm, and D. Cremers. Semi-dense visual odometry for a monocular camera. In *IEEE International Conference on Computer Vision (ICCV)*, Sydney, Australia, December 2013.

[5] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[6] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23:34–46, 2007.

[7] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013.

[8] M. Labb and F. Michaud. Online global loop closure detection for large-scale multi-session graph-based slam. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014.

[9] M. Quigley, K. Conley, B. P. 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*, 2009.

[10] Andrew Richardson, Johannes Strom, and Edwin Olson. AprilCal: Assisted and repeatable camera calibration. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, November 2013.

[11] Jarrod M. Snider. Automatic steering methods for autonomous automobile path tracking. Technical report, Robotics Institute, Carnegie Mellon University, 2009.

[12] Stephan Weiss, Markus W. Achtelik, Simon Lynen, Margarita Chli, and Roland Siegwart. Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[13] Changchang Wu. SiftGPU: A GPU implementation of scale invariant feature transform (SIFT). `http://cs.unc.edu/~ccwu/siftgpu`, 2007.