Robust and Efficient Calibration of Mobile Manipulators

Michael Ferguson
Fetch Robotics Inc.
San Jose, CA 95131

Niharika Arora
Georgia Institute of Technology
Atlanta, GA 30332

***Abstract*— Calibration is an essential prerequisite to many mobile manipulation tasks. Mobile manipulators typically have multiple sensors and at least one actuated kinematic chain which must be well calibrated. Such calibrations must also be performed in a robust manner and, ideally, quickly. This paper presents a calibration system developed for the Fetch mobile manipulator, which is highly robust and can calibrate the entire robot in less than two and a half minutes without any operator interaction. This is significantly faster than existing full-system calibration systems for mobile manipulators. Having such a calibration system has also allowed several cost reductions and simplifications in the robot design which are also discussed.**

I. INTRODUCTION

For many mobile manipulation tasks to be successfully performed the sensors and kinematic chains of a robot must be well calibrated. Only with proper calibration can sensor data be effortlessly transformed between the coordinate frames of the manipulator and the coordinate frames of the sensor. For robots with multiple sensors, each sensor must be well calibrated with respect to the other sensors if attempts at sensor fusion are to be made.

A number of systems have been proposed for the calibration of mobile manipulators, several with available open-source implementations. However, these systems tend to be crippled by one or more of the following issues:

- An inability to generalize across sensors and platforms. Many calibration systems are carefully designed or heavily tuned around a particular set of sensors and are unable to generalize beyond those sensors.
- Slow calibration times. The calibration system built for the PR2 robot [1] often takes as much as 45 minutes to calibrate the robot, with half of that time spent collecting calibration data, and half spent doing the actual optimization.
- Lack of robustness. Frequently, these calibration systems have issues which cause them to fail to get a "good" calibration on each attempt, which can be particularly detrimental when the calibration takes a significant amount of time to redo.

Our system of calibration consists of two major phases: capturing data about calibration targets which indirectly gives us information about the kinematics and sensors, and then optimizing the parameters of the system so as to minimize the errors between sensor and kinematic alignment.

During the capture phase, the robot moves to each of several dozen pre-defined poses. Each pose puts the robot joints into a particular configuration and makes sure the sensors are pointing at the desired calibration target. The position of the robot joints are recorded along with the detected pose(s) of the calibration target.

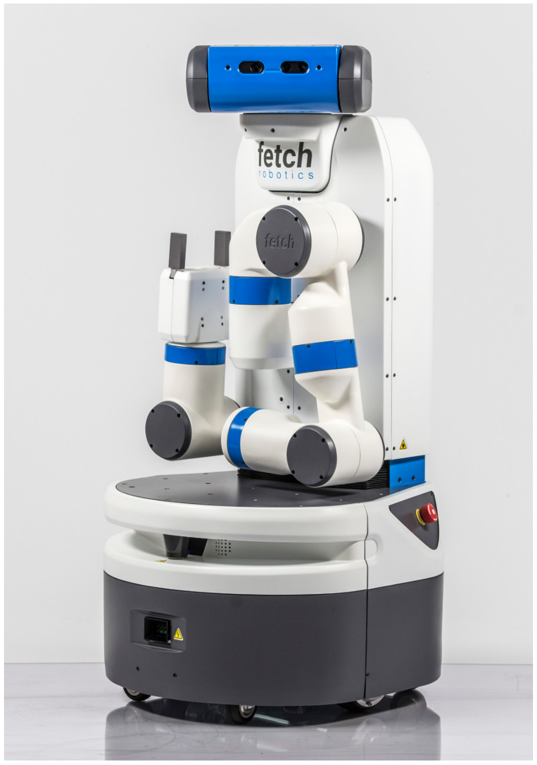

Fig. 1. The Fetch mobile manipulator for which this calibration system was initially designed. The robot comprises a 7-degree of freedom arm, pan and tilt head, torso lift joint, and RGB-D depth camera which must all be calibrated.

During the optimization phase, we re-project the expected position of the calibration target through each of the kinematic chains and sensors. The difference between these measurements form a residual error over which optimization can be performed. Our system is capable of handling multiple types of parameters, including the offsets in the joint angles, offsets in the location of kinematic frames, camera intrinsics and extrinsics. In addition to these standard parameters, we also optimize some of the parameters of various drivers, such as the Z-offset and Z-scaling parameters of the Primesense camera drivers.

For the Fetch robot, we use a set of four LEDs in the gripper as a calibration target, viewing them from the head camera. The automated data capture routine can be completed in as few as 2.5 minutes, followed by an optimization that typically runs in less than a second. The resulting calibration typically has less than three millimeters of residual error, and failure to reach this threshold is exceedingly rare at less than 1 in 100 calibrations.

The following sections of this paper discuss related work, followed by a discussion of the generic calibration system that we have built. We then discuss how this is specifically applied to the Fetch robot, followed by detailed experimental results from several robots.

II. RELATED WORK

There has been significant interest in calibration of mobile manipulators. In [2], Bennett and Hollerbach have calibrated mobile manipulators by forming them into closed kinematic chains. Puskorius and Feldkamp modified the Denavit-Hartenberg coordinates of the Merlin robot and have minimized the errors in the robot-vision model using iterative least-squares [3]. Daniilidis et al. use dual quaternions for efficient hand-eye calibration and compute the transformation between the robots coordinate frame and the sensor [4]. More recently, Pradeep et al. [1] have presented a calibration system designed for the PR2 robot, previously developed by Willow Garage. The PR2 consists of two arms, and several sensors, most on articulated kinematic chains. The primary contribution of their work is a complete formulation of the system such that the calibration is aware of covariances of different sensor models. They formulate the problem using a bundle adjustment approach. Unnikrishnan et al. use a plane constraint to solve the calibration problem between a laser range finder and a camera [5].

RGB-D cameras, such as the Microsoft Kinect [6], which combine a depth camera and color camera are one of the most commonly used sensors on modern mobile manipulators. The most common way of calibrating RGB-D cameras is by using a patterned flat surface, usually a checkerboard [7]. These calibrations usually need a human holding the checkerboard or human intervention to connect the checkerboard to the robot end effector [8]. We propose a method which calibrates the robot and the intrinsic parameters of its RGB-D sensor without using a checkerboard. In [9], Teichman et al. propose a supervised learning approach for the depth sensor calibration problem within a simultaneous localization and mapping (SLAM) framework. In [10], Zhang and Zhang calibrate the depth and colour streams separately to make the calibration more accurate. Significant work has been focused on distortion correction in the depth sensor, typically by way of creating new distortion models besides the typical pinhole camera model [8] [11].

Our approach builds on many lessons learned from the pr2_calibration ROS package, which is the basis of [1]. The later generalization of pr2_calibration to a robot-agnostic calibration ROS package, which one of our authors was responsible for, also yielded many insights into the difficulties of calibrating mobile manipulators. A major short coming of the pr2_calibration package was the slow run-time. A typical calibration involved about 25 minutes of data collection and an additional 25 minutes of optimization. Additionally, the package lacked robustness. Even a single bad data capture could ruin an entire calibration, requiring additional time to be spent calibrating. Further, during the generalization of the pr2_calibration to other robots, it was found that the calibration of the PR2 was typically dominated by having a well-calibrated tilting laser to which all other sensors and kinematic chains could be calibrated.

III. CALIBRATION OVERVIEW

Our calibration system is designed to work with the Robot Operating System (ROS) [12], and is released as a ROS package. The system consists of two phases: the capture of calibration target data and the optimization of the kinematic and other parameters of the robot to minimize the error when re-projecting the calibration targets through various sensors and kinematic chains.

A. Capture

During capture, the robot moves its joints through a series of predefined poses which have been selected as a representative sample of the positions where calibration is required to be valid.

Regardless of the type of calibration target used, the capture phase always records the position of every joint in the robot at each of the capture poses. Many of the free parameters used in the optimization will be the tiny offsets between the position reported as zero by the joint and the actual zero position which minimizes the re-projection error.

In addition to joint position information, one or more calibration target detector modules are needed. As a baseline target detector, we implemented a classical "checkerboard in the hand" method. With this method, a camera detects the position of each of the corner points in the checkerboard. The position of each point is directly measurable by the camera. We assume that the checkerboard is rigidly attached to the robotic arm that is moving it through the workspace. Upon estimating the rigid transformation from the end effector link to the checkerboard, we can re-project where the checkerboard points should be based on the arm kinematics.

Obviously, one of the limitations of such an approach is that the transformation between the checkerboard and the end effector is not known, and must become part of the free parameters of the system. To avoid this issue, the Fetch robot includes four LEDs arranged in a rectangular pattern on the face of the gripper. These LEDs are located at a known transformation from the end effector frame and can be individually turned on and off. Our LED calibration target module cycles the LEDs in succession and monitors the change in intensity of images before and after the LED is cycled on or off. We then accumulate a per-pixel value over several on-off cycles to find where each LED is in the RGB image.

A later addition to the available set of calibration targets included adding support for detecting the ground plane. Here, the head camera is used to find planes which are close in location and orientation to a true ground plane. It then samples points that are inliers in this plane. This method, while requiring some environmental setup, can improve the calibration, especially at longer ranges, since the arm can only provide information about the point cloud at a distance of up to 1 meter.

B. *Optimization*

The kinematics of the robot are modeled in the URDF markup language that is standard in ROS. They are designed such that each of the components is a "link" and each link is connected to a single parent link by a "joint". A root link, called the "base_link" is the canonical link to which all robot links can trace back to. Joints may be either static transformations or actuated joints. The actuated joints are modeled by a static transformation followed by an actuated linear or rotary joint, which has an additional positional offset which can be calibrated.

Our system uses the CERES [13] optimizer to solve the re-projection error minimization problem. CERES is based on creating a "problem" consisting of one or more "free parameters" and one or more "error blocks" that are created for each calibration pose. The optimizer will attempt to find the optimal values for each of the free parameters based on minimizing the residuals from each of the error blocks.

Within our system, free parameters are fairly straightforward. These typically consist of joint angle offsets mentioned above, link offsets which change the static transformation of the joint models, camera intrinsics such as focal length and center pixel location, and camera extrinsics which are dealt with in a manner identical to link offsets.

Error blocks are more involved. For each type of sensor pairing, we need a new error block which uses the data from two or more sensors to re-project the target points as detected by the sensor and compute the residual error between the two or more re-projections.

For instance, the LED detector captures data from the head camera sensor and the "arm sensor". Here, the arm kinematic chain acts as a virtual sensor which can "measure" the position of the LEDs through the forward kinematics of the kinematic chain. For each LED, we use forward kinematics to find the position of the LED in the base_link frame of the robot. We then re-project the position of the LED as detected by the head camera and transform this estimate to the base_link frame so that we can compute the difference between the two estimates, which forms the "residual" or "error". This same method works for the checkerboard calibration, except that there are more corner points than LEDs. We use the Kinematics and Dynamics Library (KDL) [14] for computing the forward kinematics of the robot when evaluating the error blocks.

For the ground plane method, the error block looks quite different. Here, we can again re-project the points through the head camera and into the base_link frame, but we have no way to find the full 3-D estimate of the individual points through the "ground plane". Instead, we can only compare the pose of the points against the plane Z=0 in the robot base_link frame. This is not as strong of a constraint as those created by the LED or checkerboard methods, but especially helps with longer range issues where the arm cannot reach and give data.

An important consideration in the above formulation of the error blocks is that the free parameters consist only of robot kinematics (such as joint offsets, link offsets, camera intrinsics/extrinsics) and virtual robot kinematics (the link connecting a checkerboard to the end effector). The residuals are therefore directly computed by comparing the different re-projections. The PR2 calibration used each checkboard pose as an intermediate against which each re-projection was compared, however, this required that each checkerboard (whose exact pose could not be known) was modeled as a 6-degree of freedom set of free parameters. This meant that adding additional checkerboard poses also added additional parameters to the calibration problem. The poses of the checkerboard or LED target points are not modeled as free parameters in our system, drastically reducing the number of parameters involved, as well as eliminating a large number of parameters that are not independent of the others.

C. *Updating the Robot Model*

As our calibration system is designed to work with ROS, we must therefore update the ROS-compatible robot model, which is held in a URDF formatted file. The URDF file is a plain text XML document, which is easily updated. In addition to the URDF file, the calibration system exports YAML files containing the updated camera intrinsics that will be passed into the camera drivers. Camera parameter drivers are stored as arguments to ROS launch files. On Fetch, we store the calibrated URDF, camera calibration YAML, and updated launch files in the `/etc/ros` folder and configure an `upstart` system job to start the robot drivers with the proper calibration, ensuring that the system is always running the correct calibration when users login and start running commands.

It is important to note that URDF specifies the rotation angles using roll, pitch, and yaw. This does not work well as a direct parametrization in the calibration and so we convert these numbers into angle-axis representations. We note the three angle-axis numbers as "a", "b" and "c" in the results tables later in this paper.

IV. CALIBRATING FETCH

This calibration approach was initially developed for the Fetch mobile manipulator (figure 1), developed by Fetch Robotics. The robot has an arm with seven degrees of freedom, a head with pan and tilt actuators, and a torso lift actuator which moves the arm and head in relation to the base of the robot. The head contains a Primesense Carmine 1.09 short range RGB-D camera.

One of the unique aspects of the Fetch design is that each joint contains an absolute magnetic rotary encoder attached directly to the joint output. Where many robots use optical encoders in combination with an optical flag for zeroing, the absolute magnetic encoder removes the need for joint zeroing during the power-on phase of the robot. In addition, once the robot is calibrated, the calibration will not drift due to missed encoder counts as is a common problem with optical encoders. This does leave open a question of how to zero the joint initially. Many robots use a precision alignment pin, alignment fixture or end stop, Fetch however, forgoes all of those options and simply relies on the calibration. The technician assembling the robot aligns the joint by eye, usually within 2-4 degrees of accuracy. The calibration then finishes the job by determining the true joint zero for each of the seven joints of the arm as well as the head pan and tilt, and the torso joint. Abandoning special alignment pins or fixtures makes assembly of the robot faster as well as making in-field repairs easier as technicians will not need to carry special jigs.

To speed calibration, the gripper contains four calibration LEDs which are blinked in a pattern, and the depth sensor is then able to find these LEDs using a simple threshold algorithm which looks at the transitions. An important requirement for getting a good calibration with the Primesense sensor turned out to be disabling the auto-exposure so that the camera would stay "dark", thus making the LED transitions larger in magnitude.

A. *Configuring Calibration*

Our calibration system is highly modular and designed to be configured via the ROS parameter server. The calibration poses to capture are stored in ROS message formats within a ROS bagfile, while the choice of calibration parameters is stored in YAML formatted files. The YAML file contains which coordinate frame is the "base_link", which parameters to use as free parameters and which error blocks to use.

B. *Selecting Calibration Parameters*

A common issue with other calibration systems is the inability to freely specify a large number of parameters without the calibration diverging. Most calibration systems allow the joint angle offsets to be specified, however few are able to handle link lengths or other link offsets.

With Fetch, we calibrate each of the seven arm joint offsets, torso lift joint offset, and the head tilt joint offset. The interface between the head and the torso is a bracket formed from sheet metal. While lighter and less expensive than a machined block, it also has slightly lower precision. In addition, part and assembly tolerances between the numerous components in the torso and head can "stack-up", causing significant variation between robots. To offset this, our calibration system does a full 6-degree of freedom calibration on the pose of the head_pan_joint relative to the torso_link. This calibrates out any distance or angular offsets that have accumulated through the tolerance stack-up.

Finally, we calibrate the camera intrinsics and extrinsics. This consists of a full 6-degree of freedom pose for the head camera relative to the head tilt link as well as a camera focal length and optical centers.

In this work, we do not calibrate the distortion parameters of the camera. As the exact methods of the internal algorithm of the Primesense sensors is not known, results are inconclusive as to what distortion models are actually robust across a large number of sensors. With the short-range sensor that we currently use, the distortion within the 0.35m to 1.4m range is limited, and is this is the range used for manipulation, we have not found distortion to be a significant issue. The distortion does increase drastically at farther ranges, which presents some issues for using the sensor in applications such as costmap updating while navigating. For this application, we have instead chosen to apply specific undistortion within the navigation software as the sensor is actually outside of its operating range at 2-3m.

C. *Reducing Calibration Time*

One of the primary goals of this work was to create a *fast* calibration system. Initially, the Fetch robot was calibrated using approximately 100 robot poses, which took on average about 8 minutes to capture. While we are typically able to get enough information about each LED position by cycling it off-on-off a single time, there are minimum on-times for each of the LEDs in order assure that the LED is in the expected state when the camera captures a frame. Therefore, the time spent in each capture pose is difficult to reduce and the only reasonable solution to make the system faster was to reduce the number of poses in which we sample the LED positions.

To select a better and smaller set of calibration poses, multiple sets of calibration data were captured across several Fetch robots using the original 100 robot poses. We then randomly sampled several sets of 50 poses from the 100 poses used before and ran each of these sets on multiple robots to find the best calibration pose set. We found that the calibration yields almost the same result. Moving through 50 poses reduces the capture time to 4 minutes. Finally, we did the same reducing in samples to only 25 calibration poses, which achieves similar accuracy of calibration while calibrating in less than two and half minutes. The following section shows the results of a number of calibrations performed.

V. EXPERIMENTAL RESULTS

When calibrating robots, there are two primary concerns with regards to robustness: that the calibration generates a repeatable offset when run multiple times on a given robot, and that the calibration across multiple robots has similar levels of re-projection errors for each parameter being calibrated.

To evaluate the first criteria, we performed calibration on an uncalibrated robot 10 times using each of the 25, 50, and 100 target pose sets. The resulting variance between the offsets calibrated for each of these runs is shown in Table I. As can be seen in Table I, with only 25 poses, we achieved nearly the same accuracy of calibration in less than 2 minutes and 30 seconds.

PARAMETER	100 POSES	50 POSES	25 POSES
Shoulder Pan Joint	5.0320e-06	7.5581e-06	3.1013e-05
Shoulder Lift Joint	1.3376e-07	1.5067e-07	3.4077e-07
Upperarm Roll Joint	6.7229e-08	3.6189e-07	2.2570e-07
Elbow flex Joint	2.8776e-08	2.4610e-08	9.7493e-08
Forearm Roll Joint	2.6495e-07	2.3193e-07	2.9617e-07
Wrist Flex Joint	2.5105e-08	3.6067e-07	2.9536e-07
Wrist Roll Joint	9.1720e-07	2.9379e-06	3.1101e-06
Head Tilt Joint	1.8179e-07	2.0188e-07	5.8326e-07
Focal Length (fx)	2.2515e-05	6.5977e-05	2.9960e-04
Focal Length (fy)	2.6757e-05	7.4985e-05	4.8629e-04
Center cx	1.8679e-06	8.8966e-06	1.3890e-05
Center cy	1.4155e-05	1.4586e-05	7.8062e-05
Z scaling	8.6054e-06	5.4244e-07	2.4875e-07
Z offset	2.8598e-07	1.8925e-05	9.9514e-05
Camera joint x	9.4526e-06	2.0388e-05	1.0470e-04
Camera joint y	1.7924e-07	1.7102e-07	1.3487e-06
Camera joint z	3.6800e-07	1.5980e-06	2.4063e-06
Camera joint a	5.3193e-07	2.7210e-06	5.2710e-06
Camera joint b	1.0078e-07	6.1291e-07	7.2562e-07
Camera joint c	5.8759e-07	2.8224e-06	8.4924e-06
Head pan joint x	1.0595e-08	4.8795e-09	1.1625e-07
Head pan joint y	7.6362e-08	7.8610e-08	3.7168e-07
Head pan joint z	1.2288e-08	2.9837e-08	1.4326e-08
Head pan joint a	1.4428e-08	1.6043e-08	8.7629e-08
Head pan joint b	7.2910e-08	5.3746e-08	3.6787e-07
Head pan joint c	3.3001e-06	1.0922e-05	4.3827e-05

TABLE I

VARIANCE OF FINAL CALIBRATION PARAMETER VALUE ACROSS 10 CALIBRATIONS FOR ALL JOINT AND SENSOR PARAMETERS BEING OPTIMIZED. EACH COLUMN IS THE VARIANCE FOR A PARTICULAR NUMBER OF CAPTURED POSES BEING OPTIMIZED OVER. FOR COORDINATE FRAME CALIBRATIONS, THE FRAME IS REPRESENTED BY A TOTAL OF SIX NUMBERS: THE X, Y AND Z OFFSETS, AND THE ANGLE-AXIS ROTATION DENOTED BY A, B, AND C.

For the second criteria, we performed 10 calibrations on each of three uncalibrated robots. We again evaluate the variance of the final calibrated value (the actual mean calibrated value will be different for each robot). The results of these calibrations are shown in Table II.

Finally, we evaluate the actual re-projection error of the LED points through both the arm kinematic chain and the head camera across multiple robots. The results of this evaluation are shown in Table III. As can be seen, the typical re-projection error for each of the LED points is under 3 millimeters throughout the workspace of the arm.

Figure 2 shows the visualization of a typical uncalibrated robot using the ROS visualization tool, RVIZ. The point cloud depicts the head camera looking at the gripper. The pink shade depicts where the camera thinks the gripper is, which is off by more than 2 cm from where the robot thinks the gripper is based on the kinematic model. After calibration, it can be seen from figure 3 that the robot gripper and the mesh coincide properly, and that the point cloud blends right into the mesh. This is consistent throughout the robot workspace, allowing perception and planning to work together without requiring any sort of visual servoing.

PARAMETER	FETCH1	FETCH2	FETCH3
Shoulder Pan Joint	5.0320e-06	5.2546e-05	7.8241e-06
Shoulder Lift Joint	1.3376e-07	1.5427e-04	1.0073e-07
Upperarm Roll Joint	6.7229e-08	3.3959e-05	1.2355e-07
Elbow flex Joint	2.8776e-08	1.4416e-05	3.7454e-08
Forearm Roll Joint	2.6495e-07	5.2288e-07	1.5986e-07
Wrist Flex Joint	2.5105e-08	3.7200e-05	5.1110e-08
Wrist Roll Joint	9.1720e-07	2.9987e-06	1.1503e-06
Head Tilt Joint	1.8179e-07	2.3881e-05	2.0002e-07
Focal Length fx	2.2515e-05	6.8372e-04	2.1232e-05
Focal Length fy	2.6757e-05	8.6287e-04	2.0084e-05
Center cx	1.8679e-06	5.6485e-06	1.4719e-06
Center cy	1.4155e-05	8.7725e-05	1.7352e-05
Z scaling	8.6054e-06	1.9849e-07	1.5411e-06
Z offset	2.8598e-07	6.9011e-06	9.3637e-06
Camera joint x	9.4526e-06	6.5708e-05	7.6847e-06
Camera joint y	1.7924e-07	6.5242e-07	3.4685e-08
Camera joint z	3.6800e-07	7.5355e-08	5.6251e-07
Camera joint a	5.3193e-07	9.7933e-06	4.4715e-07
Camera joint b	1.0078e-07	2.8095e-05	1.4035e-07
Camera joint c	5.8759e-07	1.5675e-06	5.4968e-07
Head pan joint x	1.0595e-08	6.0103e-06	9.9500e-09
Head pan joint y	7.6362e-08	8.2762e-08	1.3644e-07
Head pan joint z	1.2288e-08	1.1491e-05	2.1953e-08
Head pan joint a	1.4428e-08	1.0040e-07	5.4682e-08
Head pan joint b	7.2910e-08	3.5083e-07	1.6987e-08
Head pan joint c	3.3001e-06	1.6960e-05	9.5201e-06

TABLE II

VARIANCE ACROSS 10 CALIBRATIONS ON EACH OF THREE DIFFERENT ROBOTS FOR ALL JOINT AND SENSOR PARAMETERS BEING OPTIMIZED.FOR COORDINATE FRAME CALIBRATIONS, THE FRAME IS REPRESENTED BY A TOTAL OF SIX NUMBERS: THE X, Y AND Z OFFSETS, AND THE ANGLE-AXIS ROTATION DENOTED BY A, B, AND C.

ROBOT	MEAN RE-PROJECTION ERROR
FETCH1	0.00306514m
FETCH2	0.003058239m
FETCH3	0.002360324m
MEAN:	**0.002827901m**

TABLE III

MEAN ERROR BETWEEN RE-PROJECTING THE LED POINTS THROUGH THE CALIBRATED ARM KINEMATIC CHAIN AND RE-PROJECTING THROUGH THE CALIBRATED HEAD CAMERA AND HEAD KINEMATIC CHAIN.

VI. FUTURE WORK

The current system uses calibration LEDs to detect the gripper. In the future, we intend to investigate methods for directly using the mesh of the arm and gripper within the existing framework. Such methods would provide significantly more data per robot pose, possibly further reducing the number of poses required during the capture phase, which currently dominates time taken to calibrate.

The extremely fast calibration optimization times also lend themselves to possible continuous calibration, in which the robot is always checking and possibly updating the calibration parameters.

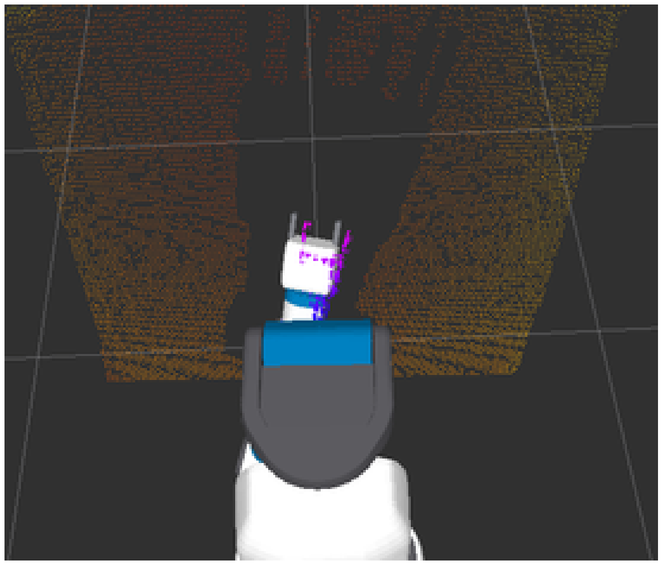

Fig. 2. An uncalibrated robot

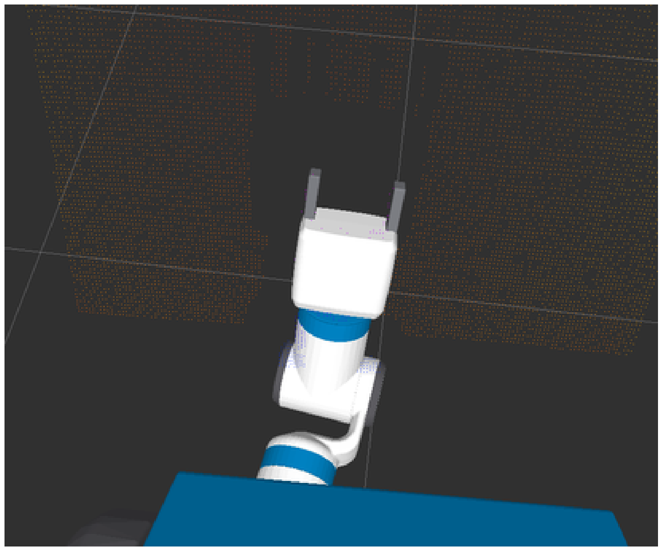

Fig. 3. Calibrated robot

VII. CONCLUSIONS

This paper presented a calibration system designed and developed for the Fetch mobile manipulator. The system was designed to be extensible to other robots, and provides quick and repeatable calibrations of multiple kinematic chains and sensors. It has been used to calibrate dozens of Fetch robots with great accuracy in a timely manner.

Our system for calibrating mobile manipulators is significantly faster and more robust than previous attempts. It rarely fails to properly calibrate a robot.

We have released our calibration system as the robot_calibration package in ROS, which is available in both source form (`http://github.com/mikeferguson/robot_calibration`) and in pre-built debian packages available from `http://www.ros.org`. The configuration required specifically for the Fetch robot can be found in in the fetch_calibration ROS package, at `http://github.com/fetchrobotics/fetch_ros`.

VIII. ACKNOWLEDGEMENTS

The authors would like to thank Derek King for his help in our numerous debugging sessions while developing the calibration system.

REFERENCES

[1] V. Pradeep, K. Konolige, and E. Berger, “Calibrating a multi-arm multi-sensor robot: A bundle adjustment approach,” in *International Symposium on Experimental Robotics (ISER)*, 2010.

[2] D. Bennett and J. Hollerbach, “Autonomous calibration of single-loop closed kinematic chains formed by manipulators with passive endpoint constraints,” *Robotics and Automation, IEEE Transactions on*, vol. 7, no. 5, pp. 597–606, 1991.

[3] G. Puskorius and L. Feldkamp, “Global calibration of a robot/vision system,” in *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, vol. 4, Mar 1987, pp. 190–195.

[4] K. Daniilidis and E. Bayro-Corrochano, “The dual quaternion approach to hand-eye calibration,” in *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, vol. 1, Aug 1996, pp. 318–322 vol.1.

[5] R. Unnikrishnan and M. Hebert, “Fast extrinsic calibration of a laser rangefinder to a camera,” 2005.

[6] “Kinect kinect description,” http://wiki.ros.org/opennilaunch/Tutorials/IntrinsicCalibration.

[7] A. De la Escalera and J. M. Armingol, “Automatic chessboard detection for intrinsic and extrinsic camera parameter calibration,” *Sensors*, vol. 10, no. 3, pp. 2027–2044, 2010.

[8] D. Herrera C., J. Kannala, and J. Heikkil, “Joint depth and color camera calibration with distortion correction,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 10, pp. 2058–2064, Oct 2012.

[9] A. Teichman, S. Miller, and S. Thrun, “Unsupervised intrinsic calibration of depth sensors via slam.” in *Robotics: Science and Systems*. Citeseer, 2013.

[10] C. Zhang and Z. Zhang, “Calibration between depth and color sensors for commodity depth cameras,” in *Multimedia and Expo (ICME), 2011 IEEE International Conference on*, July 2011, pp. 1–6.

[11] S. Clarkson, J. Wheat, B. Heller, and S. Choppin, “Assessing the suitability of the microsoft kinect for calculating person specific body segment parameters,” in *Computer Vision-ECCV 2014 Workshops*. Springer, 2014, pp. 372–385.

[12] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, “ROS: an open-source robot operating system,” in *Open-Source Software workshop of the International Conference on Robotics and Automation (ICRA)*, 2009.

[13] S. Agarwal, K. Mierle *et al.*, “Ceres solver,” 2012.

[14] R. Smits, “KDL: Kinematics and Dynamics Library,” http://www.orocos.org/kdl.