Improving Hand-Eye Calibration for Robotic Grasping and Manipulation

Benjamin Axelrod
iRobot Corporation
Bedford, Massachusetts, USA
baxelrod@irobot.com

Wesley H. Huang
iRobot Corporation
Bedford, Massachusetts, USA
whuang@irobot.com

Hand-eye calibration is an important component of robotic systems that perform manipulation and grasping tasks. However, calibration is often an onerous process – there are many parameters that must be estimated for the sensors and manipulators, resulting in a high-dimensional non-linear estimation problem. While it is easy to obtain an approximately correct hand-eye calibration, reducing the error further requires increasingly greater effort. We have developed a simple method for increasing the accuracy of an approximately correct hand-eye calibration. This method does not require any external instrumentation and is unique in that it applies a transformation to sensed object locations to produce commanded end-effector locations. This method has been applied to the robot for the DARPA ARM-S program, consisting of a 7 DOF arm and a sensor head mounted atop a 4 DOF neck. We describe the theory of our approach, our implementation, and experimental results.

Keywords: hand-eye calibration, robot calibration, grasping, manipulation.

I. INTRODUCTION

Hand-eye calibration is an important part of robotic systems that perform autonomous grasping and manipulation tasks. Sensors and actuators need to be calibrated individually and between each other to achieve the accuracy required to grasp or manipulate objects detected by the robot's sensors.

A typical robot system for grasping and manipulation has a high degree of freedom (DOF) arm and one or more imaging sensors (cameras or 3D rangefinders) on an actuated neck. In the traditional approach to grasping and manipulation, the sensors identify an object in the robot workspace and compute its pose in a world coordinate frame. Next, an end-effector pose relative to the object is chosen, the inverse kinematics (IK) is applied to compute joint angles for the arm, and a motion planner determines a collision-free path from the current arm configuration to the goal configuration. Different control methods to move the arm to the object can be employed, ranging from open loop control to visual servoing and haptic feedback. The starting point for all of these approaches is an accurate hand-eye calibration.

This work was supported by the DARPA Autonomous Robotic Manipulation, Software Track program under contract number W91CRB-10-C-0127. Distribution Statement A -- Approved for Public Release, Distribution Unlimited. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

While there have been other approaches to grasping and manipulating objects that do not require a hand-eye calibration, such as learning [6] [7], and uncalibrated visual servoing [8] [11], the hand-eye calibration is at the center of most robotic systems. The hand-eye calibration problem is to calibrate the neck/sensors and the arm in a consistent coordinate frame, so that the end-effector can be positioned accurately with respect to a sensed object.

It is easy to formulate an approximately correct hand-eye calibration: a geometric model of the robot system can be measured or derived from CAD, and there are standard camera models and calibration routines readily available. However, it is difficult to achieve high accuracy due to the many sources of error. Parameters of geometric models can be incorrect due to measurement, machining, or assembly variances. There can be unmodeled physical phenomena (such as compliance) that affect the actual pose of the arm or neck. Sensor models are approximate, and camera lenses can be imperfect.

We have developed a method to improve the accuracy of a hand-eye calibration that requires a limited amount of data collection and does not require an external position measurement system. The basis of our approach is to learn a mapping from the perceived object coordinates to the coordinates for commanded arm positions that results in more accurate end-effector positioning.

There has been much previous work on robot calibration. Older work focused on adjusting the Denavit-Hartenberg (DH) parameters of industrial robot arms. Typically, measurement of the robot's end-effector was done with a complicated external setup such as a theodolite [9] [18], probe [3], or custom device [15]. The manipulator kinematics are then adjusted according to these measurements. Closed-loop kinematic calibration is another technique for manipulator calibration [1] [10]. An open kinematic chain can be turned into a closed kinematic chain by rigidly fixing the end-effector to the world in some fashion. The arm is moved through its redundant configurations and the arm kinematics can be calibrated with only joint angle readings. These techniques provide good accuracy for the arm's kinematics, however don't take into account the neck kinematics or arm compliance.

Attaching a camera to a kinematic chain creates what is typically called the eye-in-hand problem. By viewing a static scene from multiple angles, the transformation between the

views can be determined [4] [12] [14] [16] [17]. Again, this calibration only covers part of the problem and does not take compliance into account.

A modern approach to the hand-eye calibration problem is presented in [13]. Here, multiple sensors on kinematic chains are calibrated w.r.t each other using a bundle approach. This requires the robot to hold a calibration target in its manipulator for viewing with the sensors. However, the arm configurations required for this may be far from the configurations used in the robot's standard operation. Furthermore, the authors admit that for precision tasks such as autonomously plugging-in a power cord, a local calibration offset is used that accounts for belt stretch and other unmodeled errors.

II. Approach

If we used a precision external positioning system, it would be possible to obtain hand-eye calibration by decoupling the arm calibration from the neck and sensor calibration – each would be calibrated to a world coordinate system measured by a precision external positioning system. While it is still a difficult problem to achieve high accuracy, it is simpler than doing a combined hand-eye calibration. However, our goal was to improve the calibration of our system without the use of a precision external positioning system, as such a system is often too expensive or unavailable.

By using the geometric model and nominal parameters of the robot, we can easily develop a calibration that is approximately correct. We found that our nominal arm calibration was reasonably accurate in the orientation of the end-effector, but that the position was not accurate enough for many grasping and manipulation tasks. Therefore, we focused on correcting only the position in our hand-eye calibration.

Our intuition was that since the arm calibration and neck-sensor calibration are approximately correct, we can learn a mapping from sensed object positions to the commanded positions for the arm which will result in higher accuracy.

A. Foundations

The first step when working with a robotic arm is to formulate the forward kinematics, using the geometric model of the arm to map arm joint angles $\vec{q}$ to an end-effector position $\vec{x}$:

$$\widetilde{FK}(\vec{q}) \rightarrow \vec{x} \tag{1}$$

Though forward kinematics usually produces a pose (position and orientation), we are just interested in the position at this time. We discuss incorporating components of orientation into our approximation later in this section.

We use the tilde notation to indicate that the function $\widetilde{FK}(\vec{q})$ is an imperfect model of the real robot. It is imperfect for a number of reasons. First, our parameters for the arm geometry are not exactly correct; joint offsets may be manually measured or may be from CAD (and due to machining or assembly variances, not reflect the exact geometry). Second, there are non-geometric phenomena that can cause the actual end-effector pose to differ from the geometric ideal. We consider $\vec{q}$ in Equation 1 to be the measured joint angles, rather than the actual joint angles. The difference between measure and actual joint angles may be due to unmodeled compliance in the arm links, backlash, etc. (In our case, the encoders are located at the motors, not at the joints, so cable stretch affects the actual joint angles.) These non-geometric phenomena may be configuration dependent, but we will assume that they are independent of the history of the arm motion.

We will notate the ideal or actual mapping from measured joint angles to end-effector position as:

$$FK(\vec{q}) \rightarrow \vec{x} \tag{2}$$

Note that for the reasons described above, $\widetilde{FK}(\vec{q}) \neq FK(\vec{q})$. The arm calibration problem is to estimate the geometric parameters of the arm to make $\widetilde{FK}(\vec{q})$ approximate $FK(\vec{q})$ as closely as possible.

We also typically create inverse kinematics (IK) based on the geometric model to calculate the joint angles $\vec{q}$ to move the end-effector to a desired position $\vec{x}$:

$$\widetilde{IK}(\vec{x}, \vec{\alpha}) \rightarrow \vec{q} \tag{3}$$

Here, $\vec{\alpha}$ represents orientation parameters, as well as parameters to resolve any redundancy or non-uniqueness in the arm configuration.

Note that $\widetilde{FK}\left(\widetilde{IK}(\vec{x}, \vec{\alpha})\right) = \vec{x}$ since both functions are based on the same (imperfect) geometric model. However, $FK\left(\widetilde{IK}(\vec{x}, \vec{\alpha})\right) \neq \vec{x}$ since $\widetilde{IK}(\cdot)$ only considers the arm geometry and does not include all actual physical effects represented in $FK(\cdot)$. In other words, if we compute $\vec{q}$ using our imperfect inverse kinematics, and command the arm to those joint angles, the arm does not go to the exact desired position in the world frame.

Similarly, there is the actual physical process of how the object is projected on to the image plane and where the neck has positioned the sensors. We represent the actual or ideal mapping from the object position $\vec{x}$ to sensor coordinates $\vec{u}$ as:

$$F_{imaging}(\vec{q}_{neck}, \vec{x}) \rightarrow \vec{u} \tag{4}$$

where $\vec{q}_{neck}$ is the measured neck joint angles. Again, we assume that the actual position of the neck is dependent on only $\vec{q}_{neck}$ and not on the history of the neck motion.

We can formulate a mapping from sensor coordinates to world coordinates using our models of the neck and of the sensor. We will represent this mapping by:

$$\tilde{F}_{sensor}(\vec{q}_{neck}, \vec{u}) \rightarrow \vec{x} \tag{5}$$

However, this function is imperfect (hence the tilde notation). The neck model has slightly incorrect geometric parameters, and there are unmodeled phenomena that affect the actual positioning of the sensor. Furthermore, the sensor model is generally an approximation of the actual physical imaging process.

Consequently, $\tilde{F}_{sensor}\left(\vec{q}_{neck}, F_{imaging}(\vec{q}_{neck}, \vec{x})\right) \neq \vec{x}$. In other words, if we calculate an object's world position from the perceived camera coordinates, our result will not generally be correct. The sensor calibration problem is to estimate geometric parameters for the neck and parameters for a sensor model to make $\tilde{F}_{sensor}\left(\vec{q}_{neck}, F_{imaging}(\vec{q}_{neck}, \vec{x})\right) = \vec{x}$ as closely as possible.

B. Calibration mapping

In a typical grasping or manipulation task, there is an object at $\vec{x}_{world}$ in the world frame, and the following is performed:

1. Sense the object:

$$\vec{u} = F_{imaging}(\vec{q}_{neck}, \vec{x}_{world}) \tag{6}$$

2. Calculate its position:

$$\vec{x}_{sensed} = \tilde{F}_{sensor}(\vec{q}_{neck}, \vec{u}) \tag{7}$$

3. Calculate joint angles:

$$\vec{q} = \widetilde{IK}(\vec{x}_{sensed}, \vec{\alpha}) \tag{8}$$

4. Move the arm:

$$\vec{x}_{arm} = FK(\vec{q}) \tag{9}$$

However, $\vec{x}_{sensed} \neq \vec{x}_{world}$ and $\vec{x}_{sensed} \neq \vec{x}_{arm}$. The typical hand-eye calibration problem is to estimate parameters for $\tilde{F}_{sensor}(\cdot)$ and $\widetilde{IK}(\cdot)$ so that $\vec{x}_{sensed} = \vec{x}_{world} = \vec{x}_{arm}$.

Our goal is simply for $\vec{\mathrm{x}}_{arm} = \vec{\mathrm{x}}_{world}$, i.e. for the arm to move to the actual object position. (Note that it is not necessary for $\vec{\mathrm{x}}_{sensed} = \vec{\mathrm{x}}_{world}$, and in fact, we don't care about this condition.) We can achieve this goal by transforming the sensed object position $\vec{x}_{sensed}$ before feeding it into $\widetilde{IK}(\cdot)$.

We can create such a mapping by placing an object in the workspace with the arm. From the commanded joint angles, we compute $\vec{\mathrm{x}}_{arm}$ using the forward kinematics. For reasons that will soon be apparent, we want to place the object with joint angles computed with the inverse kinematics from $\vec{\mathrm{x}}_{arm}$. Because the arm physically placed the object, we know $\vec{\mathrm{x}}_{arm} \equiv \vec{\mathrm{x}}_{world}$. We then sense the object to get $\vec{x}_{sensed}$. If, at some future time, we observe an object at $\vec{x}_{sensed}$, then we know we should command the arm to the position $\vec{\mathrm{x}}_{arm}$. (Or, rather to the joint angles computed from $\vec{\mathrm{x}}_{arm}$ using the inverse kinematics.)

However, $\vec{x}_{sensed}$ is dependent on $\vec{q}_{neck}$, i.e., the sensed position of the object will be different for different neck poses because of our neck model is imperfect. Also, $\vec{\mathrm{x}}_{arm}$ is dependent on $\vec{\alpha}$, i.e., to compute the correct joint angles for the arm, we need to give the inverse kinematics a position that is different for different orientations of the end-effector, etc. Therefore, we would have to learn the mapping:

$$M(\vec{x}_{sensed}, \vec{q}_{neck}, \vec{\alpha}) \rightarrow \vec{x}_{arm} \tag{10}$$

This would require sampling multiple positions in the arm's workspace with multiple neck configurations, multiple end-effector orientations and multiple redundancy-resolving parameters. The combinatorics preclude sampling enough points to approximate this mapping accurately.

However, we can make some simplifications to eliminate these additional parameters. To eliminate the neck angles, we assume that the neck angles are determined as a function of the object position: $\vec{q}_{neck} = f(\vec{x}_{world})$. In our case, we always point the center of the camera image at the object, but in general the neck configuration can be determined in any way that uniquely resolves $\vec{q}_{neck}$.

We eliminate $\vec{\alpha}$ by fixing the orientation of the end-effector and by utilizing a computation to uniquely resolve the arm's redundancy as a function of the object's position: $\vec{\alpha} = f(\vec{x}_{world})$. The mapping now becomes

$$M(\vec{x}_{sensed}) \rightarrow \vec{x}_{arm} \tag{11}$$

This is combinatorially much more tractable and can be learned by sampling over a range of positions in the robots workspace.

With the simplifications above, the procedure for using our calibration mapping replaces Step 3 above (Equation 8) with:

$$\vec{q} = \widetilde{IK}\left(M(\vec{x}_{sensed})\right) \tag{12}$$

C. Incorporating orientations

This strategy works well when one specific end-effector orientation is commonly used. In our system, many objects are grasped from above with the end-effector in a vertical top-down orientation, so a single mapping with this orientation can be learned and applied.

However, there are also many objects that we must grasp from the side, and we may want to use any wrist orientation in the plane. In this case, we cannot completely eliminate the $\vec{\alpha}$ parameter as we did above. Instead, we split $\vec{\alpha}$ into two parts, $\vec{\alpha}'$ and $\vec{\alpha}''$, the former containing the parameters that we wish to control (for example, a single yaw angle), and the latter containing orientation and redundancy-resolving parameters that can be computed uniquely from the object position: $\vec{\alpha}'' = f(\vec{x}_{world})$. The mapping from Equation 10 then becomes:

$$M(\vec{x}_{sensed}, \vec{\alpha}') \rightarrow \vec{x}_{arm}, \tag{13}$$

This mapping can be approximated by sampling over a range of positions in the robot's workspace and the parameters in $\vec{\alpha}'$.

III. Implementation

We implemented this calibration procedure in the DARPA Autonomous Robotic Manipulation – Software track (ARM-S) program. The ARM-S robot consists of a Barrett Technology 7 DOF WAM arm and a three fingered BH-280 hand with 4 DOF. The sensor head contains a Point Grey Bumblebee2 stereo camera pair, a Swiss Ranger SR4000, and a Prosilica GC2450C color camera. The sensor head is mounted atop two pan-tilt units (DirectedPerception PTU-D46-17P70T and PTU-

D48), creating a 4 DOF neck. A photograph of our system is shown in Figure 1.

Our approach to learning the calibration mapping M is to collect samples over the robot's workspace and end-effector orientations. These data are used in a k-nearest neighbor estimator to calculate values of M. This section provides details of our data collection and how our calibration is applied.

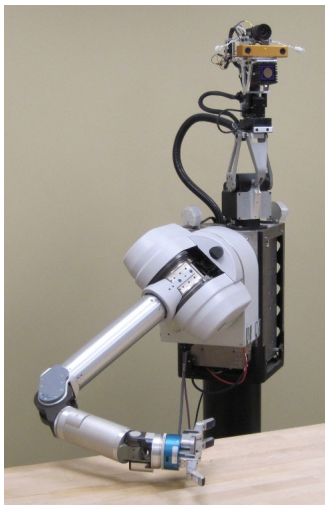

Figure 1. ARM-S robot system

Figure 2. Calibration Cube for Barrett Hand

A. *Collecting Calibration Data*

We collected calibration data by placing a special object at multiple locations on the table in front of the robot. This object is special in that it provides a convergent grasp for the robot's manipulator, so that the robot can autonomously grasp and release the object dozens of times while accurately placing the object each time. In addition, the object should be easy for the perception to accurately localize.

Our calibration object is shown in Figure 2. It is a Styrofoam cube, 0.126 m on each side, with channels cut for each finger, and a printed texture pattern affixed its sides to simplify detection via stereo vision. We perform an iterative closest point (ICP) [2] match on the stereo point cloud data to determine the position of the cube.

We believe one important aspect of our calibration method is the use of a calibration object, rather than a calibration target rigidly attached to the arm. Placing and detecting the pose of a calibration object more closely matches the manner in which the calibration will be used. A calibration target generally requires the end-effector to be in a different pose for calibration than it would be for regular grasping or manipulation tasks. For example, a calibration target provided with our system was a checkerboard pattern that required the wrist to point towards the camera to see the target. While it would be possible to put the pattern to be detected on the other side of the target (so the end-effector would be pointing away from the camera), this would require the target pattern to be offset from the end-effector; also, the arm might occlude part of the calibration target making accurate measurements of the target difficult.

The procedure for data collection is as follows:

1. A position in the robot's workspace is chosen, and the inverse kinematics are used to compute the commanded joint angles for the arm. We actually choose a position slightly above our table, move the arm to that position, and perform a guarded move down to place the cube on the table. We use the forward kinematics on the measured joint angles and record the position of the cube center as well as the orientation of the end-effector.
2. The cube is released, and the arm is removed from the scene. This ensures that the arm does not interfere with the perception of the cube. It is important that the cube does not move while the hand is releasing and moving away from the cube.
3. The cameras are pointed at the approximate cube location (from FK) and its point cloud is segmented from the table plane. The neck angles are then readjusted to center the object in the camera view, and a new point cloud is collected. Now the pose of the cube is then determined through ICP and recorded. This two step process is what allows us to remove the neck parameters from the mapping. Note that we have a good initial seed for the ICP algorithm because we know its approximate pose from forward kinematics and that is flat on the table.
4. The arm can then move back to the cube, grasp it, and place it at another location on the table. This can be done quite accurately by replaying in reverse the trajectory used when moving away from the cube.

The cube is placed at locations across the robot's workspace on the table, and this process is repeated for several different wrist orientations. In our case, the data for top-down grasps is gathered autonomously, while the data for side grasps is gathered semi-autonomously. This is due to the difficulty of having the hand release and move away from the light Styrofoam cube without changing its pose.

B. *Applying the calibration*

A calibration dataset consists of: the object positions from perception ($\vec{x}_{sensed}$), the object positions from the arm ($\vec{x}_{arm}$), and may include parameters for end-effector orientation ($\vec{\alpha}'$). The process to use the calibration data is:

1. An object is perceived on the table at $\vec{x}_{sensed}$.
2. We determine the desired wrist orientation $\vec{\alpha}'$.
3. We then search the data for the k nearest neighbors to $\vec{x}_{sensed}$. Only points that have a wrist orientation

within 45 degrees to our desired orientation are considered. We compute a weighted average of the offset $\vec{x}_{arm} - \vec{x}_{sensed}$ over the k nearest neighbors, weighting each by the distance to the sampled data point. This weighted offset is added to the sensed object $\vec{x}_{sensed}$ to produce our estimate of $\vec{\mathrm{x}}_{arm}$.

4. This $\vec{\mathrm{x}}_{arm}$ is then used with our inverse kinematics to compute commanded joint angles for the arm.

Note that in addition to adjusting the object's location, the positions of obstacles in the world will also need to be adjusted to avoid collisions.

IV. Results

We have found that an estimate of M from a reasonably sparse sampling of the workspace suffices to make a significant improvement in the accuracy of our hand-eye calibration.

We collected one dataset of 11 points for the top-down wrist orientation and a second dataset with 23 points for side grasps. Only three orientations were used for side grasps: front, and 90 degrees to the right and left. Of these 23 points, 11 are right-side grasps, 7 are left-side grasps, and 5 are front-side grasps. The different numbers of samples are because the arm cannot reach some positions with some orientations. The calibration cube (Figure 2) was placed at locations on a rectangular grid with a spacing of 0.25 meters left to right, and 0.2 meters front to back. The entire calibration procedure takes less than 1 hour to run.

We ran two experiments. In the first experiment, only the top down orientation was considered. Fifteen locations on the table were tested at a spacing half as much as the calibration data. Figure 3 illustrates both the top-down calibration data locations as well as the tested locations. In this figure, the black dot is $\vec{x}_{sensed}$ from the collected data, and the red line extends to $\vec{\mathrm{x}}_{arm}$. The unfilled circles are $\vec{x}_{sensed}$ at the test locations and the black line extends from these to the computed $\vec{\mathrm{x}}_{arm}$ estimate. Note that while we record and correct 3D positions, it is typically the error in the X-Y plane that requires the most correction for grasping and manipulation tasks.

In this experiment, the arm was commanded to touch a ball 0.066 meters in diameter with a transparent support. The Cartesian distance from the palm to the ball was measured by moving the ball to the palm, then measuring the displacement of the ball's support with millimeter graph paper. The accuracy of these measurements is probably about ±0.5 cm.

Without calibration, the average error was 6.8 cm. With calibration, the average error was 2.0 cm. A graphical representation of the error can be seen in Figure 4. Here, the blue line to the square is the measured error with no calibration, and the green line to the circle is the measured error when our calibration is applied.

In addition to showing improved accuracy with our calibration procedure, this experiment demonstrates that the course data collection is sufficient

In the second experiment, a single location on the table was used, and the arm was commanded to touch the side of the ball at 9 different orientations in the plane, from +90 degrees (right side grasp) to -90 degrees (left-side grasp), at 22.5 degree increments. The calibration data and test location can be seen in Figure 5. Similar to Figure 3, the black dot is $\vec{x}_{sensed}$ from the collected data, and the colored lines extend to $\vec{\mathrm{x}}_{arm}$ for the 3 orientations. Green indicates front orientation, blue indicates right side orientation, and yellow indicates left side orientation. The unfilled circle is $\vec{x}_{sensed}$ at the test location.

Without calibration, the average error was 6.9 cm, and with calibration the average error was 1.9 cm as seen in Figure 6. Again, the green line to the circle is the measured error when our calibration is applied, and the blue line to the square is the measured error with no calibration.

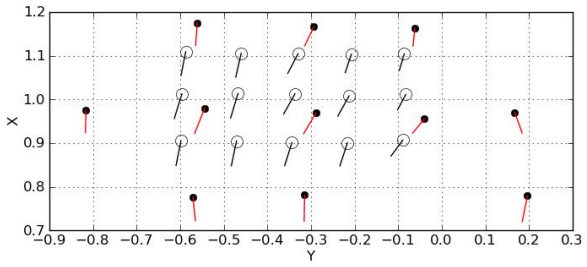

Figure 3. Top-down calibration data (●) and test locations (○). The lines indicade the measured / estimated position offset.

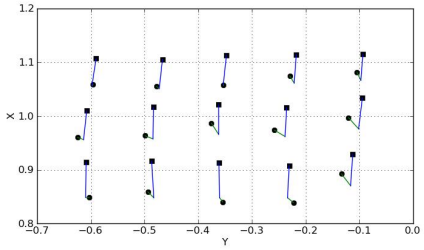

Figure 4. Top-down results without calibration (■) and with calibration (●). Lines indicate distance from actual object to palm center.

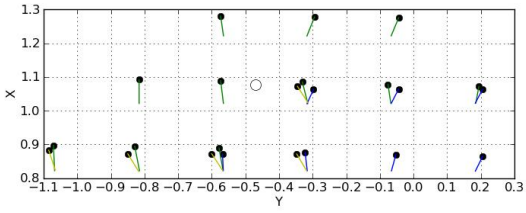

Figure 5. Side-grasp calibration data (●) and test location (○).

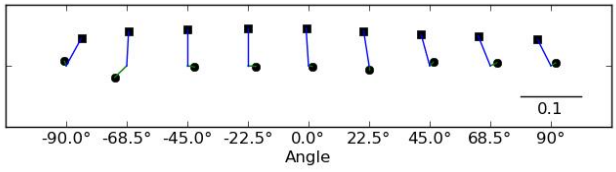

Figure 6. Side-grasp results without calibration (■) and with calibration (●). Lines indicate the offset from actual object position to the palm center.

We attribute the 2 cm errors seen with calibration to be due to the gross errors in our FK. These errors are highly configuration dependant. Indeed, the test locations that had the largest errors used arm configurations significantly different than the ones used to gather data. This FK error also causes errors when gathering the data. For example the end-effector may not be perfectly vertical when placing the cube. This misalignment causes the real world placement of the cube to be far from the FK computation of the cube. So, $\vec{x}_{arm} \neq \vec{x}_{world}$, which breaks our assumptions.

V. CONCLUSION

We have developed a new, simple calibration procedure to increase the accuracy of robotic manipulation systems. This method requires a modest amount of data collection, and it does not require an external position measurement system. Our approach starts with approximately correct arm and neck-sensor calibrations, from standard camera calibration methods and from geometric models with nominal parameters from CAD or other measurements. We learn a mapping from perceived object coordinates to commanded arm coordinates so that the end-effector reaches the object position.

This mapping is learned by having the robot place a calibration object at a number of locations in its workspace and sensing the object's position. This approach utilizes arm configurations and perception algorithms very similarly to the way they are used in grasping and manipulation tasks, unlike many other hand-eye calibration approaches.

We have implemented our approach on the DARPA ARM-S robot system, consisting of a 7 DOF arm and a sensor head mounted atop a 4 DOF neck. We found that sampling calibration data relatively sparsely over the robot's workspace and end-effector orientations was sufficient to learn the mapping that produced significant improvement in the accuracy of our system. We conducted two experiments to evaluate our approach that showed a reduction in the calibration error from an average of 6.8 cm to 2.0 cm.

REFERENCES

[1] Bennett, D.J.; Hollerbach, J.M.; "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, Oct 1991

[2] Besl, P.J.; McKay, H.D.; "A method for registration of 3-D shapes," Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.14, no.2, pp. 239-256, Feb 1992

[3] I-Ming Chen; Guilin Yang; Chee Tat Tan; Song Huat Yeo; "Local POE model for robot kinematic calibration," Mechanism and Machine Theory, Volume 36, Issues 11-12, November 2001, Pages 1215-1239

[4] Daniilidis, K.; "Hand-eye calibration using dual quaternions," International Journal of Robotics Research, vol. 18, 286-298 (1998)

[5] Driels, M.R.; Swayze, W.E.; "Automated partial pose measurement system for manipulator calibration experiments," Robotics and Automation, IEEE Transactions on , vol.10, no.4, pp.430-440, Aug 1994

[6] Fuentes, O.; Nelson, R.C.; "Learning dextrous manipulation skills using multisensory information," Multisensor Fusion and Integration for Intelligent Systems, 1996. IEEE/SICE/RSJ International Conference on , vol., no., pp.342-348, 8-11 Dec 1996

[7] Graefe, V.; "Calibration-free robots for cost-effective, dependable and robust automation," Automation and Logistics, 2008. ICAL 2008. IEEE International Conference on , vol., no., pp.262-267, 1-3 Sept. 2008

[8] Jagersand, M.; Fuentes, O.; Nelson, R.; "Experimental evaluation of uncalibrated visual servoing for precision manipulation," Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on , vol.4, no., pp.2874-2880 vol.4, 20-25 Apr 1997

[9] Judd, R.; Knasinski, Al.; "A technique to calibrate industrial robots with experimental verification," Robotics and Automation. Proceedings. 1987 IEEE International Conference on , vol.4, no., pp. 351- 357, Mar 1987

[10] Meggiolaro, M.A.; Scriffignano, G.; Dubowsky, S.; "Manipulator Calibration Using a Single Endpoint Contact Constraint." Proceedings of the 26th Biennial Mechanisms and Robotics Conference of the 2000 ASME Design Engineering Technical Conferences, Baltimore, MD, September 2000

[11] Munoz, L.A.; "Robust dexterous manipulation: a methodology using visual servoing," Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on , vol.1, no., pp.292-297 vol.1, 13-17 Oct 1998

[12] Park, F.C.; Martin, B.J.; "Robot sensor calibration: solving AX=XB on the Euclidean group," Robotics and Automation, IEEE Transactions on , vol.10, no.5, pp.717-721, Oct 1994

[13] Pradeep, V.; Konolige, K.; Berger, E.; "Calibrating a multi-arm multi-sensor robot: A bundle adjustment approach." In Int. Symp. on Experimental Robotics (ISER), 2010

[14] Puskorius, G.; Feldkamp, L.; "Global calibration of a robot/vision system," Robotics and Automation. Proceedings. 1987 IEEE International Conference on , vol.4, no., pp. 190- 195, Mar 1987

[15] Renders, J.-M.; Rossignol, E.; Becquet, M.; Hanus, R.; "Kinematic calibration and geometrical parameter identification for robots," Robotics and Automation, IEEE Transactions on , vol.7, no.6, pp.721-732, Dec 1991

[16] Shiu, Y.C.; Ahmad, S.; "Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form AX=XB," Robotics and Automation, IEEE Transactions on , vol.5, no.1, pp.16-29, Feb 1989

[17] Tsai, R.Y.; Lenz, R.K.; "A new technique for fully autonomous and efficient 3D robotics hand/eye calibration," Robotics and Automation, IEEE Transactions on , vol.5, no.3, pp.345-358, Jun 1989

[18] Whitney, D.E.; Lozinski, C.A.; Rourke, J.M.; "Industrial Robot Forward Calibration Method and Results," J. Dyn. Sys., Meas., Control 108, 1 (1986)