Closed-loop Servoing using Real-time Markerless Arm Tracking

Matthew Klingensmith, Thomas Galluzzo, Christopher Dellin, Moslem Kazemi, J. Andrew Bagnell, Nancy Pollard
{mklingen, tgalluzzo, cdellin, moslemk, dbagnell, nsp} @ andrew.cmu.edu

***Abstract*— We present a simple, efficient method of real-time articulated arm pose estimation using stochastic gradient descent to correct unmodeled errors in the robot's kinematics with point cloud data from commercial depth sensors. We show that our method is robust to error in both the robot's joint encoders and in the extrinsic calibration of the sensor; and that it is both fast and accurate enough to provide real-time performance for autonomous manipulation tasks. The efficiency of our technique allows us to embed it in a closed-loop position servoing strategy; which we extensively use to perform manipulation tasks. Our method is generalizable to any articulated robot, including dexterous humanoids and mobile manipulators with multiple kinematic chains.**

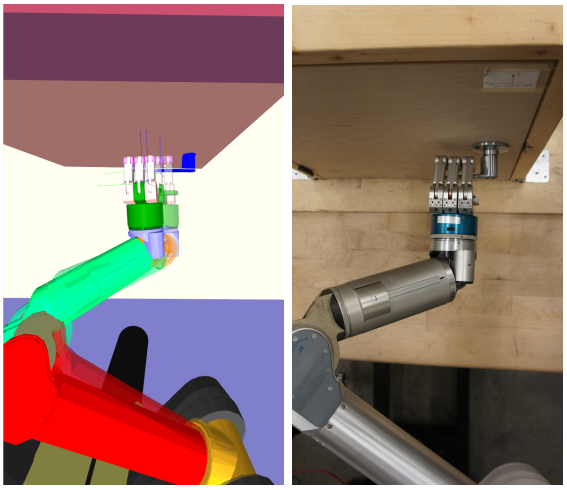
I. Introduction

In robotic manipulation, uncertainty remains a fundamental challenge. Robots struggle with inaccurate or incomplete data from sensors resulting in both external error (with respect to objects in the robot's environment), and internal error (with respect to the robot's kinematic configuration). In industrial settings, both types of error can be compensated for by altering the robot's environment to reduce uncertainty in object pose and by constructing industrial robot arms which are highly rigid and precise. However in uncontrolled settings, robots must rely on noisy sensors to determine object poses. Further, industrial robot manipulators are often unsuitable for autonomous robotics in uncontrolled settings due to their weight, cost, and extreme rigidity. What results is that autonomous robots in real-world settings often have much more external and internal error than their industrial counterparts. Since algorithms in robotic manipulation (such as grasp planners or motion planners) often assume that the states of the environment and robot are known with certainty, significant error in object pose or robot configuration may cause autonomous manipulation tasks to fail.

In this domain, uncertainty is in part accounted for by *calibration*. That is, an offline procedure may be used to estimate unknown or poorly modeled parameters of the system and then store those parameters to correct for errors during task execution. When the state of the system is defined fully by these parameters, calibrating them results in error reduction. While useful, calibration is limited to estimating simple parameters of the system that do not change during task execution (such as fixed camera parameters, or encoder offsets). However, some parameters may be difficult to model and estimate. Partiularly, in non-rigid robots with lightweight series-elastic or cable-driven motors, the dynamics of the system can be very complex. Motors may slip, metal components may expand and contract with heat, cables may become untensioned or stretched, and external forces may contort the system in ways which are difficult to predict (Fig. 2). Due to these factors, the accuracy of calibration generally decreases over time, and the parameters will have to be re-calibrated (an often tedious exercise).

For these reasons, it becomes desirable to augment calibration with real-time *estimation* of system state by analyzing sensor data. In this way, we can circumvent unmodeled system parameters not accounted for during calibration by

All authors are associated with the Carnegie Mellon Robotics Institute Pittsburgh, PA



(a) (b)

Fig. 1. The robot opens a door using arm pose estimation. Subfigure (a) shows the currently estimated state of the robot (transparent), and the pose of the robot given by the joint encoders (solid). The true pose of the arm (b) is closer to the estimated state of the robot from our method than to the state determined from the joint encoders alone.

(a) (b)

Fig. 2. Twenty inverse kinematics solutions for a Barret WAM end effector pose (a) are shown in composite (b). Cable slack and other mechanical imperfecions cause large differences in the resulting pose of the end effector.

updating our belief about the state directly.

In this work, we aim to perform common manipulation tasks by reducing errors between observed sensor data and the robot's kinematic model during the execution of a control loop. Our method is both easy to implement and extremely efficient; and it obtains much better accuracy than the joint encoders alone. Unlike existing pose estimation algorithms in robotic manipulation, our algorithm is fast enough to run during a closed-loop servo motion, requires no further data than the robot's joint encoders and a depth image, and makes no independence or Gaussian assumptions about the underlying statistical distribution of the error. We show that by using the same sensor to perceive both the environment and the robot's arm, we are able to successfully complete tasks autonomously which would otherwise fail.

II. Related Work

Tracking of articulated bodies has been well studied and employed in human motion capture research as well as robotic manipulation tasks. The general approach is to regulate to zero the error between the corresponding features extracted from the sensory input and the rendered model of the articulated bodies. In this sense, each of the following techniques are *model-based*, in that they rely on an explicit model of the state. The existing techniques differ in terms of the input data and how the error between the measurements and the rendered model is applied to approximate and correct the models. Here we relate only to some of the most relevant works in human motion capture and robotic manipulation.

In the context of human motion capture, articulated body tracking is employed to estimate the full configuration of the limbs (see [8] for an overview). Commercially available marker-based approaches [9] rely on placing reflective markers or fiducial on the articulated bodies, which limits their usage in robotic manipulation, where it is often not possible to attach such devices to articulated bodies in the scene.

To achieve markerless tracking, efforts have been devoted to use a variety of sensory inputs, e.g., 2-D features, contours, color, and depth data. In a closely related work, Grest *et al.* [4] perform a non-linear optimization to estimate the limbs' degrees of freedom based on depth data correspondence efficiently computed using ICP. We converged to a similar approach which uses online stochastic gradient descent rather than iterative Gauss-Newton least squares regression to minimize the ICP loss function. Additionally, in our problem domain, the objective is not to precisely estimate the articulated bodies degrees of freedom, but to calculate the appropriate joint offsets to minimize the error between all visible parts of the robot and the measured data. This effectively allows us to reduce the positioning error at the end-effector in a closed-loop servoing task, e.g., opening a door.

In the context of manipulation, Krainin *et al.* [7] use arm pose estimation to model objects held in the robot's hand. In their work, they make use of the Articulated ICP (AICP) [11] algorithm for pose estimation, which relies on LevenbergMarquardt (LM) optimization to minimize the error between the model and the observed sensor data iteratively for each joint. They also incorporate a Kalman filter into their pose estimation routine. While their method uses the same ICP cost function as ours, we use simple stochastic gradient descent (as in [4]) to find a minimum of that cost function, which is more efficient. Kranin *et. al.* [7] are able to spend more time accurately estimating the robot's pose than we are, since their task is to model a static object held in the robot's hand, wheras our task is to dynamically estimate the pose of the robot's arm in closed-loop position servoing in order to accomplish a manipulation task in real time.

In an independent work, Hudson *et al.* [5] combine different sensory inputs (depth data from stereo cameras and 3D ranging sensors as well as visual appearance features and silhouettes of the object and manipulator) using an unscented Kalman Filter framework to leverage the advantage of each sensor and various features. Our work, in contrast, relies only on a depth sensor and joint encoders, makes no Kalman filter approximation, and has no requirement for fiducial or visible silhouettes. This allows us to use our algorithm in real-time during a control loop, rather than as an offline procedure used before grasping, in contrast to [5].

Our work is closely related to Position-Based Visual Servoing (PBVS) [6], in which the end effector pose and/or poses of objects are estimated in the camera frame, and control commands are sent to the robot arm. Our work is unique in that we do not do 3D reconstruction, and rely on depth data directly with no further segmentation instead.

III. The Algorithm

We treat the estimation of robot's configuration as an optimization problem, which we solve by stochastic gradient descent [3]. In the optimization procedure, we attempt to minimize the sum-squared correspondance error between the closest points on the robot's body, and the observed sensor points.

A. Optimization via Stochastic Gradient Descent

Assume that the robot's body is made of one or more articulated chains with N degrees of freedom. Call $q \in \mathbb{R}^N$ a configuration of the robot. We can model the robot as a set of points in 3D space, given its configuration. Define the forward kinematics function $\mathcal{B}(q) = \{b_1, b_2, b_3, \ldots, b_K\}$, where $b_i \in \mathbb{R}^3$ is the i^{th} point of the robot's body.

At the same time, the robot has a sensor which produces a point cloud (such as a depth camera, a stereo camera, or a laser). Call the point cloud $\mathbf{Z} = \{z_1, z_2, z_3, \ldots, z_P\}$, where $z_i \in \mathbb{R}^3$ is the i^{th} point in the point cloud.

We define the minimum distance operator which gives the squared distance to the closest point in the sensor point cloud to the body point $\mathcal{B}_i(q) = b_i$.

$$\begin{aligned} D^*(q, i, \mathbf{Z}) &= \min_{z \in \mathbf{Z}} \|\mathcal{B}_i(q) - z\|_2^2 \\ &= \min_{z \in \mathbf{Z}} (\epsilon_i^{z\,T} \epsilon_i^z) \end{aligned} \quad (1)$$

where $\epsilon_i^{z^T} = \mathcal{B}_i(q) - z$ If we assume that the set $\mathcal{B}(q)$ contains only those points of the robot's body visible to the sensor, we can define the loss function:

$$\mathcal{L}(q) = \frac{1}{2}\sum_{i=1}^{K} D^*(q, i, \mathbf{Z}) \tag{2}$$

Which we call the *correspondence error* between the robot's model and the observed sensor data. Our aim is to minimize this function. One way of minimizing it is to compute its gradient with respect to the robot's configuration, and descend this gradient.

Now, we can derive the gradient of the loss function

$$\begin{aligned} \nabla\mathcal{L}(q) = \frac{\partial}{\partial q}\mathcal{L}(q) &= \frac{\partial}{\partial q}\frac{1}{2}\sum_{i=1}^{K} D^*(q, i, \mathbf{Z}) \\ &= \frac{1}{2}\sum_{i=1}^{K}\frac{\partial}{\partial q} D^*(q, i, \mathbf{Z}) \end{aligned} \tag{3}$$

Since it is true in general that for differentiable functions $\frac{\partial}{\partial q}\min_x f(q,x) = \frac{\partial}{\partial q} f(q, x^*)$, where x^* is the value minimizing the function f, (*i.e* a subgradient of the min function evaluated at the minimum equals the gradient of the min function) [1], we have

$$\begin{aligned} \frac{\partial}{\partial q} D^*(q, i, \mathbf{Z}) &= \frac{\partial}{\partial q}\min_{z\in\mathbf{Z}} \|\mathcal{B}_i(q) - z\|_2^2 \\ &= \frac{\partial}{\partial q}\|\mathcal{B}_i(q) - z^*\|_2^2 \\ &= \frac{\partial}{\partial q}\epsilon_i^{z^*T}\epsilon_i^{z^*} \\ &= 2\frac{\partial \epsilon}{\partial q}^T \epsilon_i^{z^*} \\ &= 2\big(\frac{\partial}{\partial q}\big(\mathcal{B}_i(q) - z^*\big)\big)^T \epsilon_i^{z^*} \\ &= 2\big(\frac{\partial}{\partial q}\mathcal{B}_i(q)\big)^T \epsilon_i^{z^*} \\ &= 2\mathbf{J}_i\big|_q^T \epsilon_i^{z^*} \end{aligned} \tag{4}$$

where $\mathbf{J}_i\big|_q$ is the robot's kinematic Jacobian evaluated for the point $\mathcal{B}_i(q)$ at configuration q. It follows that

$$\nabla\mathcal{L}(q) = \sum_{i=1}^{K} \mathbf{J}_i\big|_q^T \epsilon_i^{z^*} \tag{5}$$

Geometrically, this can be interpreted for a serial link robot as a sum of forces, where each force is applied at a body point of the robot, pushing it away from the nearest point in the point cloud.

Notice that while the loss function is defined over joint configurations, the quantity we want to minimize is over points in the workspace. A step along the gradient of $\mathcal{L}$ is then the steepest step in configuration space which minimizes a loss function over body points. It would be more desirable to instead take the steepest step in workspace directly to minimize the loss function. We define a metric over changes in configurations which considers the resuting change in workspace points:

$$\|\Delta q\|_w = \Delta w_q^T \Delta w_q$$

where Δw_q is the resulting change in workspace of a single point from the change in configuration space Δq. This is:

$$\begin{aligned} \|\Delta q\|_w &= (\mathbf{J}\Delta q)^T(\mathbf{J}\Delta q) \\ &= \Delta q^T(\mathbf{J}^T\mathbf{J})\Delta q \end{aligned} \tag{6}$$

where $\mathbf{J}$ is the kinematic Jacobian of the manipulator evaluated for a particular workspace point.

We precondition [10] the gradient by considering a change in configuration Δq to be "small" when

$$\|\Delta q\|_w = \Delta q^T(\mathbf{J}^T\mathbf{J})\Delta q \leq \beta \tag{7}$$

where β is a small value.

By doing this, we transform the space of the loss function into one where joint values are scaled to correspond to their resulting workspace motion [1]. This leads to a new preconditioned gradient computation:

$$\nabla\mathcal{L}(q) = \sum_{i=1}^{K} \mathbf{J}_i\big|_q^+ \epsilon_i^{z^*} \tag{8}$$

where $\mathbf{J}_i\big|_q^+ = (\mathbf{J}_i^T\mathbf{J}_i)^{-1}\mathbf{J}_i^T$ is the regularizd Jacobian pseudo-inverse for the body point b_i evaulated at the configuration q. In practice, using the pseudo-inverse rather than the transpose led to faster and more stable convergence.

Then, we have an optimization procedure:

$$q^{t+1} = q^t - \lambda\nabla\mathcal{L}(q^t) \tag{9}$$

where $\nabla\mathcal{L}(q) = \frac{\partial}{\partial q}\mathcal{L}(q)$, and $\lambda \in \mathbb{R}$ is a step size, and q^t is the robot's best estimate of its position at time t.

If the robot didn't have access to joint encoders, the estimate q^t could be used directly as a belief about the robot's position. However, the joint encoders provide a very accurate estimate of the robot's state in themselves. We can incorporate the robot's joint encoders into the estimation by modelling the error of the system as a random (configuration-dependent) offset from the reported joint angles; *i.e*

$$q^* = q + \delta^t \tag{10}$$

where q^* is the true (unknown) state of the robot, and δ^t is a random (unknown) offset from the joint encoders q, at time t. Then, we keep a running estimate of δ (initially zero):

$$\delta^{t+1} = \delta^t - \lambda\nabla\mathcal{L}(q + \delta^t) \tag{11}$$

[1]For non-differentiable functions, this holds for all but a finite set of non-differentiable points. Moreover, in regions where the loss function is locally convex, the computation gives the local subgradient even at non-differentiable points.

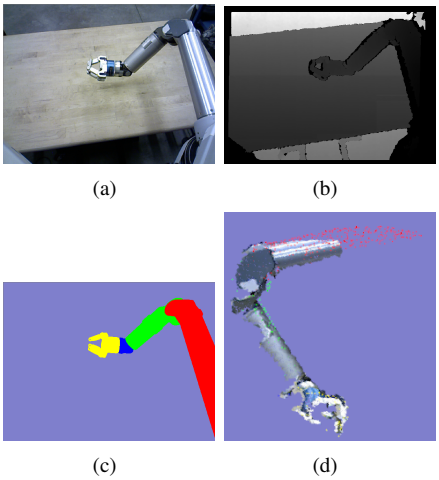


Fig. 3. Rendering phases for pose estimation. (a) the RGB image from the commercial depth camera. (b) the depth image from the sensor. (c) the 3D model of the robot, as rendered from the perspective of the depth camera. Each link is colored differently to provide a simple mapping from rendered point to link on the robot. (d) the final composite point cloud rendered in 3D, with the arm segmented out, and a random subset of model points selected.

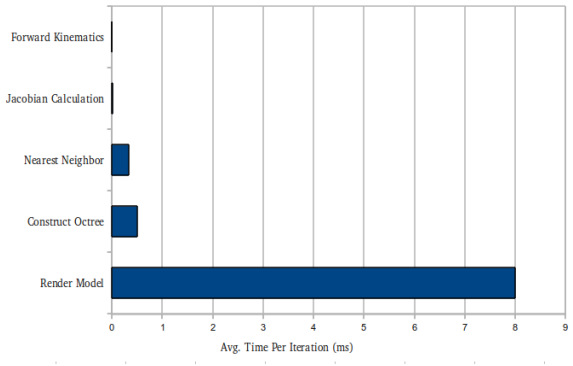


Fig. 4. Timing data for one iteration of the algorithm with 500 randomly subsampled points. Different phases of the algorithm are timed. Other phases not shown contributed neglible time. Acquiring sensor data and constructing the octree happen only once per sensor update (which in our case is 30 Hz).

Note that there might be many points in both $\mathbf{Z}$ and $\mathcal{B}$ at any given time, making the evaluation of $\nabla\mathcal{L}$ computationally infeasible. So instead, we choose a random subset of $m = \min(K, M)$ points from $\mathcal{B}$, and evaluate the gradient for those points only. In this sense, our gradient descent procedure becomes a stochastic one. Additionally, we normalize the gradient by multiplying it by $\frac{1}{m}$, since the gradient is naturally scaled by the number of observed points.

The result is that the offset δ converges to a local minimum of the loss function, so that the robot's model best describes the sensor data.

B. Performance Considerations

The algorithm has three major components which comprise a majority of the computation time.

1) Generating $\mathcal{B}$
   We model the robot as a triangular mesh, and exploit graphics hardware to extract a series of points which are visible to the sensor by rendering a simulated depth image.
2) Finding corresponding points between $\mathbf{Z}_b$ and $\mathcal{B}$
   We do this efficiently by storing the sensor data in an octree. In the process of searching for the nearest point, we reject points beyond a small threshold. By doing this, the algorithm will sometimes fail to find correspondences between certain points. If this is the case, we simply ignore points for which no correspondence could be found.
3) Calculating for each point $\mathbf{J}_i\big|_{q+\delta}^{+}$
   First, we color each rendered point in $\mathcal{B}$ so that the color of each point corresponds to the link it is attached to. From there, we use the method described in [12] to efficiently find the Kinematic Jacobian for each randomly subsampled point. We then compute the pseudo-inverse.

C. Closed Loop Position Servoing

Now that we have a sufficiently fast pose estimation algorithm, we can use it inside a control loop to move points on the robot's body to a desired position in space. Suppose that we have a frame of interest somewhere on the robot's body (e.g, the end effector frame) called x. Then, if we want to move this frame toward some desired location x_d, we can perform a linear Jacobian pseudo-inverse servo motion inside a control loop to acheive this, following the general equation [12]:

$$\dot{q} = \mathbf{J}_e^{+}\dot{x}_d + \lambda(\mathbf{I} - \mathbf{J}_e^{+}\mathbf{J}_e)\nabla\mathbf{H}(q), \tag{12}$$

where $\mathbf{J}_e$ is the robot Jacobian with its pseudo-inverse denoted $\mathbf{J}_e^{+}$, λ is a gain value, and $\mathbf{H}(q)$ is a null-space cost/utility function. Different criteria can be used to define $\mathbf{H}(q)$ depending on the objective, e.g., avoiding joint limits or kinematic singularities, and $\dot{x}_d$ is a velocity toward the desired pose.

However, since we are not confident about where x is exactly, we instead have some current estimate for x, called $\hat{x}$, which comes from the pose estimation procedure. $\hat{x}$ arrives asynchronously as the sensor provides new depth data. This is stored in a buffer, which the inner control loop reads.

If the speed at which we move is slow enough, the control loop remains stable, and the robot acheives the desired location for $\hat{x}$. The details of this control loop are laid out in (Fig. 5).

IV. Experiments

A. Servoing to a Point

We implement our algorithm in C++ on the DARPA ARM-S Robot [2], which has two Barret WAM arms, and an Asus Xtion Pro commercial depth sensor. Our primary computer is a Dell Precision T7500 with 28 GB of RAM, an 8-core Intel i7 processor, and an Nvidia Quadro FX 4800 GPU.

Our first test is to repeatedly touch a 1.5 cm radius red dot with one of the robot's fingers. To do this, we first perceive the red dot using the depth sensor and color segmentation;

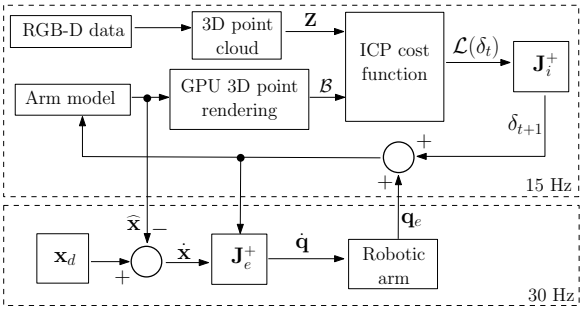
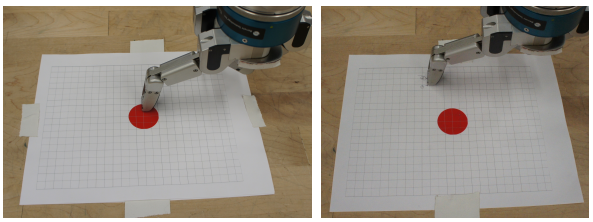


Fig. 5. Block diagram of the pose estimation being used inside a control loop. There are two blocks in the control loop. The top block corresponds to the pose estimation loop, which occurs whenever a new sensor update $\mathbf{Z}$ is received from the RGB-D sensor. The bottom block corresponds to the servoing loop, which moves the arm iteratively until a desired pose x_d is reached for a particular point attached to the robot's body. The servoing loop reads the configuration offset from a buffer that the sensor loop writes into.



(a) (b)

Fig. 6. The robot touches a red circle approximately 1cm in radius by servoing using a Jacobian pseudo-inverse move such that the finger lands on the center of the red circle. Typical cases with (a) and without (b) pose estimation are shown. Points on the paper where the finger touched were marked by hand. Each touch used a randomly selected inverse kinematics solution which puts the finger 10 cm above the red dot as the starting configuration.

then, we plan to a pose 10 cm above the red dot by selecting a random inverse kinematics solution which places the finger just above the red dot. Finally, we execute a Jacobian pseudo-inverse motion in a control loop which iteratively moves the finger of the robot toward the red dot until it touches. By including pose estimation in the control loop, we are able to iteratively correct error as the robot moves toward its target. Each touch is recorded by hand by marking the paper where the robot touches. We do this both with and without arm pose estimation (Fig. 6).

Our results (Fig. 7) show that with arm pose estimation, we are able to reduce final pose error so that all touches fall within the red dot. Without arm pose estimation, there is significant systematic error in the final pose of the arm, and the standard deviation of the touches is much larger.

B. Robustness to Exstrinsic Sensor Calibration Error

Because it is simply correcting the current error between the kinematic model of the robot and sensor data, our algorithm is robust to small calibration errors with respect to the sensor. To confirm this, we simulated a small error in the extrinsic calibration of the sensor by corrupting the joint encoders of the robot's pan-tilt neck mechanism by an artificial offset (from -9 to 9 degrees). We again performed

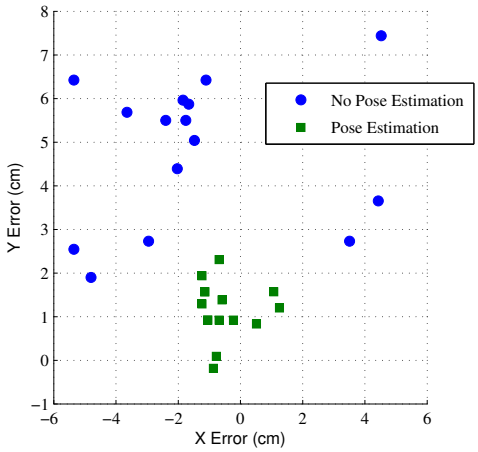


Fig. 7. Fifteen touches of the red circle for each method are displayed, the error is shown in centimeters. The red circle is approximately 1.5 cm in radius, and the robot is attempting to touch the center of the circle (which is estimated by a vision system using color segmentation). The standard deviation of the error without pose estimation is $[3.2, 1.7]$ cm in x and y respectively, and with pose estimation it is $[0.8, 0.6]$ cm.

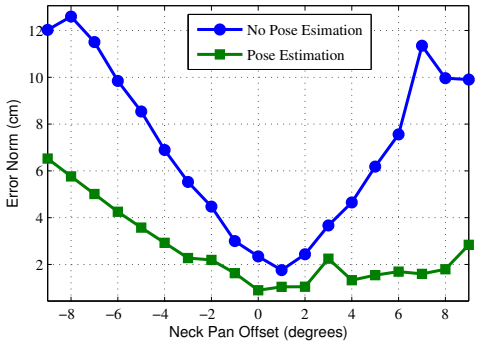


Fig. 8. We performed the dot-touching test again, but with different amounts of artificial error added to the pan joint of the robot's pan-tilt head (on which the RGB-D sensor sits). The norm of the error with and without pose estimation is shown.

the dot-touching test, but with different amounts of neck error.

The data (Fig. 8) show that, without arm pose estimation, the error of the final touch location diverges dramatically as calibration error increases. In contrast, with pose estimation, the error increases much more slowly. It just so happened that for negative encoder offsets to the pan joint of the neck, the elbow of the robot's arm occluded its hand – making pose estimation difficult. While for positive offsets, the hand was made *more* visible, and the pose estimation converged relatively faster. The lowest error for both cases is when the artificial neck error is near zero.

This result highlights the need for visibility planning in addition to pose estimation, as the estimator will converge more slowly when less of the robot's arm is visible.

C. Opening a Door

Our next experiment shows the real-time arm pose estimation algorithm's usage in everyday scenarios. For Phase I

of the DARPA ARM-S Project [2], one of the tasks was to autonomously open a small door. Due to error in the perceived location of the handle, and error in the robot's joint encoders, simply perceiving and grasping the door handle is a challenge. To solve this problem, we initially used guarded moves and force sensing [2] to blindly move the hand into position to grasp the door handle. However, this approach was time consuming and prone to failure.

Rather than using guarded moves, we now instead continuously estimate the pose of the robot's arm with the depth camera as the robot servos toward a desired pose just above the door handle. We then grasp the handle, and open the door (Fig. 1). We compared this behavior of the robot when it is able to use pose estimation, and when it simply blindly servos to a pose just above the doorhandle.

We found that without using pose estimation, the robot fails 10/10 trials to grasp the door handle. In two of the trials, the robot missed the door handle completely. However, using pose estimation, the robot succeeds in 9/10 trials, with the one failure case occuring due to visibility limitations of the chosen arm configuration. A video of this experiment can be found here [2].

V. Discussion

By optimizing the joint offset δ inside a closed control loop, we are able to iteratively correct for configuration-specific errors during the execution of guarded moves by the robot. However, it is not clear that our algorithm is actually estimating the state of the robot. Rather, since our loss function is defined over the correspondence error between the robot model and the observed sensor data, we are only indirectly finding joint offsets which happen to "explain away" perceived error. Even so, we are able to perform real-world manipulation tasks using our method.

We can frame the goal of manipulation tasks in terms of the geometric configuration of bodies in workspace. The goal of a grasping task, for example, is to move the robot's arm to a configuration where its end effector's position in the workspace corresponds to the physical position of the object we wish the robot to grasp. This is complicated by the fact that the robot only has access to data it can perceive through its sensors. Different sensors have different systemic errors – so if the perception system of the robot is used to estimate the physical workspace position of the object to be grasped, while the joint encoders are used to estimate the physical workspace position of the robot's body, the robot is bound to fail the grasping task because the errors between sensors are *not equivalent*.

Our method mitigates this issue by transforming the data from the joint encoders implicitly into the data from the depth sensor. It simply adjusts the estimated configuration of the robot so that the points on the body of the robot correspond to the (possibly incorrect or noisy) points observed by the sensor. The result is that the *actual* points of the body and the object to be grasped line up; and the robot successfully grasps the object.

[2]Youtube video of the door opening experiment http://www.youtube.com/watch?v=CbflqzBrd44

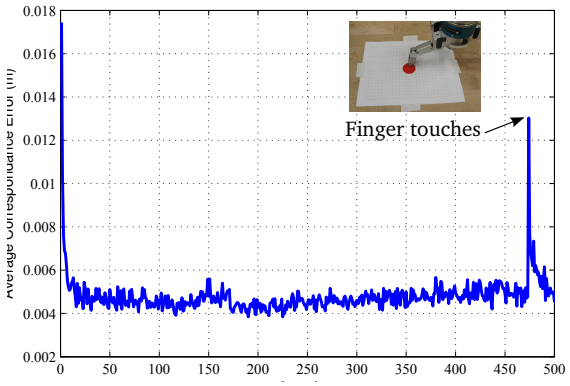


Fig. 9. The average error at each iteration between closest points in the kinematic model and the observed depth data is recorded during the dot touching experiment. The first spike represents the initial error, which quickly converges. The second spike (around iteration 480), corresponds to the point at which the robot touched the surface of the table (this causes strain on the joints, and causes them to move).

Future work might incorporate disturbance detection (Fig. 9) i.e, using pose estimation to determine when the arm has collided with obstacles or has successfully grasped an object. Another concern is our method's robustness to outliers and other disturbances. It may be worthwhile to include a statistical filter, such as a Kalman Filter, to explicitly model the uncertainty of the arm's pose. We may also wish to speed up the procedure even further by incorporating massively parallel computing. Many of the steps in our procedure are trivially parallel.

ACKNOWLEDGMENTS

The authors gratefully acknowledge funding under the DARPA Autonomous Robotic Manipulation Software Track (ARM-S) program.

References

[1] S. Amari and S.C. Douglas. Why natural gradient? In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 2, pages 1213–1216 vol.2, 1998.

[2] J. Andrew (Drew) Bagnell, Felipe Cavalcanti, Lei Cui, Thomas Galluzzo, Martial Hebert, Moslem Kazemi, Matthew Klingensmith, Jacqueline Libby, Tian Yu Liu, Nancy Pollard, Mikhail Pivtoraiko, Jean-Sebastien Valois, and Ranqi Zhu. An integrated system for autonomous robotics manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2955–2962, October 2012.

[3] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In Yves Lechevallier and Gilbert Saporta, editors, *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT'2010)*, pages 177–187, Paris, France, August 2010. Springer.

[4] Daniel Grest, Jan Woetzel, and Reinhard Koch. Nonlinear Body Pose Estimation from Depth Images. *Pattern Recognition*, pages 285–292, 2005.

[5] Paul Hebert, Nicolas Hudson, Jeremy Ma, Thomas Howard, Thomas Fuchs, Max Bajracharya, and Joel W. Burdick. Combined shape, appearance and silhouette for simultaneous manipulator and object tracking. In *ICRA*, pages 2405–2412, 2012.

[6] S. Hutchinson, G.D. Hager, and P.I. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5):651–670, 1996.

[7] Michael Krainin, Peter Henry, Xiaofeng Ren, and Dieter Fox. Manipulator and object tracking for in-hand 3d object modeling. *Int. J. Rob. Res.*, 30(11):1311–1327, September 2011.

[8] Thomas B. Moeslund and Erik Granum. A survey of computer vision-based human motion capture. *Comput. Vis. Image Underst.*, 81(3):231–268, 2001.

[9] Thomas B. Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *Comput. Vis. Image Underst.*, 104(2):90–126, 2006.

[10] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 2000.

[11] Stefano Pellegrini, Konrad Schindler, and Daniele Nardi. A generalisation of the icp algorithm for articulated bodies. In *BMVC*, 2008.

[12] L. Sciavicco and B. Siciliano. *Modelling and Control of Robot Manipulators*. Advanced Textbooks in Control and Signal Processing Series. Springer-Verlag, 2000.