Real-time Trajectory Synthesis for Information Maximization using Sequential Action Control and Least-Squares Estimation

Andrew D. Wilson, Jarvis A. Schultz, Alex R. Ansari, and Todd D. Murphey

Abstract— This paper presents the details and experimental results from an implementation of real-time trajectory generation and parameter estimation of a dynamic model using the Baxter Research Robot from Rethink Robotics. Trajectory generation is based on the maximization of Fisher information in real-time and closed-loop using a form of Sequential Action Control. On-line estimation is performed with a least-squares estimator employing a nonlinear state observer model computed with trep, a dynamics simulation package. Baxter is tasked with estimating the length of a string connected to a load suspended from the gripper with a load cell providing the single source of feedback to the estimator. Several trials are presented with varying initial estimates showing convergence to the actual length within a 6 second time-frame.

I. INTRODUCTION

A fundamental goal of artificial intelligence and learning in robotics is to provide the means for a robot to automatically synthesize actions which improve estimates of the robot's internal dynamics and dynamical interactions with real-world objects. From an early age, humans are able to constantly learn and improve upon their motor skills and interactions with objects in the physical world. We aim to provide this form of learning on robots using realtime processing of feedback from active exploration of the environment.

Since the general problem of model synthesis and learning remains a formidable one, we restrict ourselves in this paper to creating a method for real-time active synthesis of dynamic trajectories to estimate a single model parameter in a known dynamical model.

Active estimation of parameters within dynamical systems, also referred to as *optimal experimental design*, uses Fisher information as the primary metric [1]. Fisher information provides a best-case estimate of the estimator's performance given a set of measurements from a robot through the Cramer-Rao bound [2], [3]. A number of works on "exciting" trajectories by Armstrong and others [4]–[6] provide the theoretical basis for information-based estimation. In work by Emery [1], least-squares and maximum-likelihood estimation techniques are combined with Fisher information to optimize the experimental trajectories. In this case and several others, dynamics are solved as a discretized, constrained optimization problem [7]–[9]. One downside is that this time discretization can lead to high dimensional optimization problems (dimensions of 10^7 to 10^{12} are common



Fig. 1: Baxter performing real-time active parameter estimation on the length of a suspended payload.

in practice).

Previous works by the authors have demonstrated a trajectory optimization algorithm for information maximization on continuous and discrete systems [10], [11]. The algorithms successfully improve the expected information by perturbing the trajectories using first-order optimization techniques. While effective, these algorithms remains too computationally expensive for real-time application.

Sequential Action Control (SAC), a model-based control approach recently developed by Ansari and Murphey [12], has shown promise in simulation as a closed-loop recedinghorizon style controller that can compute optimal actions in real-time for nonlinear systems, similar to a nonlinear model predictive controller [13], [14]. Results have shown that the algorithm can outperform off-line trajectory optimization techniques on a number of canonical examples. This work encompasses one of the first practical implementations of the algorithm on a robotic platform, including coupling of the algorithm with a nonlinear state observer as described in Section III-B.3.

The main contribution of this paper is the experimental trial of the SAC algorithm coupled with an on-line estimator for real-time active estimation of a dynamical system, shown in Fig. 1. Detailed derivations of the underlying control principles and information theory can be found in [10], [12], [15]. The Baxter Research Robot has been used as a practical

The authors are with the Department of Mechanical Engineering, Northwestern University, 2145 Sheridan Road, Evanston, IL 60208, USA. Email: awilson@u.northwestern.edu, jschultz@northwestern.edu, alexanderansari2011@u.northwestern.edu, t-murphey@northwestern.edu

platform for a number of studies [16]–[18] while also presenting a number of challenges including high compliance and actuator saturation. Despite these potential sources of unmodeled dynamic effects, results show that the estimator successfully converges to the actual parameter value across several trials.

This paper is organized as follows: Section II provides an overview of the trajectory synthesis and estimation algorithms. The specific implementation details for Baxter and the suspended load estimation task are provided in Section III. Section IV provides results from several trials of the realtime estimation task, and Section V concludes the paper with notes on future work. One trial of the algorithm presented in this paper is also shown in the accompanying video.

II. REAL-TIME CONTROL STRUCTURE

This section presents a detailed overview of the active trajectory synthesis and estimation problem which is implemented in Section III. As shown in Fig. 2, there are two primary modules interacting with the robot hardware: the SAC controller for trajectory synthesis and the nonlinear least-squares estimator. At the highest level, the least-squares estimator requires the control inputs provided by the SAC controller to compute a predicted output which is compared to the actual measurements provided by the robot. The SAC controller is updated with new parameter estimates and state estimates by the least-squares estimator and optionally can receive state feedback directly from the robot hardware if available. These modules can be run asynchronously and at different rates with the use of a nonlinear state observer model.

In this paper, we assume that one parameter is uncertain with additive noise on observer measurements but negligible process noise. The same cost function with several unknown parameters is presented in [10]. For SAC, the state is usually derived assuming control-affine dynamics [12]. Thus, the model of the system is defined as

$$\dot{x} = f(x, u, \theta) = g(x, \theta) + h(x, \theta)u$$
(1)
$$\tilde{y} = y(x, u, \theta) + w_y$$

where $x \in \mathbb{R}^n$ defines the system states, $\tilde{y} \in \mathbb{R}^h$ defines the measured outputs, $u \in \mathbb{R}^m$ defines the inputs to the system, $\theta \in \mathbb{R}$ defines the parameter to be estimated, and w_y is additive output noise where $p(w_y) = N(0, \Sigma)$.

A. Nonlinear Least-Squares Estimator

While the robot is executing a motion, a nonlinear leastsquares estimator is used on-line to update the estimated value of the parameter as well as the robot state.

The least-squares estimator can be written as

$$\hat{\theta} = \arg\min_{\theta} \beta(\theta) \tag{2}$$

where

$$\beta(\theta) = \frac{1}{2} \sum_{i}^{h} (\tilde{y}(t_i) - y(t_i))^T \cdot \Sigma^{-1} \cdot (\tilde{y}(t_i) - y(t_i)).$$
 (3)



Fig. 2: Overview of the real-time control structure.

 $\tilde{y}(t_i)$ is the observed state at the i^{th} index of h measurements, $\Sigma \in \mathbb{R}^{h \times h}$ is the covariance matrix associated with the sensor measurement error, and $\hat{\theta}$ is the least-squares estimate of the parameter. The estimator recomputes a new estimate at a set frequency, incorporating any new measurements received since the last iteration.

Given this estimator, we will use gradient descent with a backtracking line-search to find optimal parameter values by minimizing the least-squares error in (3). To perform this optimization, the first derivative of $\beta(\theta)$ w.r.t. θ must be calculated.

The first derivative can be computed using the following equation:¹

$$D_{\theta}\beta(\theta) = \sum_{i}^{h} (\tilde{y}(t_i) - y(t_i))^T \cdot \Sigma^{-1} \cdot \Gamma_{\theta}(t_i)$$

where

$$\Gamma_{\theta}(t_i) = D_x y(x(t_i), u(t_i), \theta) \cdot D_{\theta} x(x(t_i), u(t_i), \theta) + D_{\theta} y(x(t_i), u(t_i), \theta).$$

This equation requires the evaluation of $D_{\theta}x(x(t_i), u(t_i), \theta)$ of (1), which is computed by the following ordinary differential equation (ODE):

$$\psi(t) = D_x f(x(t), u(t), \theta) \cdot \psi(t) + D_\theta f(x(t), u(t), \theta),$$

where

$$\psi(t) = D_{\theta} x(x(t), u(t), \theta) \in \mathbb{R}^n$$
 and $\psi(0) = \{0\}^n$.

Since the estimator requires a predicted output, y(t) which is based on current estimates of θ and x(t), a nonlinear state observer is also required for systems without full-state feedback. While any numerical differential equation solver may be sufficient, for this implementation, we are using the trep software package available at http://nxr.northwestern.edu/trep.

¹The notation, $D_{\kappa}\alpha(\kappa)$ represents the partial derivative of α w.r.t κ .

B. Sequential Action Control

The SAC control synthesis process follows a recedinghorizon style format to sequence together separately short optimal control *actions* into a piecewise continuous constrained feedback response to state. In SAC, *actions* are defined by a pair composed of a control vector value and its associated (typ. short) application duration which is computed for each action. The blue shaded region in Fig. 3 shows a SAC action for a 1-D control.

This paper expands the use of SAC to include a cost on the Fisher information of the uncertain parameter. For this implementation, we assume that the measurement noise of the system is normally distributed with zero process noise. Therefore the Fisher information is given by

$$I(\theta) = \sum_{k=k_0}^{k_f} \Gamma_{\theta}(t_k)^T \cdot \Sigma^{-1} \cdot \Gamma_{\theta}(t_k).$$
(4)

The algorithm iterates on the following 4-step process to synthesize each action:

1) Predict: The SAC algorithm predicts system motion from current state feedback, $x(t_0) = x_0$. The process involves simulation of a state and adjoint system (x, ρ) for a fixed time horizon, T, until (receding) final time $t_f = T + t_0$. For the purposes of this paper, the nominal control value used for simulations (x, ρ) is u = 0 so that SAC computes optimal actions relative to the free (unforced) system motion.

As detailed in [10], a cost function on Fisher information from (4) also requires the simulation of the gradient of xw.r.t. θ , i.e. $\psi(t) = D_{\theta}x$. These additional states are referred to in the paper as extended state dynamics and notated as $\bar{x} = (x, \psi)$.

For this implementation, we use only a running cost for the Sequential Action Controller which is written as

$$J_{cost} = \int_{t_0}^{t_f} l(\bar{x}(t)) dt \tag{5}$$

where

$$l(\bar{x}(t)) = \left[\Gamma_{\theta}(t)^T \cdot \Sigma^{-1} \cdot \Gamma_{\theta}(t)\right]^{-1} + x(t)^T \cdot Q_{\tau} \cdot x(t)$$

i.e., the minimization of the inverse of the information and an optional trajectory tracking cost to bias the system toward a particular part of the state space.

The adjoint variable $\rho : \mathbb{R} \mapsto \mathbb{R}^{2n}$ provides information about the sensitivity of the cost function to the extended state, \bar{x} . The algorithm maps this sensitivity to a control sensitivity provided by an inner product between the adjoint and dynamics (1). The process of control synthesis uses this sensitivity, $\frac{dJ_{cost}}{du_{\tau}}$, to search for least norm actions that optimize the expected change in cost (5). Thus SAC actions optimize the rate of trajectory improvement. These optimal actions depend directly on the adjoint, which is determined from open-loop simulation of the following equation,

$$\dot{\rho} = -D_{\bar{x}}l(\bar{x})^T - D_{\bar{x}}f(\bar{x},u)^T\rho$$

with a terminal condition $\rho(t_f) = 0$.



Fig. 3: SAC actions for a 1-D control are sequenced in receding-horizon fashion.

2) The Value of Optimal Action: After simulating the open-loop system (\bar{x}, ρ) under nominal control, SAC computes the value of actions that it should apply to optimally improve nominal trajectory cost (5). Because SAC has not yet decided on a time, τ , specifying when to act,² it searches for a curve u^* that provides the value of the optimal action it should apply as a function of time. The algorithm will search this curve to determine the best (optimal) time to act. The entire curve that provides the value of the optimal actions that maximize trajectory improvement is provided analytically from the expression,

$$u^* = (\Lambda + R^T)^{-1} h(\bar{x})^T \rho \alpha_d \tag{6}$$

with $\Lambda \triangleq h(\bar{x})^T \rho \rho^T h(\bar{x})$ [12].

As opposed to traditional receding-horizon and nonlinear trajectory optimization routines, SAC actions optimally *improve* rather than *minimize* a trajectory cost over the current horizon. The process is often less sensitive to local minima and examples in [12] show that actions based on (6) actually outperform nonlinear trajectory optimization in terms of cost and computational efficiency on a variety of benchmark nonlinear control problems. Furthermore, [12] proves min/max input constraints of the form $u_{min,k} < 0 < u_{max,k} \ \forall k \in \{1, \ldots, m\}$, can be applied on-line to actions from (6) without added computation.

3) When to Act: In application, SAC can pre-specify an action time $\tau = t_0$ so that actions are always applied from the current time and sequenced into a piecewise continuous, receding-horizon style response to state. However, SAC includes the option to search for an optimal time to act. It can therefore do nothing and choose to wait until a system

²Each cycle of control synthesis assumes SAC will specify / search for an optimal time to act between the current time, t_0 , and the end of the prediction horizon, t_f .

"drifts"³ into a more opportune state where control effort would be better spent. In searching for an optimal time to act, SAC optimizes an objective measuring the cost of waiting relative to the expected efficacy of controls (6) in improving system performance based on (5). It searches for the time, τ that minimizes the objective

$$J_t(\tau) = \|u^*(\tau)\| + \frac{dJ_{cost}}{du_{\tau}}\Big|_{\tau} + (\tau - t_0)^{\gamma}$$

with $\gamma = 1.6$ and $\frac{dJ_{cost}}{du_{\tau}}$ as the measure of expected cost improvement (see [12]).

4) How Long to Act: The SAC algorithm computes a duration, Δt , to apply control action values (6) using a line search with simple descent condition. The line search process iteratively reduces an initial duration, $\Delta t = \Delta t_0$, and simulates the expected cost (5) based on application of the control action around the application time τ . If the current duration results in a minimum improvement in cost (relative to the nominal control), it is selected and the action is fully specified based on the pair $(u^*(\tau), \Delta t)$. If the action does not provide the expected cost improvement, the line search reduces the duration and repeats. The process is guaranteed to find a duration that provides a minimum expected improvement in cost [19].

For a more detailed derivation of SAC control synthesis with examples see [12], [19].

III. BAXTER IMPLEMENTATION

This section describes both the problem formulation and software specific implementation of the control structure described in the previous section on the Baxter Research Robot created by Rethink Robotics [20]. Results from experimental trials are presented in Section IV.

A. Problem Description and Model Formulation

To test this control paradigm in real-time on a practical robotic system, we chose the task of determining the string length of a suspended payload from one of Baxter's grippers using a load cell as the sole output to the estimator. This configuration necessitates active motion to estimate the string length as the Fisher information is zero when the robot is stationary. To simplify the control of Baxter, we chose to approximate the end effector as kinematic and control motion only in the Cartesian x-axis, x_B . The equations of motion for the system are given by the following,

$$f(x, u, \theta) = \begin{bmatrix} \dot{x}_B \\ u \\ \dot{\phi} \\ \frac{u}{\ell} \cos \phi - \frac{g}{\ell} \sin \phi \end{bmatrix}$$

where u is the x-axis acceleration of the gripper, m is the known mass suspended from the robot, and ℓ is the length of the string, which will be estimated. Additionally, the equation for the force output F_s is

$$y(x, u, \theta) = F_s = mg\cos\phi - m\ell\phi^2 - u\sin\phi.$$

³Under free dynamics and the nominal control u = 0.

It is assumed that the trajectories will maintain tension in the string; therefore, the distance between the robot and mass is fixed. Given this system model, the extended state $\bar{x} \in \mathbb{R}^8$.

B. Experimental Implementation

As shown in Fig. 2, there are essentially two separate modules interacting with the robot hardware: the SAC trajectory synthesis module, and the nonlinear least-squares estimator module. In this section we describe how each of these modules is implemented for the Baxter experiments.

The backbone of communication between modules and the robot is provided by the Robot Operating System (ROS) [21]. ROS provides the ability to asynchronously run the control and estimation modules, implemented as ROS nodes, and Baxter natively uses ROS as the primary API for motion commands. Three primary nodes are employed: the SAC control node, a measurement receiver node, and the estimator node.

1) Baxter SAC Node: As mentioned in Section III-A, the end effector of the Baxter robot is approximated as a kinematic input to the dynamic suspended mass system. The dynamic model for the suspended mass system assumes only planar motion, and the kinematic input moves perpendicular to gravity in this plane. It follows that the SAC module produces target locations along a one-dimensional line that Baxter's end effector should follow. A joint velocity controller for Baxter's right arm is used to stabilize the right end effector to these target locations. To avoid issues with kinematic singularities in the manipulator while controlling to these target locations, this one-dimensional line is expanded to a candidate set of closely-spaced end effector targets in SE(3). An off-line computation is done to solve the inverse kinematics problem for each of the targets in the set producing a set of target joint angles for each target. These joint angle targets are then stored to disk as a lookup table between target horizontal positions, like those produced by SAC, and target joint angles. During an experimental run, the target locations from SAC are fed into a controller running at 100 Hz that controls reference joint velocities to stabilize to the target joint angles from the lookup table. Baxter's internal controller runs a real-time loop at high frequency to stabilize each of the joints to the reference velocities set by the 100 Hz controller.

2) Baxter Measurement Receiver Node: The payload is attached to Baxter's end effector through a one-dimensional load cell. A microcontroller samples the load cell at 100 Hz and transmits the measured forces via a serial link back to a client computer which is communicating with Baxter over an Ethernet connection. These force measurements represent the $\tilde{y}(t_i)$ in (3). The load cell has been calibrated prior to the experiment to convert the load cell output to a force (N). The resulting force is timestamped as it is received and published at 100 Hz for use by the estimator node.

3) Baxter Estimator Node: The estimator node subscribes to the actual end effector trajectory which is provided by the Baxter API. These measurements are used to generate the $y(t_i)$ terms in (3) through the use of a nonlinear state



Fig. 4: Experimental trials on Baxter using different initial estimates of the string length. The dashed line indicates the actual measured length of 0.415 m.

observer. As mentioned in Section II-A, the trep software package is used as the state observer. The trajectory is used as the input and trep provides the predicted state evolution of the suspended load. From these states, the predicted force, y(t) can be calculated.

At a frequency of 2 Hz, the estimation module solves the optimization problem described by (2), updating the estimate of the string length. This frequency was chosen as a conservative rate at which the estimator has enough time to provide an update; however, the rate can be modified depending on the required computational time for the particular system. This parameter estimate and new state estimate are provided as a service to the SAC node which queries the service at the start of each computation. Updates to the string length estimate and expected state are reflected in this service call.

IV. RESULTS

This section presents the results of real-time experimental trials of the algorithm using the Baxter robot. A total of 9 trials are presented with initial estimates of the length parameter, ℓ that varied from 0.3 m to 0.5 m in increments of 0.25 m. The tests were conducted for a total of 6 seconds for each trial. For each trial, the initial position of the gripper was set to $x_B = -0.25$ m and the suspended load was hanging in a stationary position.

The convergence of the parameter estimates over time for all 9 trials is shown in Fig. 4. The estimates clearly converge toward the actual string length which is noted by the dashed horizontal line. Qualitatively, the rate of convergence across all the trials is similar, with estimates beginning to converge around 2 to 4 seconds. This suggests that for this system, the trajectories generated by the SAC algorithm provide relatively similar levels of information despite the initial estimate. Since the algorithm is based on gradient descent techniques, estimates deviating too far from the actual value may cause the estimator not to converge. The mean estimate of ℓ from the 9 trials after 6 seconds was 0.411 m with the



Fig. 5: Measured and predicted force from one Baxter trial over time for $\ell_0 = 0.35$ m.



Fig. 6: Expected information accrued over time during one Baxter trial for $\ell_0 = 0.35$ m.



Fig. 7: Endpoint reference compared to actual endpoint position of one Baxter trial with $\ell_0 = 0.35$ m.

actual string length set to 0.415 m. The standard deviation of the final estimates is 0.0055 m.

The measured and predicted force data is shown from a single trial in Fig. 5. The figure show that as the suspended load begins to react to the control inputs, providing a much better signal for the estimator to track, the estimates begin to converge. It is also noted in this figure that the measurements taken by the load cell began to saturate our hardware setup around 2.5 N, however, the estimator is still able track the phase of the system even with this saturation.

Additionally, Fig. 6 shows the expected information accrued over time from one experimental trial. A large increase in information begins to accrue as the suspended load is excited, leading to the convergence of the parameter estimate. The Fisher information provides a best-case bound on the parameter estimate through the Cramer-Rao bound provided the estimator is unbiased [2].

The generated and executed trajectories can be seen for one of the trials in Fig. 7. As discussed in Section III, the motion of the gripper is controlled to move along the Cartesian x-axis using PID control to follow the generated reference. The combination of the PID controller and the compliance of the series elastic actuators leads to moderate overshoot in the endpoint position; however, for these trials, it is acceptable. The estimator is provided the actual endpoint position which allows for accurate prediction of the load despite this error and SAC receives the estimated state feedback from the estimator for trajectory synthesis.

V. CONCLUSION AND FUTURE WORK

This paper presented an experimental implementation of a control structure for real-time active estimation of a parameter in a dynamic system. Results from several trials on the Baxter robot with different initial estimates of the string length quickly converge to the actual length using a trajectory synthesized on-line using the Sequential Action Controller. Despite a large amount of overshoot due to high compliance in the robot's actuators, the estimation algorithm is able to use the output from a nonlinear state observer model implemented with trep to minimize the effect of the tracking error.

This work represents only the first step toward improving robot learning on physical systems by exploiting dynamic models. One possible improvement may include the use of Lie groups as the fundamental tool to build the dynamic models. While the models can be more complicated to create, they can facilitate better-posed solutions to the optimization problems, reducing some of the singularities and angle wrapping issues that occur when modeling certain robotic systems, especially non-holonomic systems.

Additionally, the extension to multiple parameters only requires that an optimality metric is set as shown in [10]; however, it would be useful for a system to realize which parameters need to be estimated in the first place. This could be achieved through forms of sensor fusion and covariance estimation as well as an exploration-based search algorithms. Eventually, the addition of a model creation and learning algorithm would enable a robot to develop not only estimates of the parameters, but also internal structure without prior knowledge of the internal dynamic model. The continued development of dynamics-based methods for robot learning will allow robots to learn and better interact with real-world objects and tasks in physical environments.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant CMMI 1334609. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

- A. F. Emery and A. V. Nenarokomov, "Optimal experiment design," Measurement Science and Technology, vol. 9, pp. 864–876, June 1998.
- [2] C. R. Rao and J. Wishart, "Minimum variance and the estimation of several parameters," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 43, p. 280, Oct. 1947.
- [3] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation: Theory Algorithms and Software*. John Wiley & Sons, 2004.
- [4] B. Armstrong, "On finding exciting trajectories for identification experiments involving systems with nonlinear dynamics," *The Int. Journal of Robotics Research*, vol. 8, pp. 28–48, Dec. 1989.
- [5] J. Swevers, C. Ganseman, D. Tukel, J. de Schutter, and H. Van Brussel, "Optimal robot excitation and identification," *IEEE Trans. on Robotics and Automation*, vol. 13, no. 5, pp. 730–740, 1997.
- [6] M. Gautier and W. Khalil, "Exciting trajectories for the identification of base inertial parameters of robots," *The Int. Journal of Robotics Research*, vol. 11, pp. 362–375, Aug. 1992.
- [7] R. Mehra, "Optimal input signals for parameter estimation in dynamic systems–Survey and new results," *IEEE Trans. on Automatic Control*, vol. 19, pp. 753–768, Dec. 1974.
- [8] T. L. Vincent, C. Novara, K. Hsu, and K. Poolla, "Input design for structured nonlinear system identification," *Automatica*, vol. 46, pp. 990–998, June 2010.
- [9] G. Franceschini and S. Macchietto, "Model-based design of experiments for parameter precision: State of the art," *Chemical Engineering Science*, vol. 63, pp. 4846–4872, Oct. 2008.
- [10] A. D. Wilson, J. A. Schultz, and T. D. Murphey, "Trajectory synthesis for Fisher information maximization," *IEEE Trans. on Robotics*, vol. 30, pp. 1358–1370, Dec 2014.
- [11] A. D. Wilson and T. D. Murphey, "Maximizing Fisher information using discrete mechanics and projection-based trajectory optimization," in 2015 IEEE Int. Conf. on Robotics and Automation, May 2015.
- [12] A. R. Ansari and T. D. Murphey, "Sequential Action Control: Closedform optimal control for nonlinear systems," *In Review*, available at http://nxr.northwestern.edu/publications.
- [13] E. Camacho and C. Alba, *Model Predictive Control*. Advanced Textbooks in Control and Signal Processing, Springer, 2013.
- [14] L. Grüne and J. Pannek, *Nonlinear Model Predictive Control: Theory and Algorithms*. Communications and Control Engineering, Springer, 2011.
- [15] E. R. Johnson and T. D. Murphey, "Scalable variational integrators for constrained mechanical systems in generalized coordinates," *Robotics, IEEE Trans. on*, vol. 25, pp. 1249–1261, Dec 2009.
- [16] T. M. Caldwell, D. Coleman, and N. Correll, "Optimal parameter identification for discrete mechanical systems with application to flexible object manipulation," in 2014 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2014.
- [17] R. Mao, Y. Yang, C. Fermuller, Y. Aloimonos, and J. S. Baras, "Learning hand movements from markerless demonstrations for humanoid tasks," in 2014 IEEE-RAS Int. Conf. on Humanoid Robots, pp. 938– 943, Nov 2014.
- [18] C. Bowen and R. Alterovitz, "Closed-loop global motion planning for reactive execution of learned tasks," in 2014 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 1754–1760, Sept 2014.
- [19] A. R. Ansari and T. D. Murphey, "Control-on-request: Short-burst assistive control for long time horizon improvement," in 2015 American Control Conf., July 2015.
- [20] Rethink Robotics Baxter Research Robot. [Online]. Available: http://www.rethinkrobotics.com/baxter-research-robot/.
- [21] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, vol. 3, 2009.