AFRL-RI-RS-TR-2015-111

SYMBIOTIC OPTIMIZATION OF BEHAVIOR

UNIVERSITY OF WASHINGTON

MAY 2015

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2015-111 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/ S /

F. TANDI PAUGH
Work Unit Manager

/ S /

MICHAEL J. WESSING
Deputy Chief, Information Intelligence
Systems and Analysis Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
MAY 2015	FINAL TECHNICAL REPORT	SEP 2012 – DEC 2014

4. TITLE AND SUBTITLE

SYMBIOTIC OPTIMIZATION OF BEHAVIOR

5a. CONTRACT NUMBER
FA8750-12-1-0304

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
62702E

6. AUTHOR(S)

Emanuil Todorov

5d. PROJECT NUMBER
ROBO

5e. TASK NUMBER
PR

5f. WORK UNIT NUMBER
05

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Washington
4333 Brooklyn Ave NE
Seattle WA 98195-0001

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RIED
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSOR/MONITOR'S REPORT NUMBER
AFRL-RI-RS-TR-2015-111

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This project was part of the DARPA Virtual Robotics Challenge (VRC). The goal was to control a simulated Atlas robot in the Gazebo/ODE virtual environment and make it walk, drive a car, and connect a hose. Our approach was based on model-predictive control (MPC). We created a parallel simulation in our virtual environment (MuJoCo) which was able to simulate the robot dynamics much faster than real-time. This made it possible to optimize the controls online -- by exploring many candidate movement strategies and finding the one that is expected to work best starting from the currently estimated state of the robot.

15. SUBJECT TERMS
DARPA VRC, Virtual Robotics Challenge, Gazebo/ODE, model-predictive control, robots.

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**F. TANDI PAUGH**
U	U	U	UU	13	19b. TELEPHONE NUMBER (*Include area code*) **N/A**

Table of Contents

1.0 SUMMARY

This project was part of the DARPA Virtual Robotics Challenge (VRC). The problem statement, competition rules and scoring procedures were identical for all teams. Briefly, the goal was to control a simulated Atlas robot in the Gazebo/Open Dynamics Engine (ODE) virtual environment and make it walk, drive a car, and connect a hose. Our approach was based on model-predictive control (MPC). We created a parallel simulation in our virtual environment (MuJoCo) which was able to simulate the robot dynamics much faster than real-time. This made it possible to optimize the controls online -- by exploring many candidate movement strategies and finding the one that is expected to work best starting from the currently estimated state of the robot. The quality of the candidate strategies was evaluated by a cost function, which we designed in advance so as to capture the requirements of each task. The operator was able to assign different tasks to the robot by selecting (from a menu) the cost function to be optimized next. The operator was also responsible for visual perception -- by positioning and orienting spatial targets overlaid on images from the robot's camera.

In the course of the project we succeeded in refining our MPC machinery to the point where it could indeed run in real-time on such a complex robot, without any dynamics simplifications or model reductions. We also made progress in cost function design, and identified families of costs that capture common tasks and yet are amenable to efficient optimization. Furthermore we developed a framework for hierarchical MPC, where the high level can modify the cost being optimized on the low level. In the present implementation this high level can be either an operator using an interface designed specifically for this type of control, or a state machine that switches costs automatically when certain predefined criteria are met. The work we did in this project is described in a paper accepted in Humanoids 2013.

In the actual competition our system did not perform as well as we had hoped. This was partly because we run out of time, and partly because Gazebo/ODE inaccuracies caused more complications for us than for most other teams (we relied heavily on model-based control, and also generated very dynamic movements which resulted in large error spikes in the Gazebo simulated Inertial Measurement Units (IMUs) for unknown reasons). As one team member put it, this was a competition for building model airplanes and instead we built a 3D printer. But now that we have this universal control machinery, it is helping us make rapid progress in other robotics projects.

Overall we believe that the MPC approach we pursued in the VRC will revolutionize robotic control in the near future. The main challenge is to be able to run the necessary optimization in real-time, and we can already do it. Additional computing hardware and algorithmic improvements will of course help, but our simulation results are already better than any other approach to automatic control we are aware of. The next step is to address modeling and estimation errors which are inevitable with physical systems. This is something we are already working on. Our first results along these lines, on controlling a physical 12-degree-of-freedom hand in dexterous object manipulation, were just submitted to ICRA 2014.

2.0 INTRODUCTION

Disaster response is an area where robots can play an important role, by performing tasks in hazardous environments. Remote human operators are likely to be available to help the robot in every way possible. However disasters tend to interfere with communications, limiting the ability to tele-operate the robot by traditional means. This calls for an approach somewhere in between tele-operation and full autonomy: the operator receives sensor data intermittently, and sends commands that activate low-level behavioral routines -- which then manage all the degrees of freedom of the robot in real-time, until the next command from the operator arrives.

Apart from its practical importance, this setting is interesting scientifically because it isolates the challenging and relatively well-defined problem of sensorimotor control from high level perception and decision making (which are offloaded to the human operator). The problem difficulty arises from the complexity of the robot and environment, combined with the requirement to perform multiple different tasks -- which in principle should limit adhoc approaches and favor more principled solutions that generalize to multiple tasks.

We opted for a universal control strategy, aiming to solve all the VRC tasks using the same algorithms and software. The only thing that changes from one task to another is the cost function quantifying the notion of performance. The key idea is real-time numerical optimization: given the current estimate of the robot's state, we optimize a trajectory plan up to some time horizon, and execute it until the next plan becomes available. This makes it possible for the robot to discover online what control signals are appropriate at the current state, without having to pre-compute global solutions that work in every possible situation (which is generally difficult due to the curse of dimensionality). The principal challenge is performing the optimization fast enough to keep up with the real-time physics. We are able to do this thanks to recent advances in algorithms and software, as well as ever-faster computing hardware.

3.0 METHODS, ASSUMPTIONS AND PROCEDURES

3.1 System Overview

Figure 1 shows a block diagram of our system. The box labeled "Control System" corresponds to two high-end workstations connected to the simulated robot via high-speed Ethernet. One of the workstations is dedicated to model-predictive control (MPC). Its job is to update the trajectory plan as often as possible. How often the plan can be updated depends on the complexity of the robot, the processing power of the computer, and the planning time horizon. In the VRC context we were able to update the plan at 30Hz on average.

The second workstation (Support Tasks) is responsible for everything else: communication with the Operator Station, communication with the Gazebo simulation of the robot, communication with the MPC server, state estimation, and evaluation of the latest plan at a high rate (1000Hz). Note that the plan supplied by the MPC server includes both an open-loop state-control trajectory and a time-varying linear feedback control law around that trajectory, so it can be used for feedback control at high update rates. Of course this plan is only locally optimal, and if the state deviates too much the corresponding feedback controller may no longer be suitable. However the MPC server updates the plan fast enough, always starting at the latest estimated state, so in practice the state does not have enough time to "run away".

The Operator Station is the computer used by the human operator, on the other side of the communication bottleneck. This computer renders a 3D model of the Atlas simulation in our own virtual environment, using the latest state estimate. It also displays images from the robot's cameras allowing the operator to identify spatial targets visually, and provide the target position and orientation by aligning a geometric model over the camera image. The Operator Station also presents a menu of cost functions, enabling the operator to select which task or subtask should be performed when. The selected cost is sent to the MPC server, which immediately starts to optimize it -- resulting in very rapid behavioral transitions.

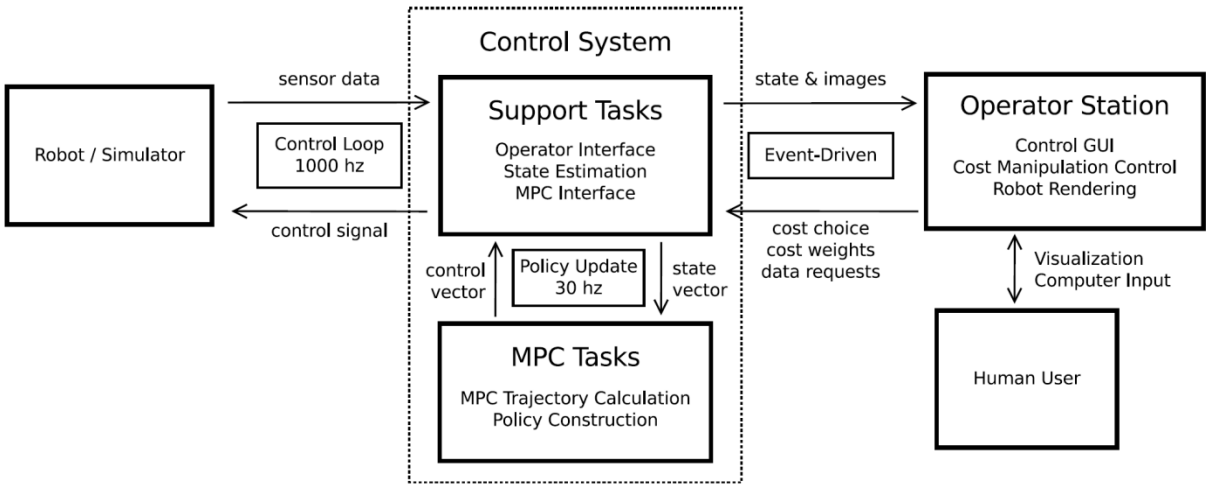

Figure 1. Block diagram of our system. From Erez et al, Humanoids 2013.

3.2 Modeling and Simulation

The Atlas simulation provided by Gazebo was neither fast enough nor stable enough for model-based optimization. This is why we created a parallel simulation in our environment (MuJoCo) and used it for MPC. MuJoCo is able to load Unified Robot Description Format (URDF) models, which made it possible to replicate the Atlas model as well as the geometry of the car, hose, valve etc. We also had the ability to edit model parameters in runtime to reflect the environment presented in each test run.

This dual-simulation approach put us in an awkward situation: we had a better physics engine than ODE, yet we had to control the ODE simulation which was defined as the ground truth. We also encountered a psychological problem: working with Gazebo was a significant step backward compared to working with MuJoCo, thus no one in the team was particularly eager to make the transition until the last minute. The fact that Gazebo was frozen very late in the VRC only reinforced this approach.

3.3 State Estimation

Our initial plan for state estimation was to combine joint angle data, IMU data, and 3D visual SLAM; the latter also providing a geometric model of the environment. We developed a 3D visual SLAM module based on an adaptation of the KinectFusion algorithm, however this was done late in the VRC because we had to wait for the Gazebo visual environment to become sufficiently rich. As a result we did not have time to integrate this module in the final system.

The state estimation approach we used in the competition only relied on joint angle data and simulated IMU data. Since the joint angles were exactly known, there were only 6 degrees of freedom (root translation and rotation) that had to be estimated. We did this using a predictor-corrector approach. In the predictor step we simply integrated the IMU data at 1000Hz. In the corrector step we assumed that the robot is always touching the ground. This was enforced by translating the estimated state in the vertical direction until the lowest point on the robot touched the ground. In addition we imposed a no-slip prior: if the estimated contact-space tangential velocity was non-zero, we modified the estimate of the root velocity so as to bring the contact-space velocity closer to zero (but not all the way, because the robot could actually slip).

The estimated state was computed by the "Support Tasks" workstation and sent to the MPC server to plan the next update, and to the operator console for visualization. It was also used locally to evaluate the feedback control law corresponding to the current plan, and send the resulting control signal to the Gazebo simulation.

3.4 Control

Our MPC approach was already described in high-level terms in Abstract and Introduction. Here we provide additional technical details. For trajectory optimization we used the iterative LQG (iLQG) algorithm we had developed earlier, with a number of refinements to speed it up for

online computation. We applied one iLQG iteration per plan update. Since we could warm-start from the previous solution, this was sufficient to track the local minimum. One iteration of iLQG involves the following computations:

1. Apply the open-loop controls from the last plan starting at the new estimated state, and integrate the dynamics forward to obtain an initial trajectory;
2. Construct a time-varying linearization of the dynamics around this trajectory, as well as a time-varying quadratic model of the cost function;
3. Perform a backward pass of local dynamic programming, approximating the optimal value function as a quadratic;
4. Perform a parallel line-search in trajectory space, by applying the improvement computed in the backward pass with different amplitudes, evaluating the cost for each resulting trajectory (on the full nonlinear model) and selecting the best amplitude.

Steps 1 and 3 are sequential but very fast. Steps 2 and 4 require more computation but fortunately they can be take advantage of parallel processing. Step 2 in particular is the most computationally demanding. This is because the dynamics are too complex for analytical differentiation, thus we have to resort to a finite-difference approximation. In other words, if we have D degrees of freedom (in the robot and any movable objects) and T time steps, the robot dynamics need to be evaluated on the order of D*T times per update. This entire computation takes 30 msec on average (see below). In our case we used T=16 timesteps, to match the number of processor cores available on the MPC server. The derivative computations for each time step were processed on a different core, resulting in 100% processor utilization. The simulation timestep for the MPC model was 20 msec, which is possible because MuJoCo is sufficiently accurate and thus remains stable at large timesteps.

As is always the case with MPC, some progress (i.e. reduction in cost) must be possible within the planning horizon in order for this control scheme to work. To this end we carefully designed cost functions that capture the tasks, but at the same time provide some cost gradient at all states. The main idea here is to avoid terminal costs and replace them with running costs. For example, suppose we want the hand to reach a spatial target 2 sec from now. If we only provided a terminal cost, MPC will not be able to accomplish the task because it does not see any evidence of progress within its planning horizon. So we use a different cost, which says that the hand should be at the target at all times. This is of course impossible, but what it does is encourage the optimizer to get to the target as fast as possible. At the same time we use a control energy cost, and the relative weighting of these two costs determines how fast the robot actually moves. In addition to these basic costs, we introduced more heuristic shaping costs (related to various stability criteria) to help overcome the limitations of the small planning horizon.

Apart from the technical challenge of applying MPC in real-time on a high-dimensional robot, we had to overcome a more fundamental issue -- which is that contact dynamics are inherently discontinuous and thus incompatible with gradient-based optimization. We have made a lot of progress over the past few years in terms of developing new formulations of the physics of contact that remain realistic, but at the same time provide just enough contact smoothing to enable gradient computations. Our MuJoCo engine has contact parameters that determine the

amount of smoothing. In practice we often plan through a smoother model and execute on a model with harder contacts, which (somewhat surprisingly) this yields good performance.

3.5 Operator Guidance

The human operator had two roles: use human vision to analyze the scene and determine spatial targets, and select costs that specify different behaviors for the robot.

Spatial target selection was done as follows. On request from the operator, the cloud simulation transmitted a JPEG image from the selected camera (head or hand). We did not stream video continuously to save bandwidth. The image was then displayed on the operator screen, and a 3D rendering of the Atlas robot (in MuJoCo) was superimposed on it using transparency and the same viewpoint as the robot's camera. This provided the operator with good situation awareness (having an operator who is an avid gamer seemed to help). The operator selected a 3D model from a menu, corresponding to the object in the visual scene. Then the operator used a 6D mouse to adjust the position and orientation of the model object so as to match the camera image. Once the adjustment was done, the operator issued a command which transmitted the object coordinates to the MPC server. These coordinates became parameters of the cost function being optimized (for example, the cost might compute the distance between the robot's hand and the target position specified by the operator).

Cost selection was done from a menu, where the different costs were labeled intuitively. The operator had prior experience with the MPC server optimizing these predefined costs, and knew what performance can be expected in different situations. Each cost was actually a composite cost, with multiple terms corresponding to energy, spatial targets, stability, or other shaping costs we introduced as needed. These cost terms had scalar weights that could also be adjusted. Both discrete cost selection and continuous weight adjustment proved useful.

Ideally the operator would be able to specify one task-level cost and let the MPC machinery fill in all the details and perform the task. In practice this is too much to expect with existing computing power (reflected in the relatively short planning horizon). Thus the operator has to provide additional guidance, specifying sub-tasks rather than complete tasks. These subtasks do not always have well-defined meaning; rather they may correspond to intuitive mid-level actions that have some physical consequences the operator has learned to expect (e.g. nudge the robot a bit to get in a configuration from which we can successfully execute a stand-up behavior).

4.0 RESULTS AND DISCUSSION

Our initial concern was that the discrepancies between the MuJoCo physics used for planning and the ODE physics used for simulation might result in poor MPC performance. To our pleasant surprise this was not the case. The performance of the ODE-Atlas was of course not as good as the MuJoCo-Atlas. However, if we could accomplish a task in MuJoCo, we could almost always accomplish it in ODE as well.

Estimation errors turned out to be a much bigger issue, for reasons beyond our control. The predictor-corrector approach described earlier relied on the IMU having low noise. The specified amplitude of the simulated noise was indeed very low, and if the simulated IMU had performed as advertised, our approach to estimation would have worked perfectly. Indeed it did work perfectly almost all the time. There was however a problem. Our MPC controller generated very dynamic movements (more dynamic than anything we saw from the other teams). For some reason this caused Gazebo to inject large noise spikes in the simulated IMU. A typical case is shown in Figure 2 below. The red trace is the vertical IMU acceleration reported by Gazebo (the version used during the competition) at 1000Hz. The blue trace is the true acceleration -- which we obtained by finite-differencing the ground truth position reported by Gazebo, also at 1000 Hz (we modified the SDF file to get 1000Hz output). Note that most of the time the two traces are perfectly superimposed (numerically there is a difference, but it is on the order of the expected noise amplitude and is not even visible). However there is the occasional large discrepancy.

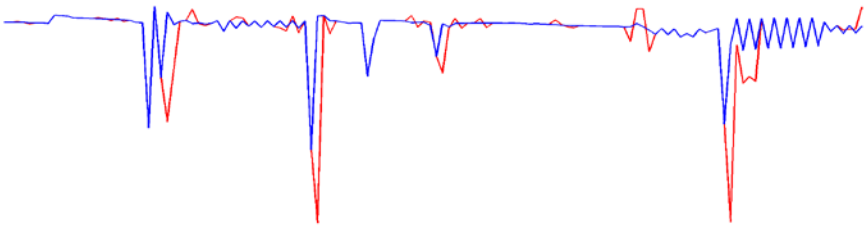

Figure 2. Illustration of IMU errors when Gazebo/ODE performed dynamic movements.

These IMU errors in turn caused the predictor step to occasionally estimate a large slip which was not there. Our no-slip prior could not really help in this case, because sometimes slip was real and we could not just ignore it. Filtering did not help either, because the dynamics involved many contacts, and so the error spikes were difficult to distinguish from real spikes in the IMU output due to large contact forces. The end effect was that MPC was occasionally given a state estimate with large slip (or some other dynamic abnormality) that was not real, and it naturally attempted to "fix" it, thus causing a real disturbance. This vicious cycle was very difficult to escape. Unfortunately we uncovered this Gazebo glitch on the first day of the competition, and it was too late to request fixing it.

Coming back to MPC, getting the machinery to work in real-time on a simulation as complex as Atlas was a significant achievement. Figure 3 shows histograms of the "policy lag", which corresponds to how long it took MPC to update the plan. This time interval is dependent on the state of the robot (states with many contacts were slower to compute), but nevertheless we observed consistent performance; the maximum lag ever recorded was 36 msec.

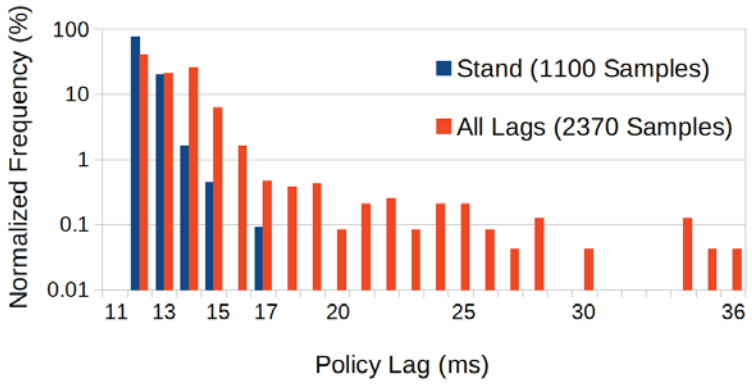

Figure 3. Histogram showing how long it took for MPC to update the plan. This is called Policy Lag, because it also corresponds to how old is the plan that is currently being executed.

Even though we did not advance to the next phase of the DRC, the system we developed is a significant step towards a universal robotic controller. The state estimation issue we encountered in the VRC is not likely to be a problem in real applications with real sensors. The main challenges we need to address are the need for longer planning horizons, and the issue of model errors. We have found locomotion tasks to be surprisingly forgiving to (simulated) model errors, possibly because gravity works to create contacts and contact forces in a predictable way. Object manipulation appears to be more sensitive in this regard, and we are now pursuing several ideas for extending MPC to operate with non-negligible model errors. Preliminary results illustrated below are quite promising. Here the task is to make the object achieve positions and orientations specified by the user online (with the 6D mouse on the bottom right of the figure). MPC computes the control signals needed to make the "fingers" (which are Phantom robots) establish and maintain contact with the object, and apply contact forces such that the object does what it was required to do.

Figure 4. MPC of dexterous object manipulation with a 12-dof "hand" made of 4 Phantom robots. This figure is from Erez et al, submitted to ICRA 2014.

5.0 CONCLUSIONS

In this project we attempted to develop a universal robotic controller, which was intended to solve the VRC among many other problems. We did not have the time and manpower to both develop this universal machinery and sufficiently customize it to the VRC, especially in the suboptimal conditions created by Gazebo. Nevertheless we believe that our approach holds great promise, and has the potential to transform robotic control in the next few years. We still have a lot of work to do, in particular increasing the planning horizon as well as the quality of the MPC solutions, and addressing unavoidable model errors in a principled way. We do not expect this to be routine; instead some of this work may require fundamental advances that remain to be made. Nevertheless, the ability to solve hard control problems in simulation with MPC, without need for simplifications or model reductions, is a significant step forward.

6.0 REFERENCES

Erez,T., Lowrey, K., Tassa, Y., Kumar, V. and Todorov, E. "An integrated system for model-predictive control of humanoid robots." To appear in proceedings of *Humanoids*, Atlanta 2013.

Erez,T., Kolev, S. and Todorov, E. "Receding-horizon online optimization for dexterous object manipulation." Submitted to *ICRA* 2014.