Real-Time Perception Meets Reactive Motion Generation

Daniel Kappler, Franziska Meier, Jan Issac, Jim Mainprice, Cristina Garcia Cifuentes, Manuel Wüthrich, Vincent Berenz, Stefan Schaal, Nathan Ratliff, and Jeannette Bohg

Abstract—We address the challenging problem of robotic grasping and manipulation in the presence of uncertainty. This uncertainty is due to noisy sensing, inaccurate models, and hard-to-predict environment dynamics. We quantify the importance of continuous, real-time perception and its tight integration with reactive motion generation methods in dynamic manipulation scenarios. We compare three different systems that are instantiations of the most common architectures in the field: 1) a traditional sense-plan-act approach that is still widely used; 2) a myopic controller that only reacts to local environment dynamics; and 3) a reactive planner that integrates feedback control and motion optimization. All architectures rely on the same components for real-time perception and reactive motion generation to allow a quantitative evaluation. We extensively evaluate the systems on a real robotic platform in four scenarios that exhibit either a challenging workspace geometry or a dynamic environment. We quantify the robustness and accuracy that is due to integrating real-time feedback at different time scales in a reactive motion generation system. We also report on the lessons learned for system building.

Index Terms—Reactive and sensor-based planning, perception for grasping and manipulation, sensor-based control.

I. INTRODUCTION

Robotic systems that integrate perceptual feedback into their planning and control loops have been developed for relatively low-dimensional control problems such as autonomous driving or flying [1], [2]. These systems are now mature enough to be the verge of becoming consumer products. For problems that require controlling many degrees of freedom (DoF) and physically interacting with the environment, it remains an open question how to best integrate perception and motion generation to allow for reactive behavior in the face of uncertainty. This is despite the fact that high-performance components for both visual tracking and reactive planning have been proposed in recent years. The teams who participated in the recent robotics challenges (DARPA Robotics Challenge (DRC) [3], [4] or Amazon Picking Challenge (APC) [5], [6]) testified to the importance of fast perceptual feedback integrated into planning and control loops.

In this letter, we present an instantiation of a robotic system which tightly integrates real-time perception with reactive motion generation for autonomous manipulation. We use visual perception to simultaneously track the target object and robot arm, and to obtain a geometrical representation of the workspace obstacles. The object pose, workspace geometry and robot configuration are then consumed by both, a local controller at a high rate (1 KHz) and a continuous motion optimizer at a lower rate (5–10 Hz) resulting in a joined kinematic policy. This is a systems letter. It aims at drawing conclusions on system integration based on empirical evidence from an extensive number of experiments. We quantify the benefit of integrating real-time perceptual feedback and reactive motion generation in dynamic manipulation scenarios for high DoF systems. We compare to baseline systems that rely on the same components but process sensory information at different rates or optimize the motion over different time horizons. This letter proposes requirements for components rather than prescribing specific perceptual, planning or control modules. This quantitative evaluation on the level of integration is one aspect that makes this letter unique in relation to related work on robotic manipulation systems.

Another unique aspect is the complexity in our experimental scenarios: they contain dynamic target objects and a dynamic environment-conditions that are common in human-robot collaboration, household, or disaster relief scenarios. Compared to the aforementioned robot challenges (APC and DRC), our experimental scenarios consider a much smaller variety of manipulation tasks and are tested on a fixed-base platform. However in terms of dynamicity, our scenarios go beyond those considered in the challenges where the environment is static and only the robot

1As common in the Computer Vision area, we use the term real-time to indicate that the computation time required by the perception methods are below the frame-rate of the depth camera, i.e., below 30 Hz.
2Throughout this letter we will use policy and controller interchangeably.
Fig. 1. One time step in our system with color coding of Fig. 2. Left to right: sensory input \( y \) (we overlay the position of target object [drill] at an earlier time step), perceived state \( s \) of the robot and the environment (target object, obstacles), local and global policies \( \pi^l, \pi^g \), and fused policy \( \pi \).

Fig. 2. Flow of information across three time steps: The perception modules continuously infer the latent state of the robot and the world from observed sensory input \( y \). The locally reactive control immediately translates this world state into a local policy \( \pi^l \). The continuous motion optimization computes a plan \( \pi^g \) for some time-horizon at a slightly lower rate. Reactive planning combines these two policies into one policy \( \pi \), which enables it to immediately react to local changes, and to look ahead in time to react to larger changes. Finally, this policy \( \pi \) produces a control output \( u \) (omitted for readability) which is sent to the robot.

interacts with it. Furthermore, we do not use any teleoperation as was the case in the DRC. We extensively evaluate the adaptivity, accuracy and robustness of the alternative systems in four different scenarios with varying degrees of geometric complexity and dynamicity. We draw the following conclusions: (i) Incorporating real-time feedback on different time-scales is crucial to achieve safe and successful task execution in uncertain, dynamically changing environments, (ii) the availability of reactive motion generation relaxes the requirement for perception systems to achieve maximum, one-shot accuracy since new, updated information can be consumed immediately.

II. SYSTEM ARCHITECTURES

We evaluate and compare three alternative system architectures along the spectrum from motion planning [5] to pure feedback control: (i) Sense-Plan-Act, (ii) Locally Reactive Control and (iii) Reactive Planning. We are aware that these names still contain some ambiguity. In the absence of existing terminology we define what we mean by them in each subsection and use them coherently in this letter. Figs. 1 and 2 present an overview of how the information flows between the perception and the motion generation modules in the different architectures. We discuss them here in relation to related work on robotic systems. A review of the vast body of work on visual tracking or motion planning is out of the scope of this letter and we refer to the respective sections in [7]–[10].

A. Sense-Plan-Act

Building systems through strong modularization into sensing, planning and acting components remains the predominant paradigm for high DoF robotic system building [11]. In this paradigm, perception provides a model of the environment, in which a motion planner finds an optimal, collision-free path that is then tracked by a stiff and accurate controller. In Fig. 2, this architecture corresponds to perception (in blue) and motion optimization components (in green), with visual feedback being considered only at the beginning of the motion planning task.

The advantage of this paradigm lies in the subdivision of the complex problem of robotic manipulation into intuitive subproblems that are easier to solve. Systems that are built according to sense-plan-act (SPA) are perfectly suited for environments that are well-defined, structured and controlled. However, they cope less well in the presence of uncertainty and a changing environment [6]. Due to the well known limitations of sense-plan-act [12], robotics researchers have proposed extension, e.g., sequential sense-plan-act (seqSPA), acknowledging the importance of environmental feedback during motion execution. Here, the robot does not only request feedback once at the very beginning but also at deliberately chosen moments during task execution. This approach is more robust against uncertainties in both sensing and actuation than SPA. However, it is still not able to cope with fast environment or target object dynamics. Some teams competing in the APC and in the DARPA ARM Challenge followed seqSPA [6], [13].

B. Locally Reactive Control

On the other end of the spectrum from feedback control to motion planning, we consider system architectures that rely entirely on visual feedback control. They do not have the global motion optimization modules (green in Fig. 2), but purely rely on local policies (red in Fig. 2) to generate reactive motion behavior. With local, we indicate that they only take the local geometry of the environment around the current manipulator pose into account to compute the optimal, immediately next control command. System architectures in this category react to changes immediately and are very robust to uncertainties in sensing and actuation. However, they may get stuck in local minima, for example when the environment has a complex workspace geometry [14].

Systems that are entirely based on feedback control have a long tradition. For example, [15] proposes a well defined interface for perceptual feedback in form of potential fields constructed from closest points. The resulting feedback control law can be computed efficiently using the superposition property of potential fields in combination with additive control laws based on the desired motion and constraints such as joint limits.

Visual servoing [16] broadly refers to the class of locally reactive control that closes the loop around visual data. More recently, there has been a lot of work on learning motion policies directly from perceptual feedback in form of raw camera images and the system joint state, e.g., [17]. Another example comes from the team who won the first APC [5]. Although they do not close the loop around visual data, they demonstrate the robustness of standard joint space and operational space controllers. Eppner et al. [5] admit that locally reactive control approach may have limitations in more complex manipulation tasks that require planning.
C. Reactive Planning

Summarizing the above, locally reactive control gives robotic systems the ability to immediately consume new information, instantly react to changes and compensate for inaccuracies. However, it is susceptible to local minima. Motion planning as typically used in sense-plan-act finds solutions even in complex situations where feedback controllers may get stuck. However, this comes at a significant computational cost that may break real-time requirements.

Ideal would be a hybrid system that combines both reactive motion planning and locally reactive control. Such a system can simultaneously adapt locally but also re-plan in case of larger changes. Compared to SPA, these systems are much more reactive and faster in completing the manipulation tasks. They rely on two motion generation modules, as depicted in Fig. 2 the global motion optimization (green) and local policies (red). The motion representations of both modules need to be fused to generate one policy for motion generation (yellow).

Combining local control with motion planning is quite common in the area of mobile robots, e.g., [1], [2], [18], [19]. However, fewer approaches exist to date that scale up such a hybrid system to robots with many degrees of freedom which are manipulating the world due to the increasing planning complexity.

One example is the elastic-strip framework [20] which combines local control with motion planning. The use of on-board vision sensing is suggested and demonstrated on a real robot platform but with a simulated, potentially changing world model. Controller funneling [21] is another hybrid approach which takes perceptual information into account. This approach requires a-priori knowledge to design the state space partitioning controllers and their switching conditions.

Dynamic Movement Primitives (DMPs) [22] can be interpreted as a combination of local control and a locally generalizing trajectory generator. Feedback terms can be learned and incorporated [23] for instantaneous reaction. Furthermore, perceptual feedback can be used to dynamically switch DMPs [24], which in turn results in local reactive control policies. [25] presents a mobile manipulation system that locally adapts and augments global motion plans in response to changes in the environment as perceived by on-board sensors. [26] present a system to find valid stance and collision-free reaching configurations in complex, dynamic environments for a full humanoid robot.

In this letter, we compare instantiations of each of these three different architectures that consist of the same components. In the next sections, we briefly describe these components and the interface between them. Section VI presents experimental results that are then discussed in Section VII.

III. REAL-TIME FEEDBACK MODULES

We consider scenarios that require manipulation of dynamic objects in dynamic environments. Continuous feedback on the location of target objects and the workspace structure is of utmost importance for systems acting in these scenarios. An important requirement for the feedback components is therefore to deliver information as fast as possible. A low-dimensional representation of this information is beneficial to keep bandwidth requirements low.

In the following, we briefly describe the methods integrated into our system. Any other method which fulfills the requirements could be used instead.

A. Visual Tracking of Target Objects

We use visual tracking to estimate the pose of every object the robot may want to manipulate. We choose the probabilistic method from our previous work [7]. This method assumes knowledge of the 3-dimensional shape of the objects of interest, represented as triangle meshes. It takes as input depth images and compresses it into 6 DoF object poses at the frame rate of the on-board camera. Its formulation makes it very robust to occlusions of object parts which are common in the context of manipulation tasks.

B. Visual Robot Tracking

Precise positioning of a robot arm with respect to the sensed environment and target object is a crucial ability for manipulation systems. This is not always possible through naive application of forward kinematics. On real robotic platforms, kinematic models and measured joint angles are commonly inaccurate and therefore lead to erroneous predictions of end-effector pose relative to the camera.

To mitigate this problem, we continuously estimate the true robot arm configuration relative to the camera mounted on the robot’s head. We choose the probabilistic, real-time method from our previous work [8]. It fuses depth images and measured joint angles to produce precise estimates of the robot configuration at 1 kHz which is the rate of the joint encoders. Even under heavy occlusion of the arm and very fast motion, this method can correct errors due to biases in the joint sensors and to imprecise kinematics of the camera relative to the rest of the kinematic chain. Furthermore, it models the delay between the measurements from the joint sensors and the camera.

C. Modeling Unstructured Workspace Obstacles

To generate collision-free motion, the robot needs to be aware of the workspace geometry and the obstacles therein. Commonly, this is encoded probabilistically and at multiple scales in occupancy grids by integrating depth measurements over time, e.g., using OcioMaps [27].

These approaches come at a significant computational cost that does not allow for reactive behaviors. Hence, we represent obstacle regions in a discrete binary occupancy grid which is newly generated frame by frame. This is performed by transforming each point cloud to world coordinates while removing the points outside the robot workspace. This transformed point cloud is then converted to an occupancy grid (with 2 cm resolution), which is filtered for statistical outliers using PCL [28]. Finally the voxels corresponding to the robot arms and the tracked object are set empty (see Fig. 1) and the occluded regions are set occupied using ray casting [27].

These filtering and transformation steps can be performed at approximately 15 Hz. We found this rate to be sufficient in practice.

IV. FROM VISUAL FEEDBACK TO CONTINUOUS SIGNED DISTANCE FIELDS

Given the processed perceptual data as described in the previous sections, we convert it into a set of Signed Distance Fields (SDFs) describing the target object, table, unstructured workspace obstacles, and the robot. SDFs provide a way to perform fast collision checking, which has been exploited in many motion optimization works [9], [29], [30]. SDFs also allow to define proper Riemannian metrics to measure path length in workspaces populated by obstacles [9], which we make use of in sense-plan-act and reactive planning.

To compute the SDFs, we rely on a combination of analytic formulations and distance propagation algorithms. The manipulated objects, the table and the robot links are represented by simple geometrical shapes such as spheres, boxes and capsules (see Fig. 3). These shape primitives allow for analytic SDF formulations. For the occupancy grid, the SDF is computed using a voxelgrid dis-
A. Locally Reactive Control

Locally Reactive Control combines multiple controllers through (1), including collision controllers to instantaneously react to the local workspace geometry, and target controllers for goal convergence. The target controller pulls the system toward position and orientation targets in a purely local Cartesian control fashion. This portion of the system can therefore be used by itself, without any higher level planning. It is visualized in Fig. 1 with red arrows. In addition to the target controller, we use a collection of obstacle avoidance controllers that take effect when parts of the body get close to an obstacle. They create workspace accelerations away from the obstacle with increasing priority as a function of proximity. They are visualized in Fig. 3. We also use a default posture potential that pulls the arms slightly toward a default posture to resolve redundancy, along with simple damping controllers in the e-space and at the end-effector to regulate the velocity of the system. Local control runs at 1 kHz for effective integration of the underlying highly nonlinear differential equation. However, it sends joint positions, velocities and accelerations for low-level execution only at 100 Hz. The lowest level of control handles the generation of the torques needed to track the desired joint states through interpolation and an inverse dynamics controller.

B. Reactive Planning and Continuous Motion Optimization

We use a motion optimizer based on Riemannian Motion Optimization (RieMO) [9] as the basis for reactive planning. It runs continuously, tracking the local minimum based on feedback while obstacles and the target change over time. This motion optimizer integrates information over a time horizon of three seconds (the (approximate) average time length of a reaching motion), enabling anticipatory behaviors and efficient coordination of multiple controllers handling collision avoidance, position targets, orientation targets, etc. As is sometimes done in optimal control and MPC, it summarizes its policies as Linear Quadratic Regulators (LQRs) built on a local quadratic approximation around the local optimum. However, this is done only kinematically (with accelerations as control variables) since the planning module addresses only movement. This is visualized in Fig. 1 with green arrows. These are sent to locally reactive control for integration with the other controllers through (1). The continuous optimizer operates at a slower time-scale than locally reactive control, updating its optimization at 5–10 Hz. To mitigate potential delays, it sends full LQR policies that represent the optimal policy within a region of the locally optimal trajectory. Rather than using a simple attractive potential pulling the end-effector toward a desired pose (which can be expressed as a differential equation in the configuration space) the motion optimizer creates a more expressive attractor differential equation that simultaneously pulls the system toward the desired target while also integrating anticipatory actions (e.g., rotating the wrist to avoid future obstacle) that enable smoother, more efficient, and well-coordinated behavior. Therefore, the local controllers are able to operate cohesively with the planned policies between planning updates.

C. Grasping

We decompose the grasp problem into multiple sequential task states—approach, establish grasp, move object, release, and retract—each governed by either local control, continuous optimization, or some combination thereof. Grasping is controlled independently of the other motion generation modules through force feedback in the fingers. The rest of the system observes the resulting movement of the hand and reacts accordingly to simultaneously adjust the arm to avoid obstacles and stabilize the hand posture to
the extent possible under the constraints of the environment. This enables us to setup consistent experimental scenarios for empirical study. We manually defined a set of grasp poses for each object that we use in our experiments.

VI. EXPERIMENTS AND DEMONSTRATIONS

We compare three different system architectures that were explained in more detail in Section II: (i) sense-plan-act, (ii) locally reactive control and (iii) reactive planning. As an experimental platform, we use a fixed-base, manipulation platform equipped with two 7-DoF Kuka LWR IV arms, three-fingered Barrett Hands and an RGB-D camera (Asus Xtion) mounted on an active humanoid head by Sarcos. All components are torque controlled using an inverse dynamics controllers to track the desired joint states. It runs at determinable worst-case execution times of 1 ms and is executed on a PC running Xenomai, a real-time framework for Linux. We placed system components that interact frequently onto the same computer (tracking and SDFs, locally reactive control and motion optimization) to meet the computational requirements and network bandwidth of the different system components.

A. System Architecture Realizations

The visual perception modules (Section III) consist of a tracker for the right robot arm, and a tracker for the target object. We assume a known table pose. Everything else in the environment is considered to be unstructured workspace obstacles modeled by an occupancy grid map. The algorithms used in all our implementations of the different architectures are identical. We vary the frequency at which information is passed to motion generation and whether we consider policies that are optimized over a longer time horizon.

1) Sense-Plan-Act: Here, we acquire just one depth image in the very beginning of the experiment. Based on this image, the poses of the objects of interest are estimated, and a model of the workspace geometry is created. Then a one-shot motion optimizer, a simple variant of Section V-B, generates a plan which will be executed without any further visual feedback. The overall planning time of sense-plan-act is limited to 2 s for all experiments. This threshold has been chosen empirically, trading off convergences success and potential restarting of the planner in case no solution can be found.

2) Locally Reactive Control: In this architecture, depth images are processed continuously to estimate the object pose and robot arm configuration. Additionally, the world model is updated online. This information is consumed by locally reactive control (see Section V-A) that immediately adapts to the observed changes in the next control cycle.

3) Reactive Planning: As in the previous architecture, the object and robot arm tracker continuously estimate the object pose and robot arm configuration. Also the world model is updated online. However, here the information from the perception modules is also used to continuously replan in addition to the locally reactive control (see Section V-B).

B. Scenarios

We present four different scenarios (see Fig. 4) and experimental results which illustrate the importance of tightly integrating real-time perception and reactive motion generation. Each experiment parameterization is performed at least 3 times.

1) Pick and Place in Static Environments of Increasing Difficulty: In this experiment we consider the static pick and place scenario shown in Fig. 4(a). The task is to pick up the pringles box and place it on the other side of the brown box, without any collisions. The box is always placed prior to starting each experiment. The closer the box to the robot base, the higher the difficulty to successfully pick and place the pringles. For each system architecture and complexity level we run three trials. At position 15, we reached the point where each system failed at least once. Table I shows the success rate of picking up the object and placing it at the target location. Even though the planning problem itself becomes very challenging, locally reactive control alone

Fig. 4. Experimental scenarios: The human hands indicate which objects are being moved during execution. (a) Static pick and place. (b) Dynamic pick and place. (c) Dynamic grasping. (d) Dynamic pointing.

Fig. 5. Visualization of the successful end-effector trajectories in a pick and place task in the presence of a box obstacle (cf. Table I). Per experiment, the box varies its pose from –10, 0, 5, 10 or 15 cm distance to the far edge of the table (as indicated by the dashed lines). The black labels indicate which trajectory belongs to which box position. The black dots indicate the start of each trajectory. The blue dot indicates the picking object position and the cross its placing position. We compare locally reactive control (Left), reactive planning (Middle) and sense-plan-act (Right).
already performs very well. Not surprisingly, sense-plan-act performs very well in such a static environment. However, in the most challenging setting it does fail more often compared to reactive planning. One reason for this is the limited planning time allocated, during which no successful plan may be found. Reactive planning is able to find a path more often since it is able to re-plan continuously, thus, has more time to find a feasible path during execution. In Table II we report the average execution time for successful trials in seconds. Time required for the initial object detection is not part of the execution. Locally reactive control and reactive planning are on par for the simple settings, whereas sense-plan-act is significantly slower. The difference in execution time is because sense-plan-act can only start planning to the next pose after it achieved the old pose. It then has to wait until a solution has been found. The execution time increases with difficulty due to more confined workspace which results in slower convergence especially for locally reactive control and in general for longer trajectories. This scenario illustrates that even in static environments, the two tightly integrated system architectures can have benefits over sense-plan-act. This experiment emphasizes the importance of continuous feedback integration into motion generation, when the target is repositioned after motion onset.

3) Grasping With Dynamic Targets: Not only the unmodeled environment is subject to constant change, but very often also target objects may move when reaching for them. In this scenario, we systematically analyze the importance of perceptual feedback integration into motion generation, when the target is repositioned.

Table II reports the average execution time (in seconds) of successful pick-and-place experiments. The robot has a maximum of five seconds to find a feasible path and place the object at the goal position.

In conclusion, this experiment shows the importance of real-time perception to avoid collisions with unmodeled obstacles. In addition, this experiment illustrates that having reactive planning is important to find collision free paths in complex dynamically changing environments whereas locally reactive control is safe but gets stuck more easily.

4) Pointing in Dynamic Environments With Dynamic Target: In our most complex scenario, we want to analyze the accuracy and reactivity to simultaneous changes in the environment and target pose. The task of the robot is to align its fingertip with the tip of a drill (see Fig. 4(d)).

We have four different levels of complexity for this task. Level 1: As a baseline we start with a static environment without obstacles while the drill is stationary. Level 2: We introduce a blocking obstacle (box) during execution, which can be avoided by going
Fig. 6. Minimum distances (i.e., clearance) with the introduced obstacle during the dynamic pick and place experiment. Locally reactive control gets stuck after successfully grasping the object. Reactive planning while slower than the locally reactive control to realize the grasping motion is able to perform the place successfully. Sense-plan-act despite higher initial clearance than locally reactive control collides with the obstacle when its position changes. (a) Locally reactive control. (b) Reactive planning. (c) Sense-plan-act.

Fig. 7. Visualization of successful grasps of in case of a (a) repositioned target and (b) repositioned and flipped target after motion onset. In both (a) and (b) we compare locally reactive control (Left), reactive planning (Middle) and sense-plan-act (Right). (Top) Grid of possible target object poses after motion onset. The grid is 24 \times 24 \text{ cm} large. The central dot marks the initial target object position. Dark blue dots indicate that all three grasp attempts were successful. Red crosses indicate positions at which no grasp trial succeeded. (Bottom) Visualization of the end-effector trajectories in the corresponding color per approach for successful grasps. Black dots indicate the starting position. Blue dots indicate the target pose. (a) Both, locally reactive control and reactive planning grasp successfully in the entire region of variation and successfully adapt the trajectory given the new feedback on target pose. Locally reactive control is on average slightly faster than reactive planning (b) Again, both, locally reactive control and reactive planning grasp successfully in the entire region of variation and successfully adapt the trajectory given the new feedback on target pose. Here, reactive planning is on average a bit faster (time in seconds \pm one standard deviation). In both settings, sense-plan-act manages to successfully grasp the target when the new position is in close to the path to the original target location.

| Difficulty   | Loc. React. Ctrl | Reactive Planning | Sense-Plan-Act |
|--------------|------------------|-------------------|----------------|
| static       | 100\% (3)        | 100\% (3)         | 100\% (3)      |
| straight     | 100\% (3)        | 100\% (3)         | 0\% (3)        |
| diagonal     | 100\% (3)        | 100\% (3)         | 0\% (3)        |
| turning      | 100\% (3)        | 100\% (3)         | 0\% (3)        |

around it. Level 3: The obstacle is moved into the way such that the arm has to move over the obstacle or take a big detour. Level 4: We start out with a blocking obstacle. After the system starts moving we remove the obstacle while also changing the orientation of the drill by 90 degrees. The reorientation of the drill means that the pointing approach has to be adapted.

We report the results for this scenario in Table III. We define success as reaching the tip of the drill up to a distance of 3 cm without a collision with any environmental obstacle. Neither locally reactive control nor reactive planning collide with any obstacle in this experiment. Sense-plan-act however collides with the blocking obstacle for both the Level 2 and Level 3 experiment. In the case of the Level 4 experiment, sense-plan-act was able to reach the initial position of the drill tip. Since no perceptual feedback is considered it was not aware of the rotation whereas both locally reactive control and reactive planning could even shorten the path towards the drill tip by taking into account the removed obstacle. For optimal performance, Level 4 requires both continuous tracking of the target (similar to Grasping with Dynamic Targets) and updates of the workspace obstacles (similar to Pick and Place in Dynamic Environments).

This experiment supports our hypothesis that integrating real-time perception with motion generation is key for task success and safe behavior in highly dynamic scenarios.

VII. CONCLUSION, LESSONS LEARNED, FUTURE WORK

Already in the 80’s [12], it has been postulated that tightly integrating real-time perception and reactive motion generation is beneficial if not even required for robotic systems that physically interact with uncertain and dynamic environments. To quantify the benefits of this integration for a high DoF robotic manipulation system, we compared three different systems (sense-plan-act, lo-
cally reactive control and reactive planning) with a varying level of integration between perception and motion generation.

We have shown that already locally reactive control which integrates perceptual feedback at the highest possible rate can be very efficient in simple tasks while being safe by avoiding collisions with the environment. Reactive Planning achieves a better performance in more complex environments due to its ability to look ahead. Sense-plan-act performs well in static scenarios as expected, but even there locally reactive control and reactive planning have advantages as they can start moving earlier while continuing to consume feedback. We also observed that on the trade-off curve between perceptual accuracy and computational speed, it is more beneficial to have fast feedback than accurate world representations. This is especially the case for dynamic and uncertain manipulation scenarios where a fast reaction to sudden changes or new incoming information is key. As communication bandwidth is limited, this also places constraints and how much information can be transferred between components. Therefore, we opted for model-based visual tracking and querying SDFs only for a small subset of points on the robot. Data association was extremely important and we therefore carefully synchronized the different sensory and information streams across the three different computers. We observed that tuning parameters like safety margins was straightforward as the underlying models of the system have an intuitive interpretation. These parameters were also invariant across the four different scenarios.

In the current architecture, we mostly take visual and joint encoder feedback into account. However, manipulation tasks are heavily concerned with contact interaction. We use the finger strain gauges as feedback in the grasp controller. Our system would however benefit from also taking haptic feedback from tactile sensor arrays or force/torque sensors into account [24]. Currently, we are optimizing motion for obstacle avoidance. However, exploiting contact constraints during manipulation has been shown to increase robustness [13], [33], [34]. Our system also does not rely on any learning yet. However, there is a large potential in e.g., learning representations of the perceptual data instead of prescribing it ourselves. Another interesting research direction is the integration of online inverse dynamics learning to cope with changed dynamics after picking up objects [35], [36].

REFERENCES

[1] J. Leonard et al., “A perception-driven autonomous urban vehicle,” J. Field Robot., vol. 25, no. 10, pp. 727–774, 2008.
[2] C. Richter, A. Bry, and N. Roy, Polynomal Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments. Cham, Switzerland: Springer-Verlag, 2016, pp. 649–666.
[3] C. Akeson et al., “No falls, no resets: Reliable humanoid behavior in the DARPA robotics challenge,” in Proc. 2015 IEEE-RAS. 15th Int. Conf. Humanoid Robots, 2015, pp. 623–630.
[4] M. Johnson et al., “Team IHMC’s lessons learned from the DARPA robotics challenge trials,” J. Field Robot., vol. 32, no. 2, pp. 192–208, 2015.
[5] C. Eppner et al., “Lessons from the amazon picking challenge: Four aspects of robotic systems building,” in Proc. Robo, Sci. Syst., 2016, pp. 4831–4835.
[6] N. Correll et al., “Analysis and observations from the first amazon picking challenge,” IEEE Trans. Autom. Sci. Eng., vol. 15, no. 1, pp. 172–188, Jan. 2018.
[7] M. Wüthrich, P. Pastor, M. Kalakrishnan, J. Bohg, and S. Schaal, “Probabilistic object tracking using a range camera,” in Proc. 2013 IEEE/RSJ Int. Conf. Intell. Robots Syst., 2013, pp. 3195–3202.
[8] C. Garcia Cifuentes, J. Issac, M. Wüthrich, S. Schaal, and J. Bohg, “Probabilistic articulated real-time tracking for robot manipulation,” IEEE Robot. Autom. Lett., vol. 2, no. 2, pp. 577–584, Apr. 2017.
[9] N. Ratliff, M. Toussaint, and S. Schaal, “Understanding the geometry of workspace obstacles in motion optimization,” in Proc. 2015 IEEE Int. Conf. Robot. Autom., 2015, pp. 4202–4209.
[10] J. Mainprice, N. Ratliff, and S. Schaal, “Warping the workspace geometry with electric potentials for motion optimization of manipulation tasks,” in Proc. 2016 IEEE/RSJ Int. Conf. Intell. Robots Syst., 2016, pp. 3156–3163.
[11] D. Kortenkamp, R. G. Simmons, and D. Brugali, “Robotic systems architectures and programming,” in Springer Handbook of Robotics. New York, NY, USA: Springer-Verlag, 2016, pp. 283–306.
[12] R. A. Brooks, “Elephants don’t play chess,” Robot. Auto, Syst., vol. 6, no. 1/2, pp. 3–15, 1990.
[13] L. Righetti et al., “An autonomous manipulation system based on force control and optimization,” Auton. Robots, vol. 36, no. 1/2, pp. 11–30, 2014.
[14] J. Kuffner and J. Xiao, “Motion for manipulation tasks,” in Springer Handbook of Robotics. Cham, Switzerland: Springer-Verlag, 2016, pp. 897–930.
[15] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” Int. J. Robot. Res., vol. 5, no. 1, pp. 90–98, 1986.
[16] F. Chummette, S. Hutchinson, and P. Corke, “Visual servoing,” in Springer Handbook of Robotics. New York, NY, USA: Springer-Verlag, 2016, pp. 841–866.
[17] S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-end training of deep visuomotor policies,” J. Mach. Learn. Res., vol. 17, no. 1, pp. 1334–1373, 2016.
[18] D. Ferguson and A. Stentz, “Using interpolation to improve path planning: The field D* algorithm,” J. Field Robot., vol. 23, no. 2, pp. 79–101, 2006.
[19] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, “Real-time motion planning with applications to autonomous urban driving,” IEEE Trans. Control Syst. Technol., vol. 17, no. 5, pp. 1105–1118, Sep. 2009.
[20] O. Brock and O. Khatib, “Elastic strips: A framework for motion generation in human environments,” Int. J. Robot. Res., vol. 21, no. 12, pp. 1031–1052, 2002.
[21] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek, “Sequential composition of dynamically dexterous robot behaviors,” Int. J. Robot. Res., vol. 18, no. 6, pp. 534–555, 1999.
[22] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, “Dynamical movement primitives: Learning attractor models for motor behaviors,” Neural Comput., vol. 25, pp. 328–373, Feb. 2013.
[23] A. Rai, G. Sutanto, S. Schaal, and F. Meier, “Learning feedback terms for reactive planning and control,” in Proc. 2017 IEEE Int. Conf. Robot. Autom., 2017, pp. 2184–2191.
[24] D. Kappler, P. Pastor, M. Kalakrishnan, M. Wüthrich, and S. Schaal, “Data-driven online decision making for autonomous manipulation,” in Proc. Robot., Sci. Syst., Rome, Italy, 2015.
[25] P. Lehner, A. Sieverling, and O. Brock, “Incremental, sensor-based motion generation for mobile manipulators in unknown dynamic environments,” in Proc. 2015 IEEE Int. Conf. Robot. Autom., May 2015, pp. 4761–4767.
[26] Y. Yang, V. Ivan, Z. Li, M. Fallon, and S. Vijayakumar, “IDRM: Humanoid motion planning with real-time end–pose selection in complex environments,” in Proc. 2016 IEEE-RAS. 16th Int. Conf. Humanoid Robotics, 2016, pp. 271–278.
[27] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, “OctoMap: An efficient probabilistic 3d mapping framework based on octrees,” Auton. Robots, vol. 34, pp. 189–206, 2013.
[28] R. B. Rusu and S. Cousins, “3d is here: Point cloud library (PCL),” in Proc. 2011 IEEE Int. Conf. Robot. Autom., 2011, pp. 1–4.
[29] M. Zucker et al., “CHOMP: Covariant hamiltonian optimization for motion planning,” Int. J. Robot. Res., vol. 32, pp. 1164–1193, 2013.
[30] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, “STOMP: Stochastic trajectory optimization for motion planning,” in Proc. 2011 IEEE Int. Conf. Robot. Autom., 2011, pp. 4569–4574.
[31] N. D. Ratliff, J. Issac, and D. Kappler, “Relmmanuel motion policies,” arXiv:1801.02854, Jan. 2018.
[32] M. Toussaint and C. Goerick, “A bayesian view on motor control and learning,” in From Motor Learning to Interaction Learning in Robots. New York, NY, USA: Springer-Verlag, 2010, pp. 227–252.
[33] M. Kajiwara, J.-S. Valois, J. A. Bagnell, and N. S. Pollard, “Robust object grasping using force compliant motion primitives,” in Proc. Robot., Sci. Syst., 2012.
[34] R. Deimel, C. Eppner, J. Álvarez-Ruiz, M. Maertens, and O. Brock, “Exploration of environmental constraints in human and robotic grasping,” in Proc. Int. Symp. Robot. Res., 2013, pp. 393–409.
[35] N. Ratliff, F. Meier, D. Kappler, and S. Schaal, “Doomed: Direct online optimization of modeling errors in dynamics,” Big Data, vol. 4, no. 4, pp. 253–268, 2016.
[36] F. Meier, D. Kappler, N. Ratliff, and S. Schaal, “Towards robust online inverse dynamics learning,” in Proc. 2016 IEEE/RSJ Int. Conf. Intell. Robot. Syst., 2016, pp. 4034–4039.