# Arm Teleoperation in Clutter Using Virtual Constraints from Real Sensor Data

Adam Leeper[1,2], Kaijen Hsiao[2], Matei Ciocarlie[2], Ioan Sucan[2], and Kenneth Salisbury[1]

[1]Stanford University, Stanford, California, USA
[2]Willow Garage, Inc., Menlo Park, CA, USA

***Abstract*—We introduce CAT, a constraint-aware teleoperation method that can track continuously updating 6-DOF end-effector goals while avoiding environment collisions, self-collisions, and joint limits. Our method uses sequential quadratic programming to generate motion trajectories that obey kinematic constraints while attempting to reduce the distance to the goal with each step. Environment models are created and updated at run-time using a commodity depth camera. We compare our method to three additional teleoperation strategies, based on global motion planning, inverse kinematics, and Jacobian-transpose control. Our analysis, using a real robot in a variety of scenes, highlights the strengths of each method, and shows that the CAT method we introduce performs well over a wide range of scenarios.**

## I. Introduction

Deployment of mobile manipulators in human settings could be accelerated by including human-in-the-loop system components. Teleoperation could be used for the more-difficult components of a task, leaving the rest to autonomy, or as a fallback option if autonomy fails. However, teleoperation can be tedious and difficult; robot arm motion in particular presents several challenges that can hinder an operator from focusing on task completion. Highly-articulated arms often have non-anthropomorphic configurations, non-intuitive workspaces and joint limit constraints. Controlling such arms is made more difficult when trying to also avoid collisions with objects in cluttered human environments.

One way to address this problem is to create tools that leverage autonomous capabilities even during operator-guided motions. In this paper, we explore how to use control and planning methods to allow an operator to command an end-effector pose without worrying about kinematics or collision avoidance, leaving those tasks to the robot. Along the way we introduce CAT, a locally-optimal trajectory generation strategy based on sequential quadratic programming [13, 17, 18, 20].

### A. Problem Description and Goals

Our problem set-up includes a robot equipped with a dexterous (7-DOF) arm. A remote operator controls the robot's arm by moving a Cartesian end-effector goal pose, and the robot continuously tries to track the current goal. Focusing on applicability to complex tasks in unstructured and cluttered environments, the problem is set up as follows:

- The goal is specified as a 6-DOF gripper pose.
- The operator can continuously update the goal, without first waiting for the robot to achieve previously-selected poses.
- The robot does not have a pre-constructed map of obstacles in the environment, but instead maintains a continuously-updating model of obstacles using a live stream of data from its sensors (in our case, a Kinect depth camera).

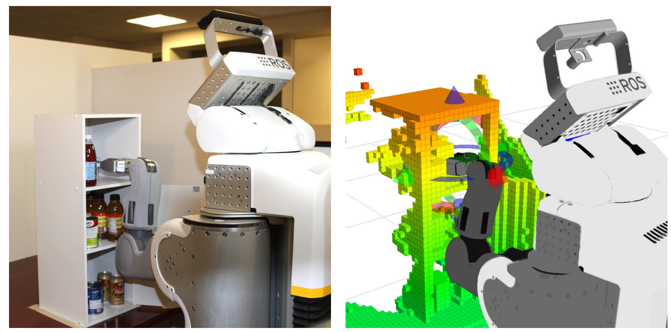


Fig. 1. Left: a teleoperated robot performing a manipulation task in a cluttered environment. Right: the scene as seen by the operator, with visual controls (colored rings and arrows) used for specifying end-effector pose goals, and the robot's collision map temporarily overlaid for visualization.

Our goal in this paper is to compare methods for tracking the end-effector goal specified as above, while enabling the user to focus on task completion rather then joint-level constraints. At each time step, the chosen arm control method takes as input the current configuration of the arm as well as the current goal pose, and outputs joint torques. In designing our arm control methods, we aspire to the following goals:

- Tracking: the end-effector should track the goal pose. When the goal is unachievable (as defined by the constraints below), the end-effector should follow as closely as possible.
- Collision avoidance: the robot should avoid undesired collisions anywhere on the arm.
- Joint limit avoidance: the arm configuration should avoid having joints close to or against their limits, to increase manipulability and make future goals easier to achieve.

### B. Contributions

This work is the first, to the best of our knowledge, to examine the problem of tracking a continuously-updating 6-DOF end-effector pose, while avoiding contact with a volumetric collision model generated in real-time from visual sensing.

To solve this problem, we introduce a constraint-aware teleoperation (CAT) technique for performing Cartesian end-effector control with the following characteristics:

- Takes input in 6-DOF Cartesian space, allowing for general end-effector movement while not requiring the operator to reason about the kinematics of the arm.
- While not limited to straight-line motion towards the goal, it attempts to reduce pose error relative to the goal at each time step, in an attempt to balance fast goal tracking with the flexibility needed to obey constraints.
- Avoids violating (observable) constraints for the entire arm, such as environment collisions and joint limits.
- Generates and continuously updates a collision model of the environment using live depth camera data.

In order to analyze the performance of CAT, we implemented three other teleoperation strategies for comparison, based on extensions to established arm-motion methods:

- Joint-space global motion planning (MP), using sampling-based planning to continuously replan as the goal changes.
- Collision-aware inverse kinematics (IK), combining IK for the goal pose, joint-space interpolation, and collision detection to find a collision-free trajectory.
- Jacobian transpose torque control (JT), with no additional constraint or collision avoidance capabilities.

In our experimental section we quantify and compare the behavior of each approach over a common set of experiments.

## II. Related Work

Our work draws from many areas of research in telerobotics, including control, haptics, and motion planning.

Dragan and Srinivasa [5] formalized the "do what I mean" problem in teleoperation by defining an *arbitration* or policy-blending component of the control loop. The CAT method (and each alternative) can be viewed as an arbitration strategy where the user provides input, the robot "predicts" geometric constraints from sensor data, and CAT creates a motion that attempts to satisfy both. Unlike [5], however, we are concerned with general arm motion rather than specific task prediction.

Control methods in telerobotics typically compute torques that "pull" a system toward one or more goals while pushing it away from obstacles. Potential fields [11] are one such method; circular fields [6] are better at avoiding local minima. Sentis and Khatib [19] outline a method for hierarchical control of task goals and constraints in which each goal is achieved as closely as possible using artificial potential fields in the nullspace of all higher-priority goals. Passenberg et al. [16] provides an excellent summary of several methods for improving bilateral teleoperation by adapting controllers to the environment, operator, or task.

We draw some inspiration from haptic rendering methods; we use a kinematic "proxy" for the robot state, and the pose error is expressed through a virtual-coupling [3]. (Our system can render the virtual coupling forces to the operator through a haptic device, but this paper does not explore that aspect.) Since we use a proxy and a sensor-based environment model to compute motions, our system is a type of model-mediated teleoperation [15]. In this context our motion constraints are also a form of forbidden-region virtual fixtures [1] created in real-time from sensor point clouds. Mitra and Niemeyer [14] used constraints from model-based geometric collision detection to avoid self-collisions and generate haptic force feedback while teleoperating two 6-DOF arms, but they did not do any environmental collision avoidance.

Randomized motion planning in arm configuration space is very popular in autonomous robot systems [4], but it has only recently reached the speed necessary for responsive performance with continuously updating goals. Hauser [8] tested a motion planning approach for teleoperation of 3-DOF tasks that is very similar to the MP method we use; however, our method uses 6- DOF end-effector goal poses, and is implemented and tested on a real robot using live sensor data for collision avoidance. Knepper et al. [12] proposed a hierarchical planning method to greatly improve responsiveness, using a rough global planner to guide a local planner.

Several recent studies have used sequential quadratic programming methods for fast generation of robot motion plans, though none have used these methods in teleoperation of a real robot or with an emphasis on real-time collision avoidance. Posa and Tedrake [18] focused on the problem of planning through rigid-body contact, applied to generation of walking trajectories; Werner et al. [20] similarly optimized walking motions. Lampariello et al. [13] created optimal trajectories for catching a ball in real-time, demonstrated in simulation. Pham and Nakamura [17] demonstrated trajectory deformation and motion stitching in response to new motion goals.

Most similar to our work, Jain et al. [10] used sequential quadratic programming to enforce joint limit and contact constraints while trying to reach a 6-DOF target. The contact constraints in their work are generated using force sensors and are intended to limit (but not eliminate) contact with obstacles. This strategy is complementary to ours; visual sensors alone cannot predict collisions with invisible or occluded obstacles, while force sensors alone cannot avoid contact with fragile or light obstacles for which even minor contact can be disastrous. Using a full collision map generated by visual depth sensing allows the CAT method to compute a multi-step trajectory (Sec. IV-C), producing smoother and faster motion. Detecting contact points only through force sensing means that the robot has no geometric model of obstacles before hitting them and thus cannot predict contacts in advance.

## III. Teleoperation System Overview

We begin with an overview of our system in order to provide context for the teleoperation strategies described later.

The input command is a 6-DOF end-effector pose read from the operator at 30 Hz. In our implementation, the pose can be commanded through mouse interaction with on-screen controls, or through a 6-DOF Razer Hydra input device.

Each teleoperation strategy is responsible for processing this input pose at every time step and sending a command to the appropriate low-level controller. Fig. 2 shows the system block diagram for three of our strategies: CAT (Sec. IV), MP (Sec. V), and IK (Sec. V). These three methods all generate

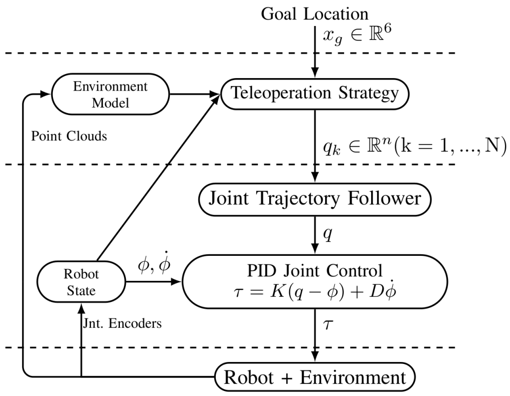



Fig. 2. *Block diagram showing the hierarchical control structure for the CAT, MP, and IK methods. The number of joints is denoted by $n$.*

a trajectory of joint configurations which is passed to a low-level joint trajectory follower. Quintic splines between joint trajectory points are computed and passed to a standard joint impedance controller. By contrast, the JT method (Sec. V) simply passes the goal pose directly to a low-level Jacobian-transpose controller not shown in the diagram. After processing a goal command, each strategy restarts using the most recent goal pose. Section IV-C describes how we ensure continuity of trajectories while the robot arm is in motion.

We assume that the robot has joint encoders for measuring its own kinematic configuration, and one or more sensors for perceiving the environment. Our implementation uses a commodity depth camera to build a continuously-updating volumetric representation of the world, which is used for predicting (and avoiding) collisions in the CAT, MP, and IK strategies as decribed in section IV-B.

## IV. Constraint-Aware Teleoperation Strategy

The first method we describe is the Constraint-Aware Teleoperation (CAT) controller, one of the main contributions of this study. In general terms, CAT creates joint trajectories by computing incremental joint position changes that reduce the value of a cost function while simultaneously obeying a set of constraints. It is designed to be used in a local fashion, quickly computing small steps away from a known joint configuration, but not expected to be globally optimal. We argue that in the domain of teleoperation it is valuable to have an option that combines responsiveness and constraint-awareness.

We formulate the problem as a instance of convex optimization, namely a quadratic program (QP) with linear inequality constraints that operates on a (time-varying) linearized model of the robotic system. The solution to the QP represents a small incremental change to the state of the system. The QP is reformulated after each step due to the nonlinear kinematics and because each new kinematic configuration may result in new constraints due to predicted contact with the environment. We note that each iteration of the QP solves for a single step, but in each command period the QP is run many times[1] to create a continuous trajectory (see section IV-C).

[1] We use CVXGEN, a tool that generates highly-optimized, problem-specific C-code (www.cvxgen.com). This allows each QP step to be solved orders of magnitude faster (around 200-400 $\mu$s) than if using a generic solver.

### A. Quadratic Program Formulation

We wish to compute joint deltas, $\Delta q$, that move us closer to a goal expressed as a quadratic objective function. In our QP, the optimization variables are $\Delta q \in \mathbb{R}^n$, where $n$ is the number of joints, and the objective is a sum of quadratic functions of the optimization variables. In particular, our **QP objective** is composed of three parts:

*1) Move toward a goal pose:* For a desired Cartesian goal pose, $x_d \in \mathbb{R}^6$, we compute the necessary pose change, $\Delta x_d$, from the proxy end-effector pose. Since our linear system model is only valid for small displacements the error is clipped to a maximum distance and angle of rotation when computing $\Delta x_d$. With the end-effector Jacobian $J \in \mathbb{R}^{6\times n}$ and a weighting matrix $W_x \in \mathbb{R}^{6\times 6}$, the objective term is:

$$(\Delta x_d - J\Delta q)^T \; W_x \; (\Delta x_d - J\Delta q) \tag{1}$$

*2) Discourage large joint changes:* This term encourages the optimized variables to stay small. Even for a small Cartesian movement, certain configurations can result in solutions with large joint displacements, which is an artifact of the linearized system. Defining a diagonal weighting matrix $W_{\Delta q} \in \mathbb{R}^{n\times n}$, the objective term is $\Delta q^T \; W_{\Delta q} \; \Delta q$.

*3) Reach a given joint posture:* This term encourages the joints to move toward a desired vector of joint positions, $q_d$. A "passive" way to use this term is to set $q_d = (q_{max}+q_{min})/2$, which has the effect of biasing each joint toward the center of its workspace and away from joint limits. Alternatively, $q_d$ can be actively commanded; for example, if $q_d$ is set to the value of an IK solution and the other objective terms are given zero weight, this term basically turns the entire CAT controller into an IK controller that obeys constraints. Defining the weighting matrix as $W_q \in \mathbb{R}^{n\times n}$, the objective term is:

$$((q+\Delta q) - q_d)^T \; W_q \; ((q+\Delta q) - q_d) \tag{2}$$

The **QP constraints** are defined as follows:

*1) Obey joint limits:* Simply put, the incremental joint change may not push any joint past its position limits.

$$q_{\min} \leq (q+\Delta q) \leq q_{\max} \tag{3}$$

*2) Do not move in the direction of contact:* After each step, the collision detector checks for contact with the environment model, providing a point, $c_i \in \mathbb{R}^3$, and normal, $n_i \in \mathbb{R}^3$, for each contact. On the subsequent step this contact set is used to constrain the change in position of each contact point. The velocity Jacobian is computed for each contact point and is denoted $J_{c_i} \in \mathbb{R}^{3\times n}$. We assume that motion perpendicular to the contact normal is acceptable, but the point of contact may not move further into collision: $n_i^T(J_{\mathrm{c_i}}\Delta q) \geq 0$.

Additional objective terms or constraints are certainly possible. For example, while the contact constraints used in this paper can be viewed as an example of forbidden-region virtual fixtures [1], terms could be added to the objective to help a user follow a particular path or track a moving object.

### B. Step Validation and Collision Detection

Collision detection is a key component of the CAT, MP, and IK strategies.[2] Our system uses a commodity depth camera to build a continuously-updating volumetric representation of the world. Specifically, we use the Flexible Collision Library (FCL) to perform collision checks with an octree representation of the environment maintained using the Octomap library [9]. An example of a real-world scene and the corresponding Octomap is shown in Fig. 1.

In the CAT controller, collisions are computed after each QP step. The updated kinematic state is used to find the contact points and normals between the robot model and the environment model. Steps that have taken the robot into a colliding state can be accepted or rejected depending on the tolerance for predicted collisions. The contact information is used to formulate the constraints on the next QP step, which will prevent further motion into the contact constraint. We note that cases of desired contact, such as between the gripper and the environment, can be handled by disabling collision checking for links or objects that should not be constrained.

### C. Generating Suitable Motion Trajectories

The CAT, MP and IK methods each generate joint-space trajectories which are executed on the robot by a low-level 1kHz joint impedance controller. In general, the following elements complicate the relationship between responsiveness, speed, and safety in our teleoperation strategies:

*1) Velocity Limits:* The trajectory of joint motions generated via CAT (and the IK method) is checked for collisions. The number of steps is limited by computation time in each command period, so the robot has a finite horizon in which it can predict constraints. Hence, the robot must limit joint velocities such that each joint can come to a complete stop at the end of any given trajectory. Otherwise, this could lead to an inevitable collision state, where the robot is unable to stop in time to avoid obstacles that appear suddenly.

Since longer computed trajectories will result in higher allowable movement speed, we run the CAT and IK methods at 30 Hz (which is the update rate of the user command input) and compute as many steps as possible in the allotted time (approximately 25 in our implementation), as opposed to sending a single trajectory point at a time at a much faster rate ( 1kHz). This is a major difference from the MPC used in [10], which must compute a single point at a time and move slowly since the collision detection comes from force-contact sensors and cannot be predicted.

*2) Continuity:* The trajectory that was most recently sent to the joint trajectory follower will be partially executed by the time a new trajectory result is available. Hence, it is necessary to ensure continuity of commanded joint positions and velocities when splicing in the new trajectory.

Each strategy is responsible for returning a result before a time limit, $t_f$, at the end of each command period. The starting state, $q$, for the computation is picked to be the first point in the previous trajectory that was scheduled to be executed after $t_f$. Assuming the trajectory is computed in time, it can be spliced into the execution queue without a problem.

If a trajectory is not computed in time for any reason, the result must be discarded altogether. To prevent repeated failure due to insufficient time, we use an adaptive command period (similar to [7]), allowing more time if the previous computation failed and less time if there was time to spare. We note that this adaptive timing is most important for the MP strategy, as CAT and IK work incrementally and can almost always return a result before the time limit.

## V. Alternative Methods

We implemented three established approaches to arm teleoperation for comparison to our CAT method.

- **Collision-Aware Inverse Kinematics (IK).** In each command period (at 30 Hz), for the current end-effector goal pose, we look for a set of collision-free joint angles that achieve the goal. If successful, we create a trajectory to the goal using linear interpolation in joint-space. A collision check is performed for each step in the interpolation, ensuring the motion will avoid collisions. If the command period will be exceeded by checking the entire trajectory, a partial trajectory is used (as described in section IV-C).
- **Motion Planning (MP).** Sampling-based planning algorithms [4] are starting to approach the computational efficiency and responsiveness needed for arm teleoperation [8]. Our MP method uses RRT-Connect[3], a bi-directional sampling-based planner, to compute a trajectory to the most recent goal configuration (found using collision-free inverse kinematics). Motion planning has the advantage of operating in the global configuration space, and is able to find paths that take locally error-increasing steps in order to escape local minima. The downside is the additional computation time needed; the MP method was run at a nominal rate of 4 Hz (with adaptive timing as needed).
- **Jacobian-transpose Control (JT).** This method tracks the Cartesian goal using a standard torque control law of the form $\tau_p = J^T \mathbf{f}$, where $\mathbf{f}$ is a wrench computed from the error between the actual gripper pose and the goal pose at 1 kHz. New goal poses are sent to the controller at 30 Hz.

## VI. Experimental Results

We implemented, tested and compared all the teleoperation methods described above on a variety of scenes and trajectories. The hardware consisted of a PR2 robot, equipped with a 7-DOF arm. For environment sensing we used a Kinect sensor mounted on the robot's head.

For each scene, we used one of the two input devices described in Sec. III (mouse and Razer Hydra) to record a trajectory of end-effector goals over time. Starting from the

[2]Collision detection is by *far* the most expensive part of the trajectory computation process, on the order of 1 ms per check in our implementation.

[3]Recent benchmarks [2] show that RRT-Connect works well in typical manipulation settings. We found that other bi-directional planners (e.g., SBL) produce similar results, while single-tree planners significantly increase planning times. We used the planner implementations available in OMPL [4].

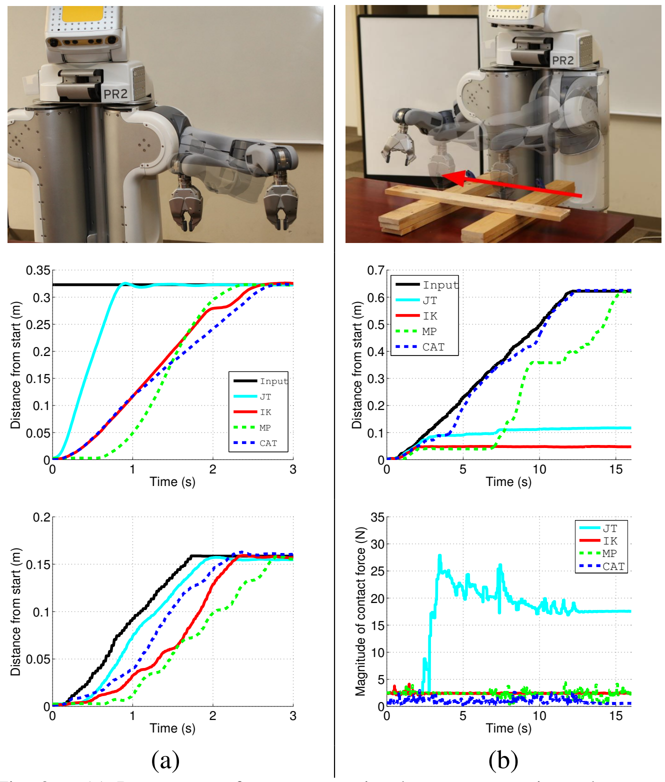



Fig. 3. (a) Response to free-space motion between a starting robot pose (transparent) and the goal end-effector pose (opaque). The top plot shows tracking results for a step input, showing the distance from the starting pose over time, for both the goal pose and the end-effector pose achieved by each teleoperation method. The bottom plot shows results for a continuous input. (b) Motion across a set of rigid boards 9 cm wide, 6 cm high, and 24 cm apart, in response to a *continuous input* translation.

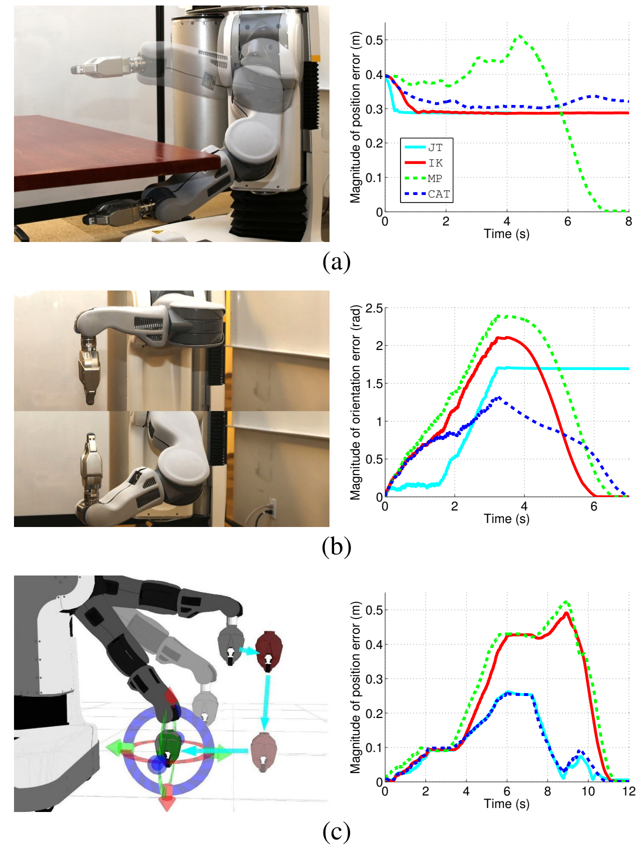



Fig. 4. Additional constrained teleoperation examples. a) Collision constraint perpendicular to straight line motion to goal. b) Straight path to goal hits joint limit (starting with gripper pointing down and ending with gripper pointing up). c) Part of the goal trajectory is outside the robot's workspace. In this case, end-effector goals outside the workspace are shown as disembodied grippers; closest reachable gripper pose for the goal in the bottom right is shown as a transparent robot posture. (Plot colors are consistent across all figures.)

same initial robot pose, the recorded goal trajectory was then played back for each of our teleoperation methods, and the resultant arm motion was recorded. In addition, we used a 6 DOF force/torque sensor mounted in the wrist to record end-effector forces from contacts with the environment.

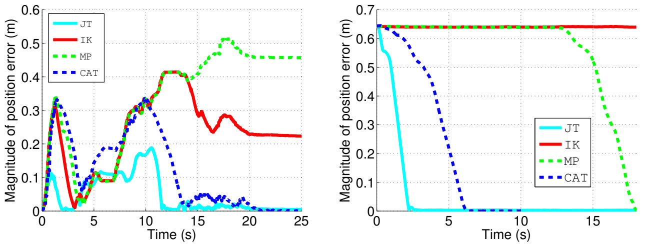
The first scene (fig. 3a) did not contain any obstacles. While the focus of our methods is efficient operation in cluttered settings, numerous tasks will contain at least parts of the trajectories where no obstacles play a role, and good tracking in these region is important for overall efficiency. We present two cases: a *step input* where the goal pose is directly set to the desired final pose of the gripper; and a *continuous input* where the goal is moved from the starting location of the gripper to its desired final pose over the course of approximately 1.5s. We believe the second case to be more representative of real-life teleoperation, where the operator continuously updates the goal and moves it towards the desired location.

We notice that all four methods track the desired pose. We focus in more detail on two metrics: response time (time after a new goal is specified until the arm starts to move) and total time (needed to reach the goal). As expected, JT achieves the lowest times for both metrics, while MP is the slowest. In the step input case, CAT shows lower response time but higher total time than MP; in the continuous input case, it achieves lower times on both metrics, ranking immediately behind JT.

Fig. 3b looks at a case where the direct path to the specified goal is blocked by obstacles (in this case, boards clamped to the table). As seen in the tracking results, the JT method gets physically stuck on the boards due to friction, and the IK method detects the impending collision along the path and stops; neither method reaches the final goal. MP is able to plan a path around the obstacles, though it has some trouble following the continuously moving goal as it passes through infeasible locations. CAT performs the best in this scenario as it is able to follow the goal quite closely while "sliding" up and over the virtual constraints without actually touching the boards (as shown in the force data).

We note that in the scene of Fig. 3b the obstacles do not create a local minimum. However, in a case where the straight path to the goal is perpendicular to the obstacle surface (Fig. 4a), local methods such as JT and CAT are unable to escape. MP is able to plan a path that temporarily increases tracking error in order to eventually reach the goal.

Fig. 4b demonstrates how joint limit constraints affect the presented methods. In this common scenario, achieving the goal pose is not possible using just wrist DOFs, as it requires also rotating the forearm. By tracking the straight path to the goal, JT gets stuck against the wrist joint limit with the gripper




(a) (b)

Fig. 5. a) Tracking results for reaching into a shelf with a collision-free goal trajectory. b) Results for exiting the shelf with a colliding goal trajectory.

in a horizontal pose. In contrast, CAT, MP, and IK are able to keep the arm away from joint limits and reach the goal (the final pose as achieved by MP is shown in the image). Fig. 4c shows the case where part of the goal trajectory is outside the robot's workspace. Here we notice that MP and IK simply stop tracking while the goal is unreachable (from 4 seconds to approximately 8.5 seconds into the trajectory). In contrast, JT and CAT track the goal along the edge of the workspace.

To show off more general, overall behavior in a typical cluttered scene, Fig. 5 plots tracking results for reaching into and retreating out of the cluttered shelf shown in Fig. 1. The goal trajectory for the reach in was demonstrated with a Razer Hydra, with the operator being careful to select goal poses that were not colliding. Tracking error is shown in Fig. 5a. In this case, JT performs best because the goal poses happen to be collision-free. CAT follows close behind, with the larger tracking error due to joint limit avoidance. IK gets stuck because of momentarily infeasible goals. MP follows the initial part of the trajectory, then stops, and eventually moves to the goal (the plot is truncated to better show the behavior of the other methods). Fig. 5b shows results for the retreat, in which the goal trajectory collides with the shelf. JT again gets there fastest, but hits the shelf with high force (like in Fig. 3b) on the way. CAT follows close behind without hitting the shelf, IK gets stuck, and MP eventually gets there.

## VII. Discussion and Conclusions

In this paper, we focused on the problem of assisted teleoperation in unstructured environments based on visual sensing. We introduced CAT, a constraint-aware teleoperation controller based on sequential quadratic programming, which continuously attempts to reduce the tracking error while taking into account constraints such as collisions or joint limits.

Manipulation in cluttered environments is by nature a highly-constrained task. In our experiments, we used a real robot and real sensor data collected during task execution to explore a number of these constraints. Our analysis included the CAT controller, as well as three additional methods: global motion planning with a continuously-updating goal pose (MP), collision-aware inverse kinematics with joint-space interpolation (IK), and Jacobian-transpose control (JT).

Our results show that each of the teleoperation methods has desirable characteristics in some situations. The JT approach is the most responsive and fastest to reach the goal in less-constrained settings. Unfortunately, it lacks collision-avoidance assistance, and thus is prone to undesirable collisions. The IK approach is also very responsive in less-constrained settings, avoids colliding with itself and with the environment, and is better than JT at dealing with joint limit constraints. However, it easily gets stuck when obstacles are in the way or goals are infeasible. Since the MP method is based on global motion planning, it is the only one that will reach the goal when the others get stuck in local minima. However, it is also the least responsive of those tested, and current implementations also lack the ability to provide a "best effort" approach in cases where the goal pose is unreachable.

Based on the presented results, we believe that the CAT controller provides the best balance in terms of our stated goals: it can track constrained end-effector goal poses with a fast response, including providing "best effort" approaches to infeasible goals, avoid undesired collisions with itself and with the environment, and deal with joint limit constraints. With an operator in the loop to provide high-level motions that can take the robot around major obstacles that would cause the CAT controller to get stuck in local minima, we believe that the CAT controller is the overall winner in most situations.

However, our results also suggest that complete manipulation tasks could benefit from the strengths of each of these methods. An interesting approach is to allow switching among the methods based on context. However, asking the operator to actively select situationally-appropriate teleoperation methods requires a higher level of expertise (and can introduce delay), while automated switching implies task-level context awareness on the part of the robot. We plan to investigate these paths as part of our future studies.

## References

[1] J. Abbott, P. Marayong, and A. Okamura. Haptic virtual fixtures for robot-assisted manipulation. *Robotics Research*, pages 49–64, 2007.

[2] B. Cohen, I. Şucan, and S. Chitta. A generic infrastructure for benchmarking motion planners. In *IROS*, 2012.

[3] J.E. Colgate, M.C. Stanley, and J.M. Brown. Issues in the haptic display of tool use. In *IROS*, 1995.

[4] I. Şucan, M. Moll, and L. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, December 2012.

[5] Anca D Dragan and Siddhartha S Srinivasa. Formalizing Assistive Teleoperation. In *RSS*, 2012.

[6] S. Haddadin, S. Belder, and A. Albu-Schäffer. Dynamic motion planning for robots in partially unknown environments. In *IFAC*, 2011.

[7] Kris Hauser. On responsiveness, safety, and completeness in real-time motion planning. *Autonomous Robots*, 32(1): 35–48, September 2011.

[8] Kris Hauser. Recognition, Prediction, and Planning for Assisted Teleoperation of Freeform Tasks. In *RSS*, 2012.

[9] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An Efficient

Probabilistic 3D Mapping Framework Based on Octrees. *Autonomous Robots*, 2013.

[10] A. Jain, M. D. Killpack, A. Edsinger, and C. C. Kemp. Reaching in clutter with whole-arm tactile sensing. *IJRR*, 32(4):458–482, March 2013. ISSN 0278-3649.

[11] O. Khatib. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *IJRR*, 5(1):90–98, March 1986.

[12] R. Knepper, S. Srinivasa, and M. Mason. Hierarchical planning architectures for mobile manipulation tasks in indoor environments. In *ICRA*, 2010.

[13] Roberto Lampariello, Duy Nguyen-Tuong, Claudio Castellini, Gerd Hirzinger, and Jan Peters. Trajectory planning for optimal robot catching in real-time. In *ICRA*, 2011.

[14] P. Mitra and G. Niemeyer. Haptic Simulation of Manipulator Collisions Using Dynamic Proxies. *Presence: Teleoperators and Virtual Environments*, 16, 2007.

[15] P. Mitra and G. Niemeyer. Model-mediated telemanipulation. *IJRR*, 27(2):253–262, 2008.

[16] C. Passenberg, A. Peer, and M. Buss. A survey of environment-, operator-, and task-adapted controllers for teleoperation systems. *Mechatronics*, 20(7), 2010.

[17] QC Pham and Yoshihiko Nakamura. Affine trajectory deformation for redundant manipulators. In *Robotics: Science and Systems*, 2012.

[18] Michael Posa and Russ Tedrake. Direct Trajectory Optimization of Rigid Body Dynamical Systems through Contact. *Algorithmic Foundations of Robotics X*, 86:527–542, 2013.

[19] L. Sentis and O. Khatib. Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *Intl. J. of Humanoid Robotics*, 2(04):505–518, 2005.

[20] Alexander Werner, Roberto Lampariello, and Christian Ott. Optimization-based generation and experimental validation of optimal walking trajectories for biped robots. In *IROS*, October 2012.