Cartesian Motion Planning & Task Programming with CRAFTSMAN

Patrick Beeson
TRACLabs Inc.
pbeeson@traclabs.com

Stephen Hart
TRACLabs Inc.
swhart@traclabs.com

Seth Gee
TRACLabs Inc.
seth@traclabs.com

I. Introduction

Deploying multi-degree-of-freedom robot systems in real-world environments will require motion and task planning software for manipulation that is robust, flexible, and easy to use by non-PhD users. While many robot manufacturers provide custom, proprietary solutions for their systems, there is no industry standard that can be used on different platforms, can quickly adjust to new tasks or environments, or can be used effectively by both robotics researchers and trained experts. While open-source community efforts such as KDL [7] or MoveIt! [8] have been used in the ROS ecosystem [6] to provide generic Inverse Kinematics and pick-and-place functionality, PhD-level robotics expertise is often required to integrate these software components into specific fielded applications. Moreover, these software components are designed to individually provide reliable, generic functionality, but typically have narrowly defined APIs that do not take full advantage of the capabilities of other components when integrated into a larger robotic software architecture.

To address this community deficiency, TRACLabs is developing a software suite called CRAFTSMAN (CaRtesian-based AFfordance Template Suite for MANipulation) designed to make robot task and motion planning for object manipulation and tool-use practical in real-world applications. CRAFTSMAN currently consists of integrated open-source libraries for

- Inverse Kinematics, https://bitbucket.org/traclabs/trac_ik
- Cartesian Motion Planning, https://bitbucket.org/traclabs/trac_ik
- Graphical Teleoperation, https://bitbucket.org/traclabs/robot_interaction_tools
- Task Programming. https://bitbucket.org/traclabs/affordance_templates

These libraries, taken together, provide state-of-the-art capabilities for robot manipulators and have the potential to create a new standard by which users can operate their robots or developers can base their applications. Each of these four libraries is briefly described below.

II. Overview of the current CRAFTSMAN system

The CRAFTSMAN software components have been designed to work together in order to provide advanced manipulation capabilities to robot applications. The current software implementations utilize libraries from the ROS ecosystem, including the popular inter-process messaging and 3D visualization tools. The unified system architecture is shown in Figure 1.

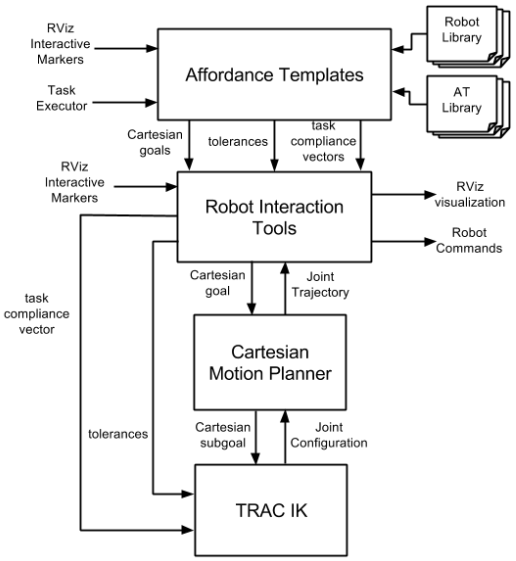


Fig. 1. The CRAFTSMAN system architecture.

Given a Cartesian end effector goal, goal tolerance settings, and task compliance (or conditioning) requirements, the **TRAC-IK** module provides inverse kinematic solutions to a higher-level **Cartesian Motion Planner**. The **Robot Interaction Tools** module allows applications to specify Cartesian goals and requirements—either by a teleoperator (through the RViz 3D interaction environment [4]) or by the **Affordance Templates** layer—and executes the resulting plans on the robot hardware. Affordance Templates exist as robot-agnostic data structures that provide an advanced level of flexibility in task programming and human operability appropriate for both full and "adjustable" autonomy [3].

A. The TRAC-IK Inverse Kinematics Library

TRAC-IK is an open-source drop in replacement for KDL's `CartToJnt()` API that overcomes many of the issues users of joint-limited manipulators encounter [1]. By running two types of IK solvers concurrently, TRAC-IK can use inverse Jacobian derived answers when they converge quickly, or sequential quadratic programming (SQP) derived answers when inverse Jacobian methods are slow to converge. It has many fewer false negative failures and is faster than KDL's inverse Jacobian IK solver (see Figure 2).

Chain	DOFs	Orocos' *KDL* solve rate	Orocos' *KDL* Avg Time	*TRAC-IK* solve rate	*TRAC-IK* Avg Time
Atlas 2013 arm	6	**75.54%**	1.35ms	**99.97%**	0.33ms
Atlas 2015 arm	7	**75.71%**	1.50ms	**99.18%**	0.48ms
Baxter arm	7	**61.07%**	2.21ms	**99.17%**	0.60ms
Denso VS-068	6	**27.92%**	3.69ms	**99.78%**	0.38ms
Fanuc M-430iA/2F	5	**21.07%**	3.99ms	**99.16%**	0.58ms
Fetch arm	7	**92.49%**	0.73ms	**99.96%**	0.44ms
Jaco2	6	**26.23%**	3.79ms	**99.51%**	0.58ms
KUKA LBR iiwa 14 R820	7	**37.71%**	3.37ms	**99.63%**	0.56ms
KUKA LWR 4+	7	**67.80%**	1.88ms	**99.95%**	0.38ms
PR2 arm	7	**83.14%**	1.37ms	**99.84%**	0.59ms
NASA Robonaut2 'grasping leg'	7	**61.27%**	2.29ms	**99.31%**	0.67ms
NASA Robonaut2 'leg' + waist + arm	15	**97.99%**	0.80ms	**99.86%**	0.79ms
NASA Robonaut2 arm	7	**86.28%**	1.02ms	**99.25%**	0.50ms
NASA Robosimian arm	7	**61.74%**	2.44ms	**99.93%**	0.44ms
TRACLabs modular arm	7	**79.11%**	1.35ms	**99.80%**	0.53ms
UR10	6	**36.16%**	3.29ms	**99.47%**	0.49ms
UR5	6	**35.88%**	3.30ms	**99.55%**	0.42ms
NASA Valkyrie arm	7	**45.18%**	3.01ms	**99.63%**	0.61ms

Fig. 2. TRAC-IK versus KDL's Inverse Jacobian IK. Results are averages from 10,000 trials requesting IK solutions for random, yet reachable, Cartesian poses from a "nominal" configuration.

To enhance planning and task performance, TRAC-IK can sort the multiple solutions it finds by a user-requested secondary metrics. Currently available secondary metrics include path-shaping criteria, such as *Minimize Distance* (from the initial seed), or kinematic conditioning metrics such as *Maximize Manipulability* (two commonly used manipulability metrics are currently provided) [9]. Future additions will include task compatibility metrics that will find solutions that condition the robot to apply forces or movements effectively in certain directions as defined by the task being performed [2]. TRAC-IK also ensures that solutions do not place the robot in collision with itself. Though currently only self collisions are implemented, ongoing work is being performed to extend TRAC-IK to provide solutions that are also not in collision with perceived environmental obstacles.

B. The CRAFTSMAN Cartesian Planner

The Cartesian planning layer takes as input end effector goal poses, tolerances on those poses, and secondary objectives and iteratively calls IK in order to provide a guaranteed Cartesian motion through the workspace. In addition to the inputs above, this planner can accept tool offsets so that Cartesian planning does not simply have to be with respect to the end effector, but can be to any arbitrary pose offset from the end effector frame of reference. In this way a tool, like a drill tip, can perform smooth straight-line Cartesian motion (maybe with tolerances that allow roll to be unbounded), even if the robot's hand is moving along a complicated trajectory. Similarly, if the robot is grasping a wheel, the center of the wheel can be used as the tool offset, with a 1-DOF Cartesian roll goal results in a plan that moves the robot's hand in a smooth arc to turn the wheel.

Because the IK solver is guaranteed to return collision free results, any Cartesian trajectory is by definition collision free. Currently, if a requested Cartesian path cannot avoid collisions, no plan is returned; thus, the current implementation is more restrictive than stochastic planners, but yields more reliable plans for object manipulation and tool usage. Additionally, unlike most other Cartesian trajectory generators, the current solution does not try to enforce Cartesian motion only via interpolated waypoints in Cartesian space. To ensure that the end effector (or tool tip) cannot deviate from the smooth Cartesian trajectory—even between two nearby waypoints—a time-based filter is used to detect potential joint motions that would take longer than straight-line Cartesian motion between waypoints. If such a deviation occurs, alternative IK solutions for the trajectory are considered. Future work will investigate scenarios where the complete failure to find Cartesian trajectories falls back to using near-Cartesian motion or even joint-level plans if the task allows.

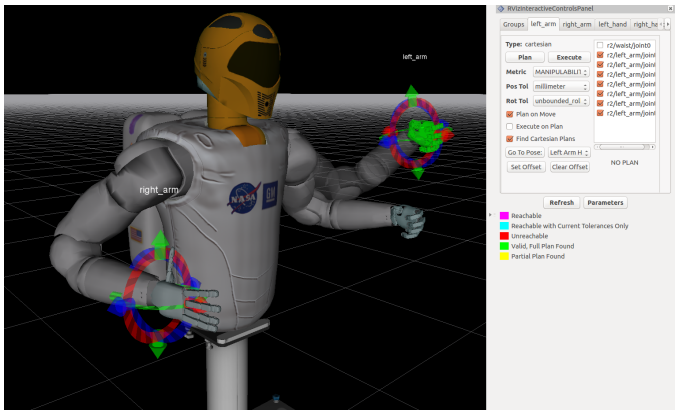


Fig. 3. The RViz *interactive controls* node and control panel.

C. Robot Interaction Tools for Teleoperation

The Robot Interaction Tools layer provides a number capabilities to bridge the gap between the CRAFTSMAN Cartesian motion planner and the application layer. This layer, upon request to compute motion plans to goals provides visualization capabilities for animating these plans in RViz. This is shown in Figure 3 where the end point of the plan can be seen as a transparent "ghost" of the robot overlaid next to the (solid) visualization of the robot in its current configuration. Also apparent in the figure are the 6-DOF RViz interactive markers that the operator can use to move the end effectors. Within this 6-DOF marker, a virtual representation of the robot's end effector is displayed, colored according to feedback from the Cartesian motion planner. For example, if a plan to that location is found, the end effector is colored green. If a plan is found that is only possible within the specified tolerance bounds, the end effector is colored blue. The legend specifying the meaning of colors is shown in the panel. An *interactive controls* panel is also provided in the CRAFTSMAN system to allow a teleoperator to specify certain parameters such as the secondary criteria, tolerance bounds, or a joint mask (to force certain joints to stay at their current position during robot movement; applicable if the robot has kinematic redundancy with respect to the requested path goals).

A demonstration of the interactive controls node can be seen at https://youtu.be/COpq03PHdi8.

Interactive controls for any manipulator can be added from the RViz panel with an option for whether it is a CARTESIAN, JOINT, or END EFFECTOR group. If the controls are specified as CARTESIAN, a 6-DOF marker will be made available at the

tip link of that group, such as can be seen on the left and right hands of Robonaut 2 in Figure 3. If the controls are specified as JOINT or END EFFECTOR interactive controls will be made available that allow the operator to click on corresponding links (such as the robot's head or palm) to expose a context menu that allow the operator to choose stored configurations (e.g., "hand close", "hand open", etc.). JOINT groups allow the operator to visualize plans before execution, whereas END EFFECTOR groups will directly command the robot.

D. Affordance Templates for Task Programming

The CRAFTSMAN software integrates the *affordance template* framework for task programming and operation [5]. Affordance templates allow a user to define manipulation behaviors via a collection of sparse end-effector waypoints with respect to an object or tool's frame of reference. This can happen offline, prior to running the robot. At runtime, the template is matched to perceptual data to seed a fine-grained obstacle-avoidance trajectory generation and control system.

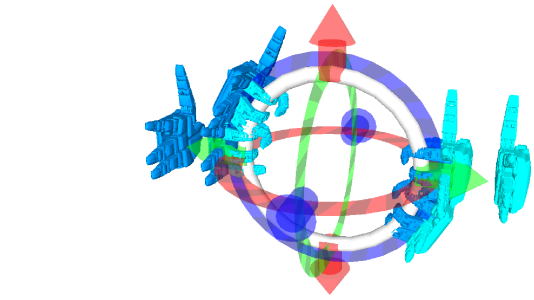
Affordance templates provide a common task representation for defining manipulation tasks that supports full autonomy when possible, but can "share" that autonomy with a human operator by providing interactive controls to an operator that verifies and/or tweaks object perception or manipulation plans prior to robot execution. Affordance template strategies are defined in terms of end effector waypoint sequences (represented in object-centric coordinate frames) and pose configurations. For example, a simple pick and place template would specify pre-grasp, pick, place, and release waypoint goals (and end effector configurations) in pick-object and place-goal task frames. An example of a wheel-turning affordance template defined for the NASA Valkyrie robot is shown in Figure 4.

Affordance templates are robot and environment configurable, providing a useful application framework that is general and applicable to many different task contexts. Each template is defined in task coordinate frames independent of any particular robot, so if an operator wishes to command a new robot in the framework, all that that operator needs to do is create a single configuration file that maps a few robot properties (like end effector coordinate frames) to template constants. In the CRAFTSMAN system, affordance templates allow additional features to be specified at each Cartesian waypoint that exploit the full power of the Cartesian motion planner. Specifically, each waypoint can specify a tool offset, arbitrary Cartesian tolerance bounds, or kinematic or task conditioning criteria. These features allow tasks to be defined as more then just a series of spatial Cartesian end effector locations, and enable the task programmer to incorporate particulars about the task.

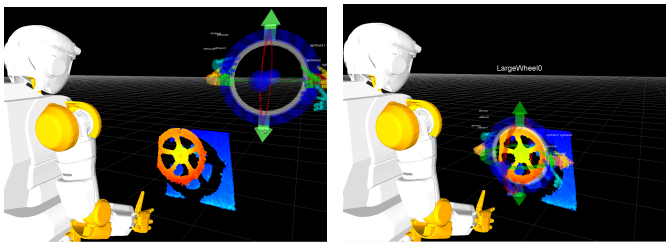
A demonstration of affordance templates can be seen at https://youtu.be/wf5a-HLw8Zk.

III. Ongoing Work

Ongoing work is investigating further capabilities for the CRAFTSMAN system. These capabilities will require further development of the Cartesian motion planner and affordance template framework and will ultimately allow task programs



(a)



(b) (c)

Fig. 4. (a) A wheel-turning affordance template instantiated for Valkyrie. The wheel is shown in white and can be oriented in the 3D environment via the 6-DOF controls. A two handed strategy for turning the wheel is shown as a sequence of end effector waypoint visualizations. (b) The view of the robot's 3D sensor data in which the valve is clearly recognizable, along with the robot avatar. (c) An operator can use the 3D controls to resize and register the wheel template to the sensor data using the interactive arrows. Right-click menus allow adjustment of the individual waypoint parameters.

to incorporate additional features for improved performance. These features include how to specify more reactive strategies that keep the robot's end effector in appropriate locations with respect to objects, even if those objects move, or to maintain conditions, such as keeping a grasp on that object. Other dynamic properties such as compliance or force/torque application are also being considered to enable potentially complex, non-spatial tasks.

References

[1] P. Beeson and B. Ames. TRAC-IK: An open-source library for improved solving of generic inverse kinematics. In *Proceedings of the IEEE RAS Humanoids Conference*, 2015.

[2] S. Chiu. Control of redundant manipulators for task compatability. In *International IEEE Conference on Robotics and Automation (ICRA)*, 1987.

[3] J. W. Crandall and M. A. Goodrich. Experiments in adjustable autonomy. In *IEEE International Conference on Systems, Man, and Cybernetics*, 2001.

[4] D. Gossow, A. Leeper, D. Hershberger, and M. Ciocarlie. Interactive Markers: 3-D User Interfaces for ROS Applications. *IEEE Robotics & Automation Magazine*, 18(4):14–15, 2011.

[5] S. Hart, P. Dinh, and K. Hambuchen. The Affordance Template ROS Package for Robot Task Programming. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[6] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. ROS: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[7] R. Smits. KDL: Kinematics and Dynamics Library. http://www.orocos.org/kdl.

[8] I. A. Sucan and S. Chitta. MoveIt! http://moveit.ros.org.

[9] M. Tsai. *Workspace Geometric Characterization and Manipulability of Industrial Robots*. UMI, 1986.