Prospects for Combining Task and Motion Planning for Bi-Manual Solution of the Rubik's Cube

Boling Yang, Patrick Lancaster, and Joshua R. Smith

***Abstract*—We first describe a robot that solves the Rubik's cube using separate task and motion planning with sensor-enabled closed-loop execution. The task solver uses iterative deepening A-star to plan the sequence of cube moves. Then a special-purpose grasp planner and controller places the manipulator hands on the cube to execute the task plan. The task requires two hands and two-handed re-grasping maneuvers.**

We then discuss the potential benefits of combining the task and motion planning, and discuss potential approaches to making this combination. Finally, we consider the prospects for combining task planning, motion planning, and sensor-based closed loop control.

I. Introduction

The Rubik's cube is a puzzle that requires a long sequence of manipulation operations to solve. There is a discrete set of required manipulation operations, but it is a fairly rich set of operations. Changing the cube state requires two hands (one to constrain part of the cube, and one to rotate the remainder of the cube). Also, solving the cube requires re-grasping, and (while it may not be necessary) it is very natural to use bi-manual re-grasping.

Our initial motivation for the robotic solution of the Rubik's cube was to explore the use of pre-touch sensing during the course of long manipulation sequences to correct actuation errors as they arise. Indeed we found that without pre-touch sensing [3] , the PR2 typically makes a catastrophic error after an average of 8 moves; with pre-touch sensing, an indefinite sequence of moves is possible. We developed a system that demonstrates the end-to-end functionality using a task-based planner to choose a sequence of cube moves, and a single-purpose planner to choose bi-manual hand actions to execute the cube plan. In our initial implementation, these planning layers are completely separate: there is no way for hand constraints to affect the planned cube transition sequence. A demonstration of the system can be viewed here: https://www.youtube.com/watch?v=kzzbPFHNmPM

Now that the end-to-end system works, we aim to increase its speed by jointly planning the task and motion sequences. For example, we hypothesize that by re-ordering certain task sequences (cube transitions), it should be possible to, for example, eliminate some slow re-grasping operations, or more generally choose hand actions that are faster.

II. Initial Approach: Separate Task and Motion Planning

The initial system enables a PR2 robot to solve the Rubik's cube from any initial configuration. A head-mounted camera is used to detect the initial color state of the cube. Note that the only modification to the robot is the addition of the pre-touch sensors; no modifications were made to physically constrain the cube to a certain position within the robot's gripper. The goal of the puzzle is to rotate the faces of the cube so that the nine squares of each face are all of the same color. Our current working approach used separate modules for task and motion planning, but its limitations have inspired us to integrate them together, as discussed in the next section. The robot picks up the cube and inspects the faces using the Kinect camera to estimate the initial cube condition (i.e the color of each square). This visual inspection operation requires several re-grasps and changes in cube orientation.

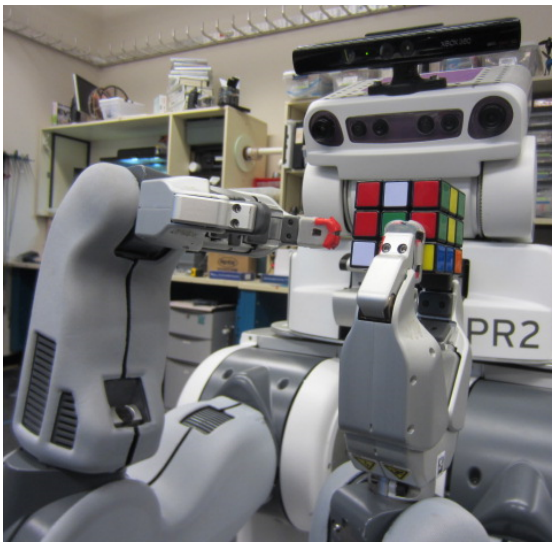

Fig. 1: By combining task planning with pre-touch sensor aided motion planning, the robot is able to solve the Rubik's cube

Task Planning Given the initial state of the Rubik's cube, the robot generates a solution using Kociemba's Two Phase algorithm [1][4]. To solve the Rubik's cube, the algorithm first uses an iterative deepening $A*$ algorithm to search for maneuvers that transform the scrambled cube to a state that is an element of a particular group G with special properties that correspond to a partial solution with corners and edges in the right locations but possibly with incorrect orientation. In phase 2, the 8 corners and all edges will be set to the correct orientations. The algorithm searches to find a solution that requires a minimal number of cube face rotations. It is known that any cube can be solved with at most 20 moves.[4]

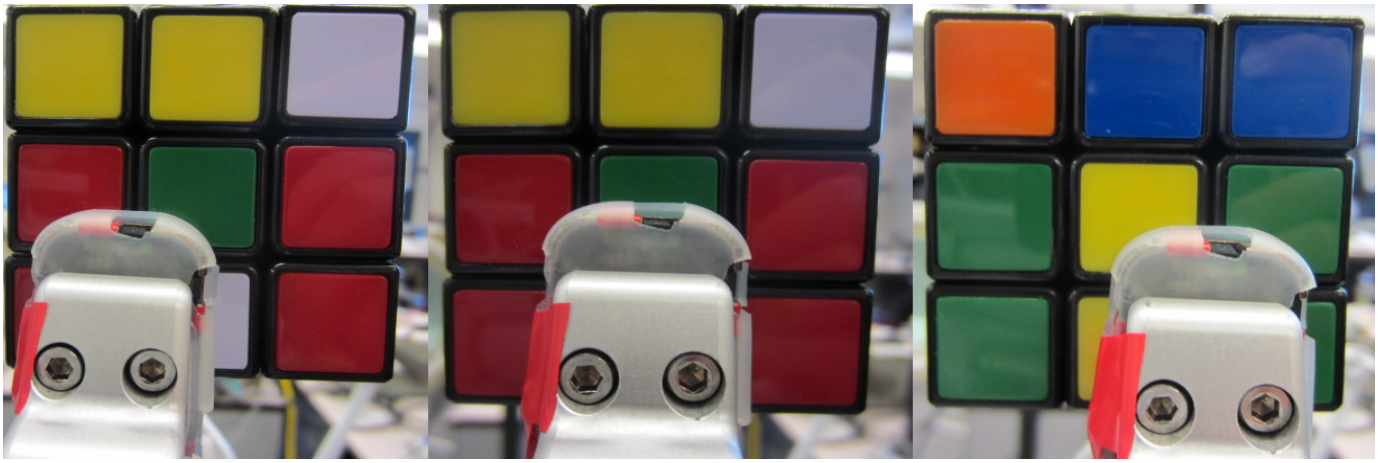

Fig. 2: Three grasp points used by the current system. These correspond to the grasp points labeled 3, 5, and 7 in Fig. 3.

Motion Planning The solution generated by the task planner is a sequence of face rotations that will bring the cube to the solved state. In our initial system, we use a simple state-machine motion planner to cause the grippers to execute the task plan.

The motion planner maps each cube face rotation to an appropriate two-arm trajectory. Pre-touch sensing has been integrated into these motions so that the robot can adjust prior to making contact with the cube. The combination of these motions can successfully execute any face rotation sequence provided by the task planner.

The space of the general grasp planning problem is surprisingly large. There are up to 120 feasible grasp points for one hand on a cube: each hand can be placed in one of 6 headings, and in one of two wrist roll states, which yields 6x2 = 12 hand approach orientations. For each of these, there are 10 valid grasp points, as shown in Fig. 3. This yields a total of 12x10 = 120 single handed grasps. When the second hand is considered, there are no more than 5x2x10 = 100 grasp points. (The second hand must have a different approach orientation than the first hand, so there are 5x2 rather than 6x2 hand approach orientations.) Thus the total number of two handed grasps is no more than 120x100 = 12,000. (In actuality, kinematic constraints and additional hand collisions reduce the set of feasible two-handed grasps further.)

Most generally, one can imagine wanting to transition from any one of the initial two-handed grasp states to any other two-handed grasp state. Making such a transition may require a motion plan such as: open gripper 1, re-position it, close gripper 1 at a new location, open gripper 2, reposition it, close gripper 2.

In our initial working system, we introduce a number of constraints to simplify the motion planning. We use one gripper to fix the Rubik's cube while the other rotates one of the faces. In some cases, the cube must be transferred from one gripper to another in order to rotate a certain face. Re-grasping can also be necessary in order to shift the grip position with respect to the cube. Two 'home' poses were defined to allow the robot to return to a kinematically unconstrained position before attempting each rotation. All the arm motions and manipulation on the cube comply with the following constraints:

1) After performing a cube face rotation, the grippers return to the home pose.
2) During a cube rotation, one of the grippers holds two rows of the cube stationary and the other rotates one face of the cube. The cube holding gripper always constrains two layers of the cube and the manipulating gripper rotates only one layer. Each finger of the holding gripper has three feasible grasp points as shown in Fig. 2. There is only one allowed grasp point for the rotating gripper: the mid-point of the face, which allows the face to rotate with a very simple wrist rotation.
3) The cube's orientation is kept consistent with respect to the holding gripper.
4) The front, back, and right faces can only be rotated when the left gripper is holding the cube. The up, down, and left faces can only be rotated when the right gripper is holding the cube.

These constraints effectively simplify the motion planning problem enough that it can be handled by a finite state machine. The current system only considers a subset of the grasp points shown in Fig. 3. As a result, it disregards many possible motion plans that may require less execution time. For example, by allowing a larger set of grasp points, the first constraint previously listed may no longer be necessary. Furthermore, although the task planner is nearly optimal in the context of minimizing the number of required rotations, not all rotations are equivalently costly in terms of execution time, since some may require more motions than others. All of these limitations make the current system highly inefficient. Executing 20 cube rotations requires approximately 34 motions over a time period of 7 to 8 minutes. A planner that considers both robot motion and cube solving simultaneously will generate motion plans that are optimal in manipulation speed with more flexible grasping and re-grasping strategies.

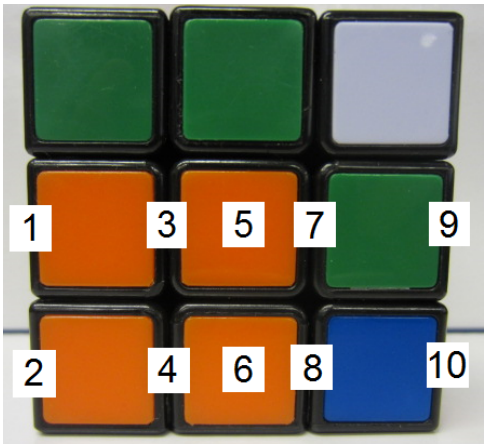

Fig. 3: The proposed system will use ten unique grasp points. Assuming the gripper is approaching from the bottom of the cube, each white box represents a possible grasp point.

III. Proposed Joint Task-Motion Planning

While our initial work focused on planning in the state space of the cube alone, our future work will reformulate the Rubik's cube problem as a search through a larger state space that is the product of cube states and the relevant robot states. By defining such a space, the robot will be able to more effectively assess the true cost of proposed solution sequences. We believe that such a formulation will result in faster solutions.

The more general state space we are contemplating should capture the interaction between the robot's manipulators and the object of interest. The original Rubik's cube state will be augmented with variables describing the state of each gripper. For each gripper, these variables include

1) The position of the gripper with respect to the cube. When the robot positions either of its grippers prior to making contact with the cube, there is a discrete set of grasp points that it can choose. Along each parallel pair of edges of the cube, there are ten unique grasp points, as shown in Fig. 3.
2) The orientation of the gripper. The gripper's orientation consists of a heading (north, east, south, west, up, down) and a roll that rotates the gripper in 90 degree increments. This yields a total of 12 unique orientation configurations, although some may be invalid depending on which gripper is of interest. The gripper orientation determines which face and which pair of sides along that face the gripper will make contact with. While these discrete gripper poses are sufficient for achieving all possible Rubik's cube manipulations and will keep the state space relatively small, this particular discretization (or any other) is not absolutely required.
3) Whether the gripper is open or closed.

We are evaluating the feasibility of performing joint task-motion planning by searching this augmented robot-cube state space. The set of valid transitions can be viewed as the edges in a bi-partite graph, where initial states are represented by nodes on the left and final states are represented by nodes on the right.If we start with an initially empty set of valid transitions, and add only transitions that are known to be valid, then we will never make illegal moves (such as ones that would put the grippers in collision with one another).

Given the relatively simple 'physics' of the Rubik's cube and the robot hands, it might be possible to generate the table of valid transitions by 'compiling' a small set of rules (that models the physics of the system) into this explicit state transition table. Assuming that we can build such a table, an open question is whether it will be feasible to search the resulting state graph. If the state space is too large, we will need to consider more sophisticated planning strategies.

More generally, a full motion planner that takes in to account collisions between the robot's grippers could be used to correctly generate the matrix of valid state transitions. Note however that even if valid transitions are missed, the system is still likely to solve the problem, as our initial system shows. Missing potentially valid state transitions simply means that the robot will miss certain short cuts that it could have taken.

Principles from our proposed work could be extended to other tasks as well. As mentioned previously, we can grow an initially empty set of state transitions by validating candidates with an offline motion planner. This results in full integration of motion and task planning: motion planning constrains the types of task plans that can be generated through state transition validation, and task planning specifies which types of motion plans are needed when the task is actually attempted. We are particularly interested in how this idea can be applied to sequential tasks that require precision, such as constructing objects. Knepper [2] generate and execute plans to build IKEA furniture using multiple mobile robots. Their high level planner only considers the physical constraints of the objects when formulating a plan. We believe that by augmenting the object states with the relevant robot state, more effective plans can be produced. Furthermore, as demonstrated in our current work, precise and error-prone motions, such as screwing in a table leg in [2], can be aided by positional feedback provided by pre-touch sensing.

References

[1] pr2 rubiks solver, 2011. URL https://github.com/uu-isrc-robotics/pr2_rubiks_solver.

[2] Ross A Knepper, Todd Layton, John Romanishin, and Daniela Rus. Ikeabot: An autonomous multi-robot coordinated furniture assembly system. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 855–862. IEEE, 2013.

[3] Brian Mayton, Louis LeGrand, and Joshua R Smith. An electric field pretouch system for grasping and co-manipulation. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 831–838. IEEE, 2010.

[4] Tomas Rokicki, Herbert Kociemba, Morley Davidson, and John Dethridge. The diameter of the Rubik's cube group is twenty. *SIAM Review*, 56(4):645–670, 2014.