Constrained Tool Planning in Work Space Using Inverse Jacobian Control

Adam Cantor, Jonathan Martin, Leo Keselman, Stewart Butler

***Abstract*—This report details an approach to known object interaction with a Schunk Arm. A motion planning theorem is applied with a focus on workspace control. The implementation of this system was done in the GRIP/DART simulation environment designed/used by the Golems Lab at Georgia Tech.**

I. Introduction

The purpose of this work is to create a methodology for interacting with known objects in a loosely defined workspace. This goal is directly in line with the DARPA robotics challenge to apply humanoid robots to search and rescue applications by using tools and exploration algorithms.

No single robot can be equipped to deal with all environments equally well. A general purpose robot, given a concrete goal for which it is not configured correctly, must be able to use objects from its environment to reach its goals. In an anthropocentric environment this means that correctly wielding tools designed for human use gives a robot a degree of operational flexibility that would be otherwise impossible.

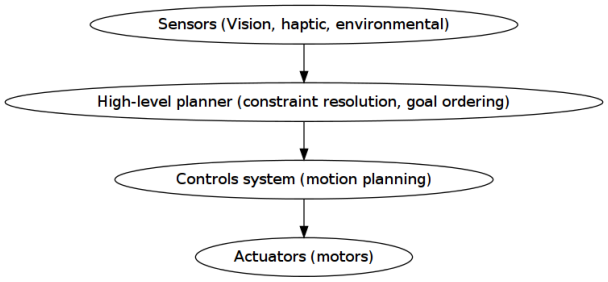

Fig. 1: Layer model of robot control systems.

Using human-configured tools requires multiple layers of control and planning, as shown in figure 1.

1) Sensors - The robot must be capable of discerning its environment, discovering what constraints that environment will impose upon it, and constructing subgoals in order to accomplish its target task. In an un-augmented anthropocentric environment, this means having a sufficiently sophisticated computer vision system (visual, radar, lidar, etc) to map the environment, recognize objects in the environment, and classify them as a given tool. It also means using haptic feedback (if possible) and torque feedback from the actuators during manipulation tasks as part of a control loop to ensure that the manipulated objects are behaving according to plan.
2) Planner - The robot needs a high level planner in order to take a goal state, break it into sub goals, match those sub goals with tasks it is able to handle, resolve any dependency conflicts, and order those goals into as efficient a plan as possible.
3) The control system is responsible for implementing each of the tasks necessary to accomplish a given sub-goal. This involves things like the lower-level motion planning necessary for manipulation, including workspace control.
4) Actuators - The robot must be capable of using its actuators to accomplish commands sent to them from the control system. Physically, one must be able to manipulate the tool in question. General purpose manipulators are not generally well suited for using tools designed for humanoid hands, and most commonly available humanoid hands have issues with grip strength, digit control, haptic sensing, and friction.

This document describes an effort to explore the challenges associated with general purpose tool usage from within a simulated environment. To reduce the scope of an otherwise unmanageable problem, the goal was simplified to be the control system for a single simulated Schunk LWA-4 arm, a 7-degree-of-freedom arm with an attachable manipulator. A screwdriver was chosen as the target tool, since it is the simplest tool that could be easily manipulated by a Schunk arm and still require a complex set of actions. In order to remove the necessity for a complicated sensor system, the work was done using the GRIP/DART simulation environment used by the Humanoid Robotics laboratory at Georgia Tech.

Complications arose during the work which prevented all our goals from being reached. The implementation of the vital control elements of the proposed system was unusable due to bugs in the implemented workspace control. These were fixed to a usable point, but there was not enough time remaining to integrate the high level planner into the decision loop. As such, the robot is capable of all the physical manipulation tasks needed to reach its goal state, but those states must be manually traversed by a human in order for the simulation to work. These issues will be further discussed in later sections.

II. Related Work TODO

Tool manipulation in workspace is very common in both research and industry. However, most of this tool manipulation is with specialized tools that are not designed for human manipulation, and the robots are generally used in environments where the state space is largely deterministic – the planner for the robot can safely assume that a part will enter the assembly line at a certain time, reach a certain location in a certain

orientation, and if there are any deviations from this expected behavior it will simply stop and alert a human of a problem.

What made our planned approach unique is that we account both for the motion planning for the robot as well as the constraints of the tool. For example, when driving the screw into the target block, we must follow the screw with the driver (held in the manipulator) as it threads into the target block by matching 'downward' translation with the rotation of the screw driver, then back the tool out and repeat until the screw is settled into the block. It also took into account the set of actions required to set up the environment – finding the screw, placing it in the correct location for tool use, finding the tool, and finally driving the screw into the target block.

[Tucker] [general grasp]

III. Methods

A. Simulation Software TODO

In the current experiments, there are three interactive objects and one robot arm. The interactive objects are a screwdriver, a bolt, and a goal block, all located within the arm's range of movement.

Our system then follows the following set of actions to generate a plan.

B. High Level Planning TODO

1) Locate screw: After the screw is inserted into the world file, it is passed to the robot as a motion goal point. The robot queries the world to find the object and obtains the transform of the current tool position in the world's coordinate reference frame. The 4x4 affine matrix representing the rotation and translation of the screw is then converted into a 6x1 vector:

$$q = \begin{bmatrix} a \\ w \end{bmatrix} \tag{1}$$

Here, a is a three dimensional translation vector and w is a roll, pitch, and yaw vector relative the world origin which define the origin of the screw.

2) Grasp the screw: Jacobian workspace control is used to align the robot's manipulator with the coordinate frame of the screw. After matching rotation and position with the target, the robot grabs the screw. Subsequently, the screw is manipulated as though it were an additional link in the arm.

3) Locate goal:

1) Move the bolt to alignment with the goal.

4) Release bolt:

5) Grasping tool: Grasp the screwdriver; updating the end effector after each interaction.

6) Move tool to screw: After moving the screwdriver into the bolt, update the arm to include the bolt as well

7) Drive screw into goal point: Moving it into the goal position.

8) Release bolt: Release the bolt

9) Release tool: Replace the screwdriver back to its original position.

C. Low Level (Motion) Planning

1) Resolved Rate Trajectory Planning: Resolved rate trajectory planning, or pseudo-inverse Jacobian control, was used to move the manipulator from a current world configuration $\mathbf{x_i} = \begin{bmatrix} x_i y_i z_i R_i P_i Y_i \end{bmatrix}^\top$ to a target goal configuratfon $\mathbf{x_f} = \begin{bmatrix} x_f y_f z_f R_f P_f Y_f \end{bmatrix}^\top$ in a linear fashion. The world coordinates are described as 6×1 vectors of X,Y, and Z positions with corresponding Roll, Pitch and Yaw.

Resolved rate trajectory control stems from 2, where $\mathbf{V_e}(t)$ is the end effector velocity at time t, α is the joint space position vector describing the current configuration of the robot, and is the joint space velocity vector.

$$\mathbf{V_e}(t) = \mathbf{J}_\alpha \dot{\alpha} \tag{2}$$

This can be rearranged into 3.

$$\dot{\alpha} = \mathbf{J^{-1}} \mathbf{V_e}(\mathbf{t}). \tag{3}$$

Thus, if the inverse Jacobian and desired end effector trajectory are know, it is possible to make a differential equation for the joint angles of the manipulator. In this case, the trajectory is to be linear, so $\mathbf{V_e}(\mathbf{t})$ was a constant 6×1 vector equal to $\mathbf{x_f} - \mathbf{x_i}$.

- Extracting a target coordinate
  The world coordinate of a given node is generated from DART/GRIP as an affine transformation described by the 4×4 homogeneous coordinate matrix $C = \begin{bmatrix} \mathbf{R} & \mathbf{X_{xyz}} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}$, so the 6×1 XYZRPY vector representation must first be extracted.
  $\mathbf{X_{xyz}}$ is a 3×1 vector describing a point in 3-space, and does not require any manipulation. $\mathbf{R}$ is a 3×3 rotation matrix, so we convert this to roll-pitch-yaw as shown in equation 4:

$$\mathbf{X_{rpy}} = \begin{bmatrix} \tan^{-1}(\frac{R_{2,1}}{R_{2,2}}) \\ -\sin^{-1}(R_{2,0}) \\ \tan^{-1}(\frac{R_{1,0}}{R_{0,0}}) \end{bmatrix}, \tag{4}$$

  This yields our target coordinates $\mathbf{x_f} = \begin{bmatrix} X_{xyz} \\ X_{rpy} \end{bmatrix}$.
- Inverting the Jacobian
  A direct inverse of the Jacobian is not possible in our case, as our manipulator had 7 degrees of freedom producing a 7×6 matrix. Instead, a Moore-Penrose pseudo-inverse was calculated according to equation 5.

$$J^\dagger = J^\top (J J^\top)^{-1} \tag{5}$$

  From this we compute for each time step the change in joint angles via equation 6:

$$\dot{\alpha} = J_\alpha^\dagger V_e(t) \tag{6}$$

  This is added to the previous time step's joint configuration and the robot is updated. The process continues until

the global coordinates (both rotation and translation) of the end effector of the manipulator is less than a threshold distance ϵ of the desired position. In other words,

$$\|\mathbf{x_f} - \mathbf{x_i}\| < \epsilon \tag{7}$$

- Joint limits
  This equation for the joint velocities is not always well behaved. In order to improve the results, we will use a variation of the joint-limits avoidance strategy described in [**?**].
  We now know from 6 the regular form of our velocity equation used to generate a trajectory toward our primary task. We will now augment that with an additional secondary task that will bias the trajectory towards keeping the joints as close to their zero points as possible.
  First, we augment equation 6, so our equation describing the change in joint angles is now 8.

$$\dot{\alpha} = \mathbf{J}_\alpha^\dagger \mathbf{V_e(t)} + (\mathbf{1} - \mathbf{J}_\alpha^\dagger \mathbf{J}_\alpha)\mathbf{h} \tag{8}$$

  The vector h is our secondary task, multiplied by $(\mathbf{1} - \mathbf{J}_\alpha^\dagger \mathbf{J}_\alpha)$, the orthogonal complement of J_α. The result is a projection of $\mathbf{h}$ into the null space of $\mathbf{J}_\alpha$, a "virtual motion" which results in a bias toward the secondary objective.
  The secondary task is constructed as $\mathbf{h} = z$, where z is a fitness function described in 9.

$$\mathbf{z} = \frac{\mathbf{1}}{\mathbf{2}}(\alpha - \bar{\alpha})^\top \mathbf{W}(\alpha - \bar{\alpha}) \tag{9}$$

  The variable $\bar{\alpha}$ is the joint position vector describing the mid-joint position; in our case, this is an all-zero vector, as the entire arm is constructed from symmetrically revolute joints centered around zero.
  The matrix $\mathbf{W}$ is a diagonal weight matrix describing the acceptable deviation from this zero point – if it ;is desirable for a particular joint to have a greater latitude during movement than another, altering the corresponding row in the weight matrix will generate that effect. In our case, an identity matrix was used, as we only needed a general bias away from the joint limits.
  The effect is that $\mathbf{h}$ describes a gradient where the zero positions on each joint is a minima, growing steeper as you progress further away from this center point. When projected into the null space of $\mathbf{J}_\alpha$, it will cause joints that are otherwise unconstrained by the target motion to descend toward their zero positions, preventing the robot from approaching too close to the joint limits and preventing some of the anomalous behavior produced by unconstrained inverse Jacobian control.

D. Object manipulation

In the real world, the actual manipulation of an object is a trivial task – once you grasp the object, it will behave as defined by the laws of physics, and can be treated as an extension of the end effector. However, the DART/GRIP simulation package does not currently support true connection of manipulated world objects as parts of the kinematic model of the robot, so the position of the world object must be manually updated according to the position of the end effector that is "grasping" it.

Maintaining the relationship between the end effector and the grasped object is difficult using XYZ-RPY coordinates since any accidental change in the order of rotation will cause resulting transform to change. However, describing the different positions and orientations of the end effector and target object as the 4×4 matrices of homogeneous coordinates describing the affine transformation used to take them from the global origin to their current global coordinates simplifies the task immensely.

Instead of making multiple operations to transform the grasped object via relative rotations and translations from its previous position, we need now only describe the affine transformation $\mathbf{R}$ which takes an object at coordinates $\mathbf{O}$ to the end effector coordinates $\mathbf{E}$, as calculated in equation 10.

$$\mathbf{R} = \mathbf{E}^{-1}\mathbf{O} \tag{10}$$

Subsequently, updating the position of object $\mathbf{O}$ to $\mathbf{O}'$ is a simple matrix multiplication of the new end effector location $\mathbf{E}'$ by the relationship $\mathbf{R}$, as shown in equation 11.

$$\mathbf{O}' = \mathbf{E}'\mathbf{R} \tag{11}$$

Setting the new location requires transforming the homogeneous coordinates back into global XYZ-RPY coordinates, as DART/GRIP do not currently expose a function for setting an objects position using homogeneous coordinates, but performing the transformations with homogeneous coordinates greatly reduces the risk of error.

E. Task motion – driving the screw

Once the screw is in place, the screw driving motion is performed to keep it stationary at the block while the screwdriver is retrieved. The screw driving motion pattern consists of a rotation of the end effector about its axis by 60 degrees, an ungrasping event, derotating the end effector 120 degrees (for a net -60 degrees away from the zero point), regrasping, and repeating as necessary.

A similar motion pattern is used when the screwdriver is in use, though with a significant offset to compensate for the longer tool. Though currently unimplemented, a second modification to the motion pattern when the tool is in use is a motion away from the tool during the derotation rather than an ungrasping and regrasping motion that leaves the tool suspended in midair.

IV. Experiments

A. Analysis

B. Discussion