Thesis Proposal for the Master of Science Degree in Computer Science

Presented by Glenn V. Nickens
Advised by Dr. David Touretzky

Thursday, January 29, 2009

HIGH-LEVEL MANIPULATION PRIMITIVES FOR A ROBOT ARM

Supported by National Science Foundation awards 0717705 and 0742106 to Carnegie Mellon University, and 0742198 to Norfolk State University.
Outline

• Introduction to the research topic
• Problem statement
• Accomplished background work
• Proposed method of completion and testing
• Expected outcome and contributions
• Timeline
Introduction

• Access to the field of robotics has been limited in computer science undergraduate programs due to:
  – The complexity of the underlying mathematics and theory that is needed to implement kinematics, path planning and manipulation
  – The size, cost and/or limited capabilities of available robots

• To help solve this problem, researchers at Carnegie Mellon University have developed:
  – The Tekkotsu software framework
  – An inexpensive hand-eye robot ($995 from RoPro Design)
Introduction

• Tekkotsu
  – is an open source robot programming framework based on C++.
  – is an object-oriented and event passing architecture that makes full use of the template and inheritance features of C++.
  – programmers use high-level primitives, such as “nod your head” and “walk to that location”, to control robots.
Introduction

• The hand-eye robot:
  – has a three-link planar arm, with a webcam attached to a pan-tilt above it for vision.
  – is a small, affordable robot that fits on a classroom table.
  – can be programmed to manipulate objects identified by the camera.

• However, Tekkotsu does not have high-level manipulation primitives for an arm.
Introduction

• Creating the high-level primitives necessary to perform manipulation with the arm is the subject of this research proposal.

• Once the primitives are implemented, I will test them by writing a tic-tac-toe player.

• The player will use only the high-level primitives to manipulate the game pieces.
Problem Statement

I will:

– design the high-level manipulation primitives required to program the hand-eye robot.
– formalize and implement the algorithms for the primitives.
– demonstrate that the primitives are suitable for performing complex tasks.
Background

• To create the manipulation primitives the following are necessary:
  – A robot arm
  – Kinematics
  – Collision detection
  – Path planning
  – Path smoothing
Three-Link Planar Arm

- Three Dynamixel AX-12 servos
- Upper arm and forearm are constructed of aluminum tubes and elongated c-brackets
- End-effector is a c-bracket
Manipulation Surfaces

• No closeable fingers
• Must move objects by pushing them
• Three manipulation surfaces:
  – Interior of the c-bracket
  – Wrist
  – Forearm & Upper arm
Why not two links?

- Manipulation in the plane requires at least 3DOF (x, y, and $\theta$).
- A two-link arm only has 2DOF.
- A three-link arm can reach a target from many angles; a two-link arm can reach it from only two angles.
Why not four or more links?

• Given a target and angle of manipulation, a three-link arm will have at most two solutions, a four-link arm may have infinitely many solutions.

• Solving kinematics for a four-link arm is more complex than for a three-link arm.

• Links are expensive (power, weight, wiring).
Kinematics

- Kinematics describes the relationship between the position and orientation of the end-effector and the joint angles of the arm.
Forward Kinematics

• Forward/direct kinematics calculations determine the end effector configuration given the joint angles.

• Forward kinematics calculations are easily solved with a series of matrix multiplications, and will always produce a unique solution.

\[
\begin{bmatrix}
c \theta_1 & -s \theta_1 \\
s \theta & c \theta_1
\end{bmatrix}
\begin{bmatrix}
L_1 \\
0
\end{bmatrix}
+ \begin{bmatrix}
c(\theta_1 + \theta_2) & -s(\theta_1 + \theta_2) \\
s(\theta_1 + \theta_2) & c(\theta_1 + \theta_2)
\end{bmatrix}
\begin{bmatrix}
L_2 \\
0
\end{bmatrix}
+ \begin{bmatrix}
c(\theta_1 + \theta_2 + \theta_3) & -s(\theta_1 + \theta_2 + \theta_3) \\
s(\theta_1 + \theta_2 + \theta_3) & c(\theta_1 + \theta_2 + \theta_3)
\end{bmatrix}
\begin{bmatrix}
L_3 \\
0
\end{bmatrix}
\]
Inverse Kinematics

- Inverse kinematics calculations search for a set of joint angles that produce a specified end effector configuration.
- Inverse kinematics calculations are harder; there may be multiple solutions, or there may be none.
Inverse Kinematics

• In this research, inverse kinematics is used to determine the shoulder, elbow, and wrist angles given a target position and the manipulation surface orientation.

• The inverse kinematics calculations for the arm use a four step process based on a common analytical approach documented in the book ‘Robot Modeling and Control’ (Spong 2006).
IK Calculation Process

Determine the:

1) wrist position \([x_w, y_w]\) from the target point \([x_t, y_t]\) and desired orientation, \(\phi_t\).
2) elbow angle, \(\theta_2\), from the wrist position \([x_w, y_w]\) and arm dimensions, \(L_1\) and \(L_2\).
3) shoulder angle, \(\theta_1\).
4) wrist angle, \(\theta_3\).
IK Step 1

• Determine the wrist position \([x_w, y_w]\) from the target point \([x_t, y_t]\) and desired orientation, \(\phi_t\).

\[
\begin{bmatrix}
  x_w \\
  y_w
\end{bmatrix} = \begin{bmatrix}
  x_t \\
  y_t
\end{bmatrix} - L_3 \cdot \begin{bmatrix}
  \cos(\phi_t) \\
  \sin(\phi_t)
\end{bmatrix}
\]
IK Step 2

• Determine the elbow angle, $\theta_2$, from the wrist position $[x_w, y_w]$ and arm dimensions, $L_1$ and $L_2$.

\[
D = \frac{x_w^2 + y_w^2 - L_1^2 - L_2^2}{2L_1L_2} = \cos(\alpha)
\]

\[
\theta_2 = \tan^{-1} \frac{\pm \sqrt{1 - D^2}}{D}
\]
Multiple Solutions

• The step for determining the elbow angle, $\theta_2$, may produce two solutions.
• These are called the elbow up and elbow down configurations.
IK Step 3

• Determine the shoulder angle, $\theta_1$.

$$\theta_1 = \tan^{-1}\left(\frac{y_w}{x_w}\right) - \tan^{-1}\left(\frac{L_2 \sin(\theta_2)}{L_1 + L_2 \cos(\theta_2)}\right)$$
IK Step 4

- Determine the wrist angle, $\theta_3$

\[ \theta_3 = \phi_t - (\theta_1 + \theta_2) \]
IK Modifications

• The inverse kinematics calculation process described thus far is ideal for arms whose joints turn without bound and whose environments are obstacle-free.

• Modifications have been made to these calculations to create an IK solver specifically for the arm in this research.
IK Modifications

- The joints cannot turn a full 360° because adjacent arm segments will collide.
- The IK solver only uses configurations that are within the turning limits of each joint.
IK Modifications

- Sometimes a target is not reachable from a desired orientation, but may be reached from another orientation.
- The IK solver will search around the desired orientation at 10° intervals for 180° in both directions until a valid solution is found.
IK Modifications

- When both elbow up and elbow down configurations are valid, the IK solver must choose one.
- It chooses the one that has the same elbow direction as the arm’s current configuration.
IK Modifications

• The IK solver uses a collision detection algorithm to check for collisions of the arm with itself or with objects in the environment.
• Arm segments and obstacles are represented as rectangles.
• The algorithm is based on the Separating Axis Theorem.
Path Planner

Once the kinematics solver has determined the start and end configurations for the arm, a collision free path must be computed to move the arm.
Path Planning

• Path planning is the construction of a continuous sequence of arm configurations from the start to the end configuration while respecting constraints (Kuffner 2009).

• The constraints in this research are the arm’s joint limits, obstacles in the environment, and the need to maintain contact with objects.

• The path planner in this research is based on an algorithm called RRT-Connect (Kuffner 2000).
Rapidly-exploring Random Trees

- Rapidly-exploring Random Trees (RRTs) are trees whose vertices encode configuration states of the arm (LaValle 1998).
- The RRT grows by alternately extending the tree in random directions and moving toward the goal configuration.
RRT-Connect

- RRT-Connect grows two RRTs, one from the start and one from the goal configuration, and biases the trees to grow toward each other.
- Once the RRTs connect, the path is extracted using backtracking.
RRT-Connect

• Path extracted using backtracking from the two trees.
Path Smoothing

• The random component of the RRT-Connect search often results in a jerky and meandering solution, therefore a smoothing algorithm is applied to the path.

• Smoothing is accomplished by selecting random segments to be snipped from the path after verifying that the shortened path will still be collision-free.
Paths

• The pictures to the right show the arm’s trajectory along a path from the start (green) to the end (red) configuration.

• The first image shows a path constructed by the path planner.

• The second image shows the same path, but after the smoothing algorithm has been applied to it.
Methodology

• In this section I will discuss:
  – The high-level primitives I have chosen so far
  – How I plan on implementing them and
  – How I will demonstrate the success of my work
High-Level Primitives

Move an object to a target position
High-Level Primitives

Sweep a collection of objects
High-Level Primitives

Pick an object out of a group
High-Level Primitives

Adjust the orientation of an object
High-Level Primitives

Propel an object in a direction
Manipulation Planner

• To realize the primitives, I will develop a manipulation planner to determine:
  – Which manipulation surface to use
  – How to acquire contact with an object
  – What trajectory the arm should follow
Planning Process Complications

• Joint limits and obstacles constrain the arm’s freedom of motion.
• The forearm will collide with the upper arm or the red obstacle if it tries to move inward or outward, respectively.
Planning Process Complications

Planned trajectories must keep the manipulation surface in contact with the object
Planning Process Complications

Moving an object with the upper arm or the forearm is imprecise because it may slide.

Initial point of object on upper arm

Object slid along upper arm
Planning Process Complications

Some tasks may require multiple manipulations

In this sequence of images, the arm first moves the blue object with its forearm, then moves it with the c-bracket to get it to the target position.
Executing the Plan

• I will develop a manipulation engine to execute the plans produced by the manipulation planner.

• The engine will:
  – Monitor the execution of each step
  – Take corrective action should something go wrong
Manipulation Engine

• Monitoring will be done by:
  – verifying arm positions by checking the joint angles.
  – verifying object positions using vision.

• Corrective action may have to be taken because:
  – The arm may not have acquired the object.
  – The object may not be where it should be.
  – The arm may have accidentally moved the object.
Validation: A tic-tac-toe player

• The player will use only three of the primitives mentioned earlier:
  – Moving a game piece
  – Sweeping game pieces off the board
  – Picking a game piece from a group

• The tic-tac-toe player will not reference any low-level concepts such as kinematics, path planning or manipulation surfaces.
Playing tic-tac-toe
Contributions/Expected Outcome

1. My work will allow robot programmers to easily manipulate objects by specifying desired effects and letting the planner figure out how to achieve them.

2. Students in undergraduate robotics courses will be able to experiment with manipulation without having to write their own kinematics solvers, path planners, or collision detection algorithms.
Contributions/Expected Outcome

3. The manipulation planner and engine will be integrated into the Tekkotsu framework and will be used on other robots, including the Chiara.
# Timeline/Milestones

<table>
<thead>
<tr>
<th></th>
<th>06/08</th>
<th>12/08</th>
<th>01/09</th>
<th>02/09</th>
<th>05/09</th>
<th>06/09</th>
<th>07/09</th>
</tr>
</thead>
<tbody>
<tr>
<td>Background</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thesis Proposal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Develop</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manipulation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Primitives</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thesis Paper</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thesis Defense</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
References


Questions