My Summer at Willow Garage

Benjamin Cohen
University of Pennsylvania
The Bosses

Dr. Sachin Chitta
my mentor
Professor Maxim Likhachev
my advisor
My Summer Goals

SBPL Arm Planner
continue development of new algorithm
integrate into move_arm
perform tabletop manipulation
publish

SBPL Door Planner
(Sachin did all of the integration. I'm working on the algorithm and debugging it with him)
continue development of algorithm
make a sweet video
publish
SBPL Arm Planner

Goal:
Develop a motion planner for robotic manipulation that can robustly plan paths in cluttered environments.
Why not run IK and plan to a known valid joint configuration?

IK is very fast but may not find the 'best' solution
may find a visually awkward solution
may orient the arm very close to obstacles
may choose a solution very close to joint limits
Why not run IK and plan to a known valid joint configuration? (cont'd)

Let the planner optimize the goal joint configuration according to the desired cost function:
- length of path
- smoothness
- distance from obstacles
- keep away from joint limits
Graph Construction

similar to lattice based planners in planning dynamically feasible trajectories

a discretization of the configuration space into a set of states, the edges that connect the starts are dynamically feasible

edges are pre-defined motion primitives
Motion Primitives

7 basic primitives:

- $m_1$: shoulder pan
- $m_2$: shoulder pitch
- $m_3$: upper arm roll
- $m_4$: elbow pitch
- $m_5$: forearm roll
- $m_6$: wrist pitch
- $m_7$: wrist roll
Costs

Smoothing Cost
minimize change in velocities between states
keep the same joint(s) moving if possible

Action Cost
minimize the number of actions to the goal

Distance Cost
apply a cost for planning close to obstacles

\[ c(s, s') = c_{cell}(s') + w_{action} \cdot c_{action}(s, s') + w_{smooth} \cdot c_{smooth}(s, s') \]
Heuristic

3D Dijkstra search
robust to obstacles
pre-computed
directs the end effector to the goal pose
difference in orientation of the end effector

Goal

Dijkstra Shortest Path
ARA : **Anytime Random A***

is consistent

guarantees completeness

'anytime results' - can trade-off path cost vs. computation time
ARA : Anytime Random A*

ARA* finds a feasible solution quickly and then continually works on improving it until the allocated time runs out.

ARA* uses an admissible heuristic (a bound on the plan-execution cost) to focus its search.

Each solution comes with a bound, $\varepsilon$, on its sub-optimality:

$$\text{cost}(\text{solution}) \leq \varepsilon \cdot \text{cost}(\text{optimal solution})$$
Features

given multiple goals, find the path to goal pose with lowest cost
plan to 'goal region'
plan with path constraints, such as:
keep a cup upright
don't obstruct field of view of forearm cameras (future work)
plan to a goal pose in collision, capable of manipulation (future work)
Drawbacks

- high dependency on the heuristic function
- local minima
- doesn't describe kinematics of the arm
- high dimensionality slows down the search
- discretizes paths
Future Work

learn a set of motion primitives with good coverage of the workspace and natural looking movement
add randomization to escape local minima
R* is an A* type search with randomization
research smoothing methods
SBPL Door Planner

Goal:
open doors
inward and outward opening doors
right handed and left handed doors
with collision avoidance
guaranteed feasibility

Method:
ARA* in a 4-Dimensional state space
{x,y,theta,door interval}
door interval: {0,1} (open or closed)
Other Packages

Robot Voxelizer  \( (\text{sandbox/robot\_voxelizer}) \)
wrapper around Ioan's planning models
voxelize robot bodies, meshes, shapes

PM Wrapper  \( (\text{planning\_research/pm\_wrapper}) \)
wrapper around Ioan's planning\_monitor class
easy collision checking, collision map manipulation
Other Packages (cont'd)

Arm Command Tool (sandbox/move_arm_tools)
'safe' movement with move_arm
'unsafe' movement for tabletop manipulation
Thank You!

Scott & Steve
Sachin
Max
Ioan and Mrinal
Everyone else!

thanks!