

# Smooth Motion Planning and Control

Willow Garage Internship, Summer 2009

Mrinal Kalakrishnan

University of Southern California

August 28, 2009

## Smooth Motion Planning

- CHOMP - Covariant Hamiltonian Optimization for Motion Planning, Ratliff et. al., ICRA 2009.
- Implemented in the `chomp_motion_planner` ROS package.

## Smooth Control

- Automatic spline trajectory generation from waypoints.
- Usable by all motion planners.
- Implemented in the `spline_smoother` ROS package.

## Sampling based planners. . .

- Find feasible collision-free paths very fast.
- Typically produce jerky / redundant motion.
- Paths require post-processing, smoothing, optimization.

## Sampling based planners. . .

- Find feasible collision-free paths very fast.
- Typically produce jerky / redundant motion.
- Paths require post-processing, smoothing, optimization.

## CHOMP. . .

- Approaches the problem from a different perspective.
- Inherently generates and optimizes smooth trajectories.
- Uses gradient information to push trajectories out of collision

# CHOMP: The Algorithm

- Create a naive initial trajectory from start to goal.
- Define cost = trajectory smoothness cost + collision cost.
- Run gradient descent on this cost function.
- Not just regular gradient descent. . .

# CHOMP: The Algorithm

- Create a naive initial trajectory from start to goal.
- Define cost = trajectory smoothness cost + collision cost.
- Run gradient descent on this cost function.
- Not just regular gradient descent. . .

The key:

Covariant gradient updates

# CHOMP: The Algorithm

- Create a naive initial trajectory from start to goal.
- Define cost = trajectory smoothness cost + collision cost.
- Run gradient descent on this cost function.
- Not just regular gradient descent. . .

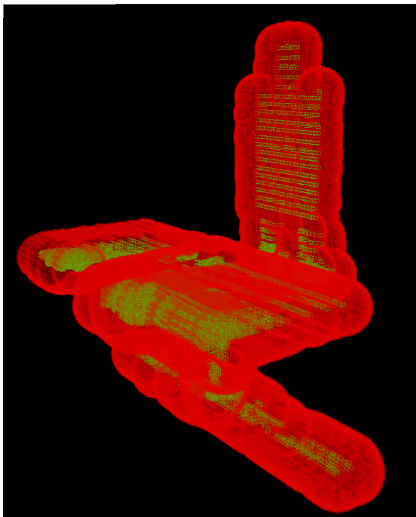
## The key:

### Covariant gradient updates

- Every update to the trajectory is ensured to be smooth.
- Obstacle costs obtained from *signed distance field*.

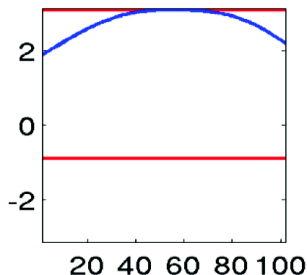
# CHOMP: Distance Fields

- Euclidean Distance Transform in 3-D.
- Each cell contains distance to closest obstacle.
- Gradient information easily obtainable.
- Cartesian gradient converted into joint space using robot kinematics (Jacobian transpose).



# CHOMP: Smooth Joint Limit Projection

- Typical  $L_1$  joint limit projections (clipping) destroy smoothness.
- Smooth the  $L_1$  projection using a covariant update.



Further technical details about CHOMP available in the paper:

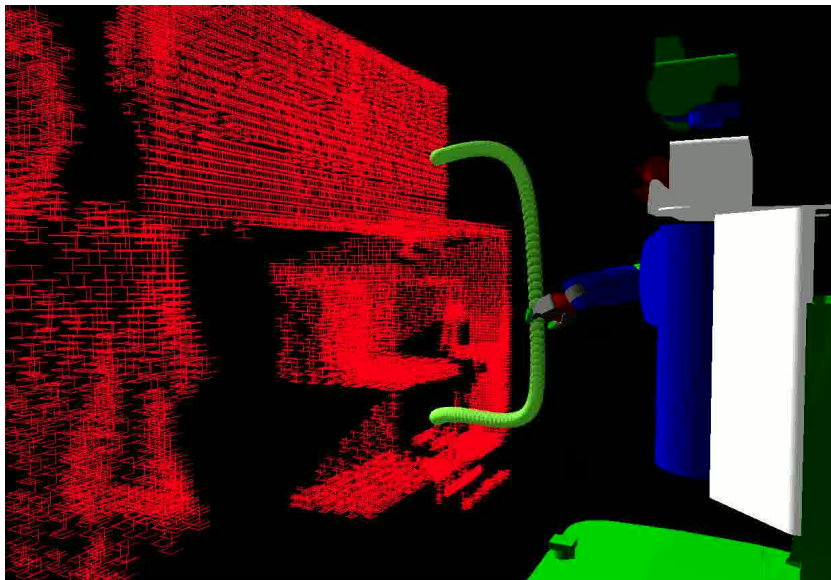
Nathan Ratliff, Matthew Zucker, J. Andrew (Drew) Bagnell, and Siddhartha Srinivasa. IEEE International Conference on Robotics and Automation (ICRA), May, 2009.

# CHOMP: Demonstration (See Video)

Table Bookshelf-top Manipulation



# CHOMP: The Optimization Process (See Video)



# Smooth Control using `spline_smoother`

- A set of algorithms for converting a trajectory of waypoints into splines.
- Use case: Motion planners output a kinematic “path” of joint positions, expect this path to be executed smoothly.
- Currently contains the following three algorithms:
  - Simple numerical differentiation
  - Clamped cubic splines (constrained start/end velocities, continuous velocities and accelerations)
  - Fritsch-Butland monotonic cubic splines
- Optimization-based algorithms not yet available due to lack of BSD-licensed quadratic program solver.
- Currently works in conjunction with `joint_waypoint_controller` to provide a backwards-compatible joint trajectory interface, but with smooth execution.

# Smooth Control using `spline_smoother`

- A set of algorithms for converting a trajectory of waypoints into splines.
- Use case: Motion planners output a kinematic “path” of joint positions, expect this path to be executed smoothly.
- Currently contains the following three algorithms:
  - Simple numerical differentiation
  - Clamped cubic splines (constrained start/end velocities, continuous velocities and accelerations)
  - Fritsch-Butland monotonic cubic splines
- Optimization-based algorithms not yet available due to lack of BSD-licensed quadratic program solver.
- Currently works in conjunction with `joint_waypoint_controller` to provide a backwards-compatible joint trajectory interface, but with smooth execution.

# Smooth Control using `spline_smoother`

- A set of algorithms for converting a trajectory of waypoints into splines.
- Use case: Motion planners output a kinematic “path” of joint positions, expect this path to be executed smoothly.
- Currently contains the following three algorithms:
  - Simple numerical differentiation
  - Clamped cubic splines (constrained start/end velocities, continuous velocities and accelerations)
  - Fritsch-Butland monotonic cubic splines
- Optimization-based algorithms not yet available due to lack of BSD-licensed quadratic program solver.
- Currently works in conjunction with `joint_waypoint_controller` to provide a backwards-compatible joint trajectory interface, but with smooth execution.

Questions, comments?

I am reachable at: `kalakris@usc.edu` **and** `mail@mrinal.net`