

Sampling-Based Planning for Hybrid Systems

by Joshua A. Levine

Advisor: Dr. Michael S. Branicky

Thesis Contributions

- Demonstrates that hybrid systems can be modeled using sampling-based planning
 - Applies Rapidly-Exploring Random Tree (RRT) algorithm
 - Semi-decision result
 - Developed a visual tool to automate modeling
- Presents new results regarding the RRT
 - Convex Hulls, Optimal Paths, Metric Trees

Defense Outline

- Background
 - Hybrid Systems & Sampling-Based Planning
- Development of a Tool
 - Extended Motion Strategy Library (MSL)
- Experimentation
- Investigating & Improving Sampling-Based Methods

Hybrid Systems

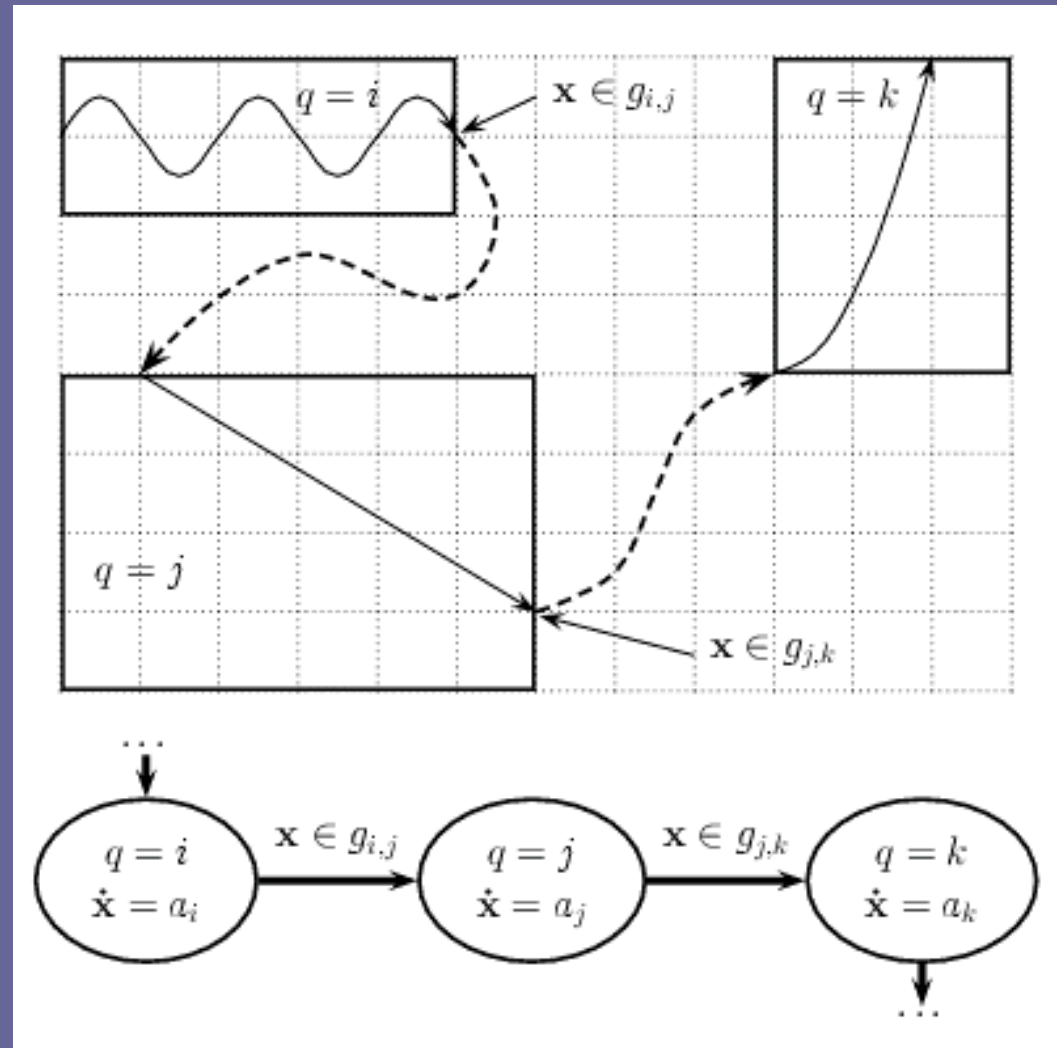
- Model systems that contain both continuous and discrete elements in their descriptions
- Discrete variables represent different state or mode
- Continuous variables change based on mode
- Example: Manual Transmission Car

Hybrid Systems

- Modeled with Hybrid Automaton (HA)
- An HA is defined as: $H = (X, Q, A, E)$
 - X = Continuous State Space
 - Q = Discrete State Space
 - A = Activity Functions
 - E = Edges
- HA are classified by types of functions in A
 - Types: Rectangular, Linear, Nonlinear

Trajectories

- Trajectory defined as a sequence of states in a HA
- Can study both continuous and discrete trajectories



Reachability Problem

- Primary problem to be studied
- Asks “Starting with an initial configuration, is it possible for a hybrid system to reach a given configuration?”
- Rephrased: “Does a trajectory exist from a start state to a goal state?”
- Current techniques focus on studying all reachable paths—limited success.

Sampling-Based Planning

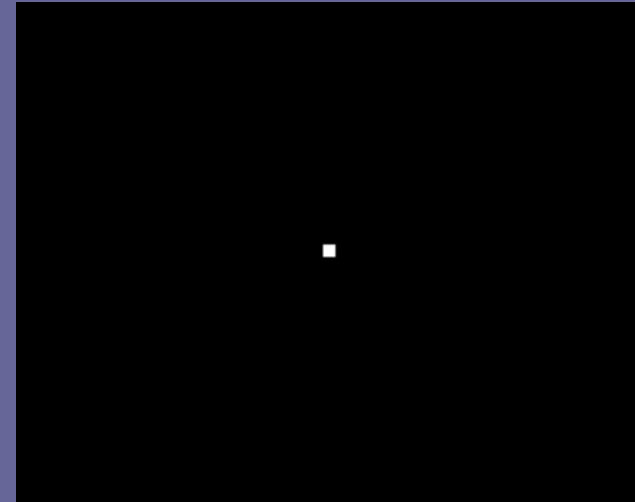
- Determine specific properties of a system from a subset of the state space consisting of a number of samples
- Analogy: A bag of red & blue marbles
- For HA, we sample from the set of all trajectories to answer the reachability problem

Rapidly-Exploring Random Trees

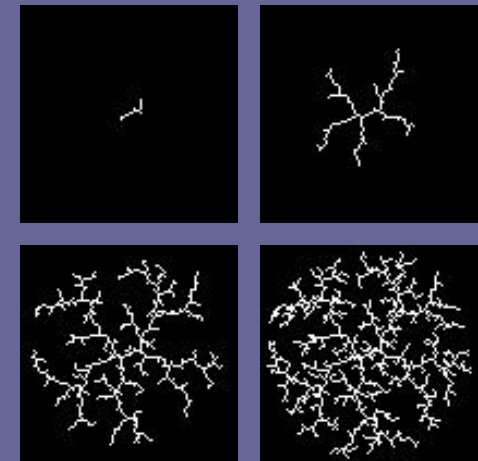
- RRTs introduced by LaValle (1998)
- Originally used to search high-dimensional spaces, particularly motion planning problems
- Samples the space and biases exploration to search the largest unexplored region
- Key point: requires space to have a metric to determine nearest neighbors

RRT Algorithm

```
1 Build_RRT( $x_{init}$ ) {  
2   //initialize tree,  $T$   
3    $T.init(x_{init});$   
4   FOR  $k=1$  TO  $K$  {  
5      $x_{rand} \leftarrow Random\_State();$   
6      $Extend\_RRT(T, x_{rand});$   
7   }  
8   RETURN  $T;$   
9 }
```



```
1 Extend_RRT( $T, x$ ) {  
2   //find node in  $T$  nearest to  $x$   
3    $x_{near} \leftarrow Nearest\_Neighbor(x, T);$   
4   //construct a new node,  $x_{new}$   
5   IF  $New\_State(x_{near}, x_{new}, u_{new}, \Delta t) = TRUE$  {  
6      $T.add\_vertex(x_{new});$   
7      $T.add\_edge(x_{near}, x_{new}, u_{new});$   
8   }  
9 }
```



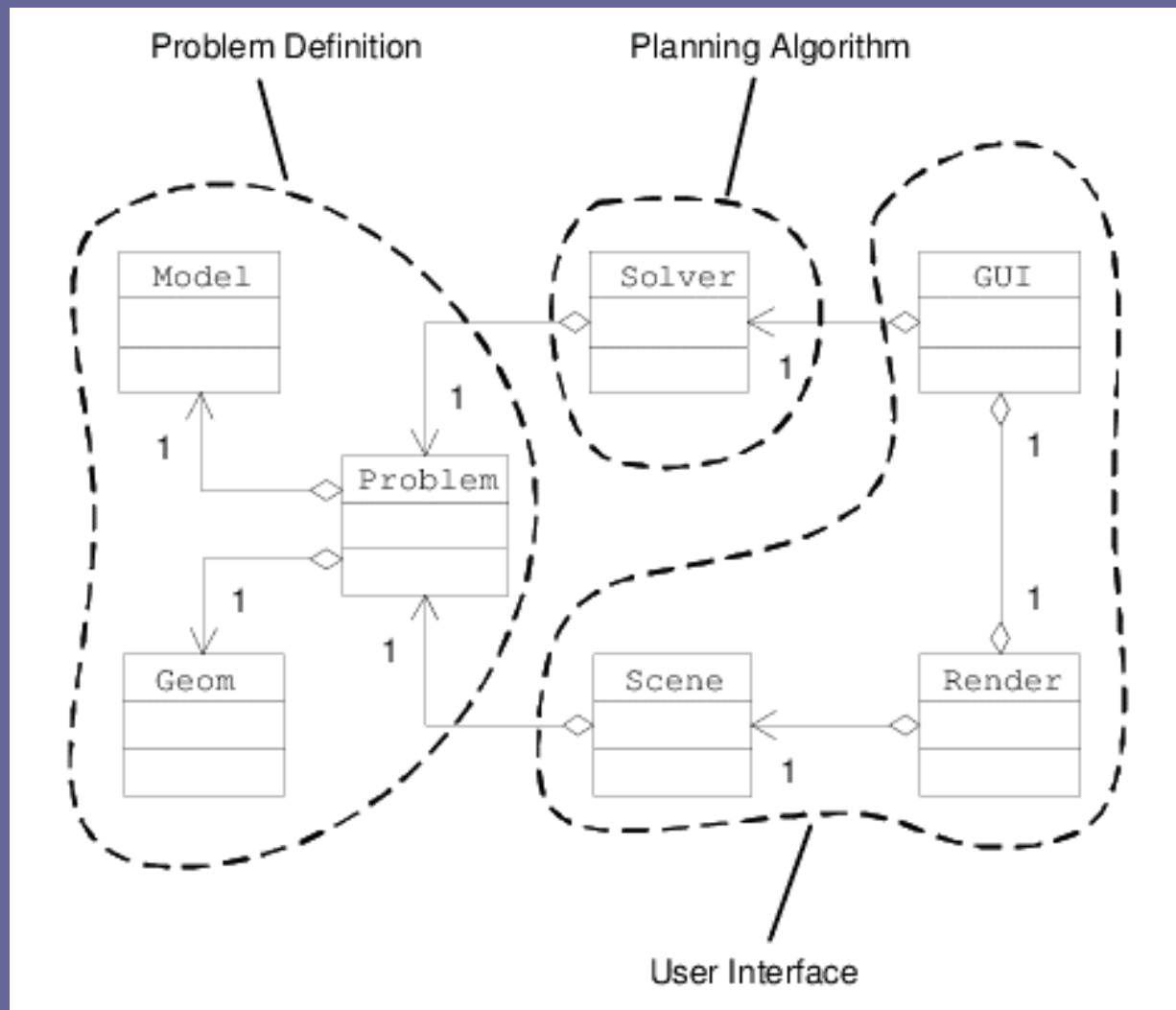
Synthesizing HS & RRTs

- Initially, developed a set of classes to run RRTs in an arbitrary state space
 - Used templates for `State` & `Step` objects
- Created sample experiments to run against this framework: stair climber & peg-in-maze
- Successful, but heavy coding required for each new experiment

Motion Strategy Library (MSL)

- We chose to extend the MSL to allow sample problems of a hybrid nature
- MSL designed to do motion planning using sampling-based techniques
- Visual tool:
 - Excels at viewing problem space
 - Allows multiple planners to be used
 - Problems specified by text input & coding

MSL Class Hierarchy



MSL Class Hierarchy

- `Problem` wraps `Model` & `Geometry`
- `Model` encapsulates system dynamics
- `Geometry` encapsulates physical world
- `Solver` contains the planner algorithms
- `Scene` computes object configurations
- `Render` draws and animates objects
- `GUI` encapsulates user interface

Extending the MSL

- Added hybrid states to `Geometry`
- Added hybrid dynamics to `Model`
- The `Solver` subclassed for hybrid RRTs
- Added hybrid state information to `Render`
- State toggle controlled added to `GUI`
- The `Render` modified to draw RRT
- Threading support added to `GUI`

Stair Climber

- Trying to find a path in a four story building
 - $X = X\text{-}Y$ Plane, $Q = \{1,2,3,4\}$
 - Holonomic

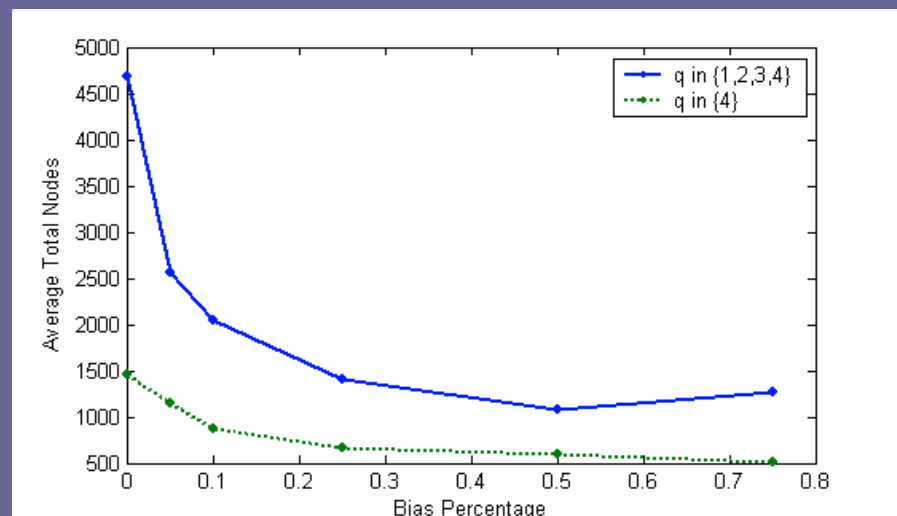
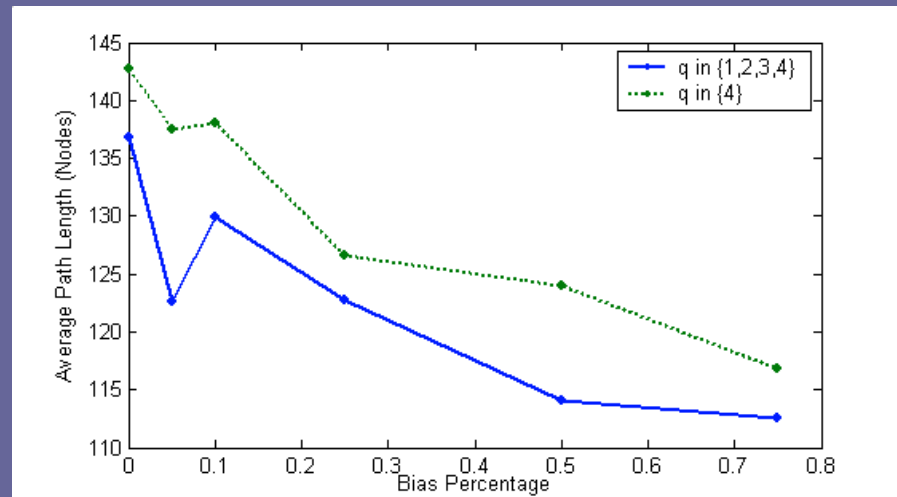
- Makes use of metric:

$$\rho(\mathbf{s}_1, \mathbf{s}_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} + k * |q_1 - q_2|, k \in \mathbb{R}^+$$

- Euclidean distance plus k -factor for discrete
- Originally developed by Curtiss (2002)

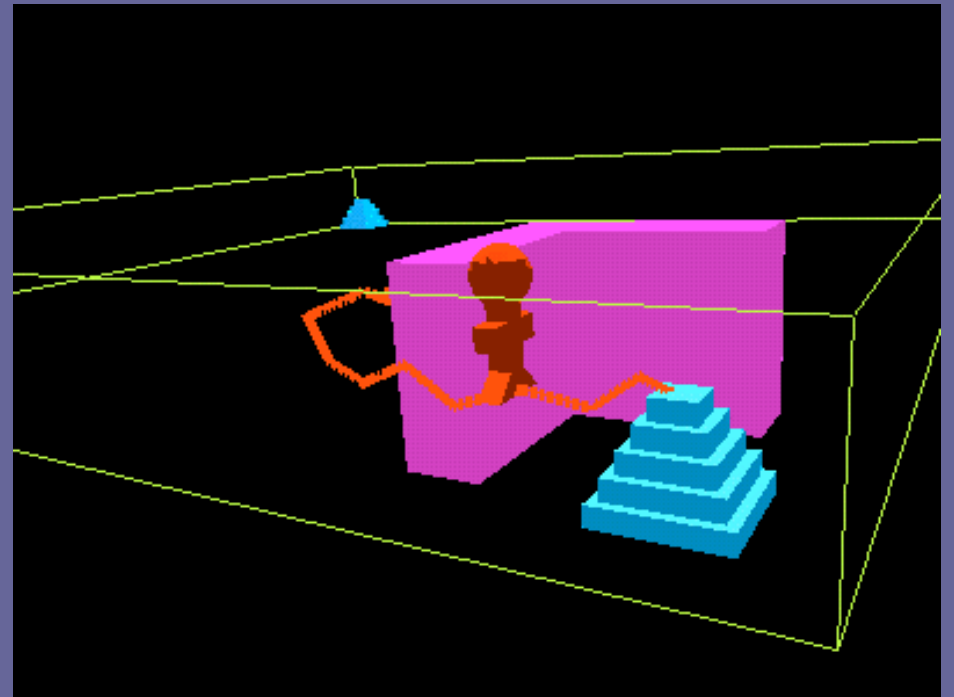
Biased Stair Climber

- Bias the growth of the RRT based on random states selected
- Also created a biased planner to grow towards a set of subgoals



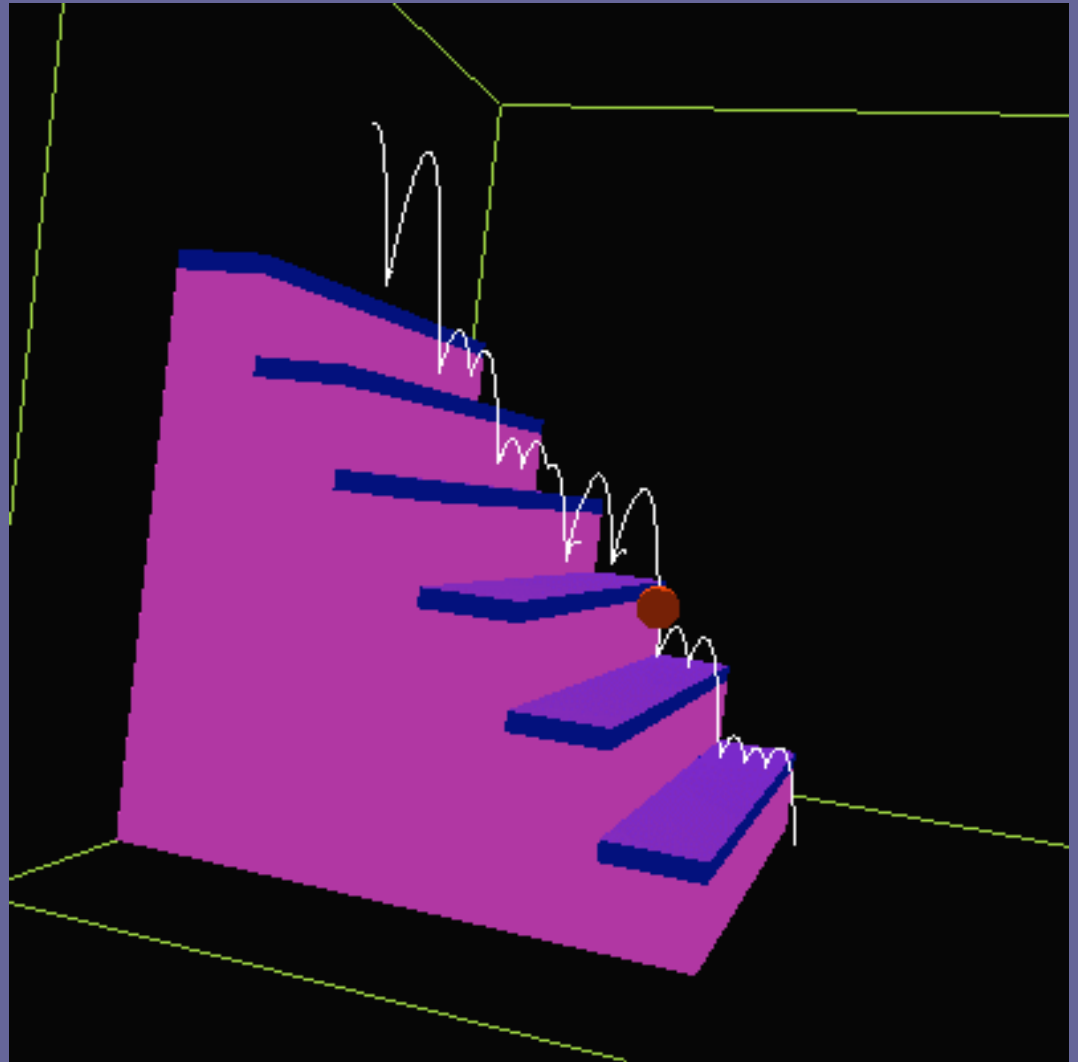
Stair Climber (Cont.)

- Extended 2d to 3d, exploiting MSL's rendering power
- Original stair climber has constant dynamics, extended the `Model` object to support different step sizes based on floor



Ball with Staircase

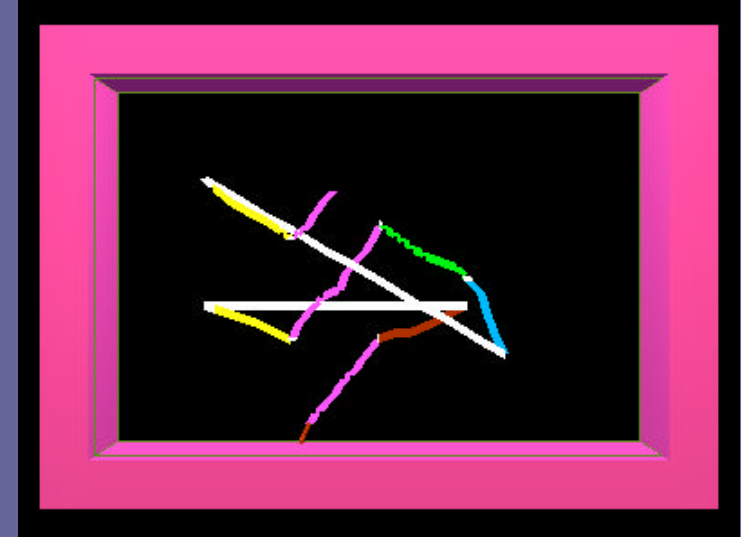
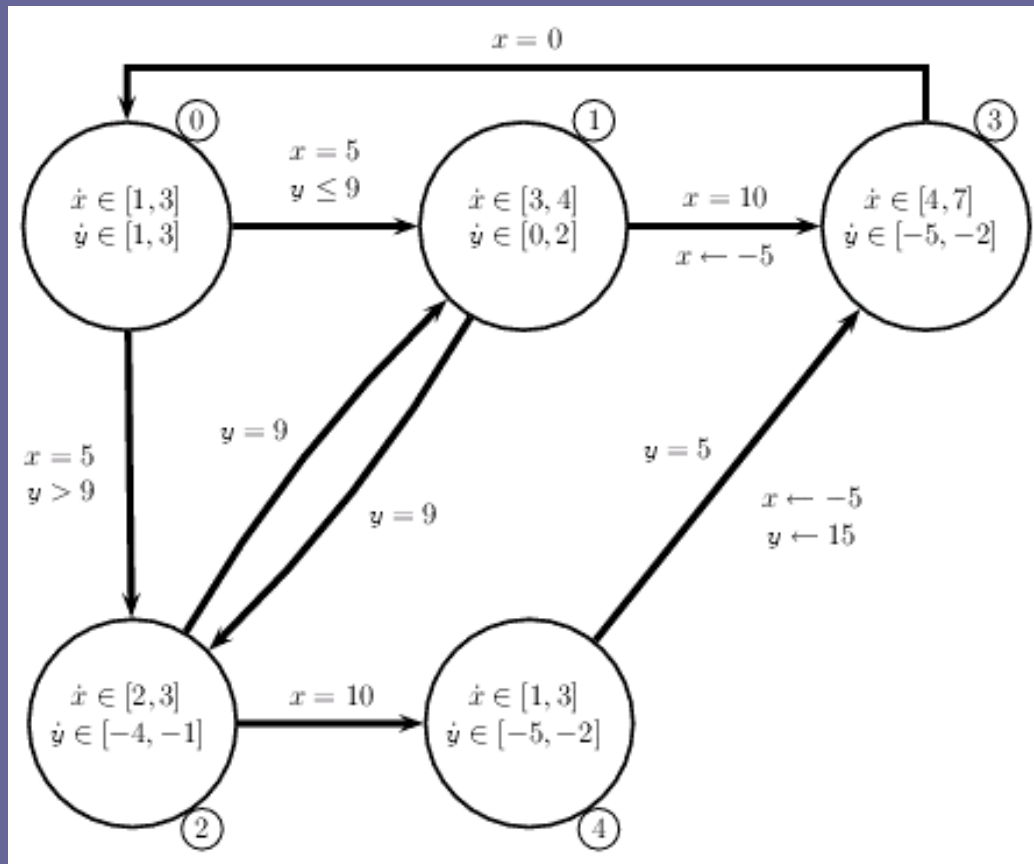
- Models a system with nonconstant dynamics
- Single-state model, when ball collides with step, velocity inverts with random elasticity



Rectangular HA

- Desired more examples with nonconstant dynamics
- Implemented a rectangular HA, where the dynamics of the system are differential inclusions, $\dot{x} \in [L, U], L, U \in \mathbb{N}$
- Requires extending the `Model` to support jump resets as well

Rectangular HA (Cont.)

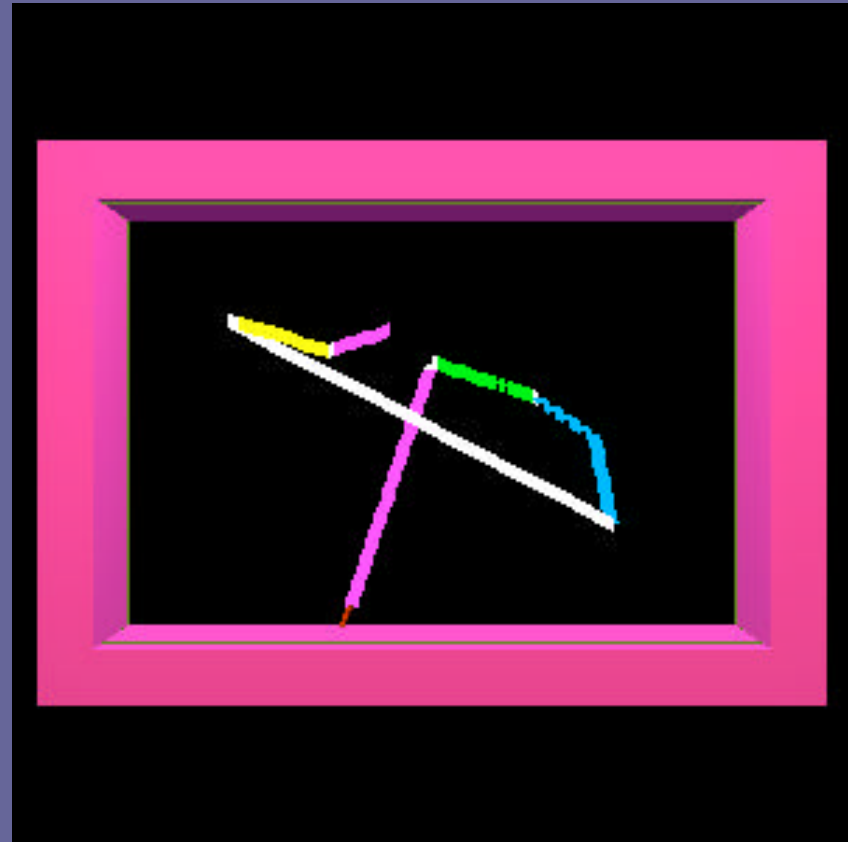
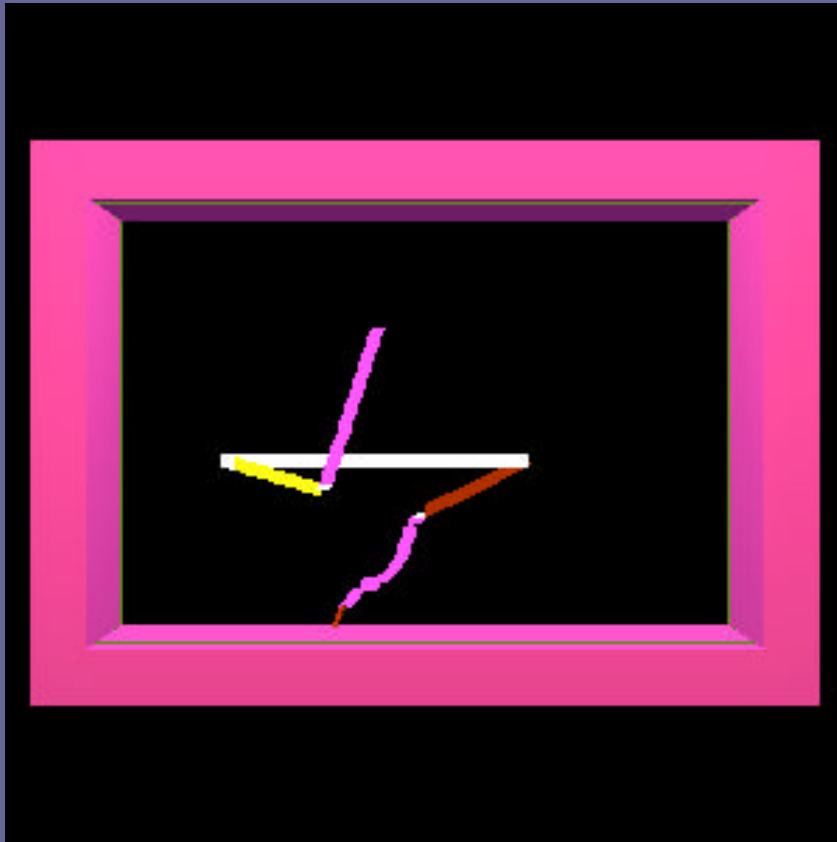


Multi-Action Growth

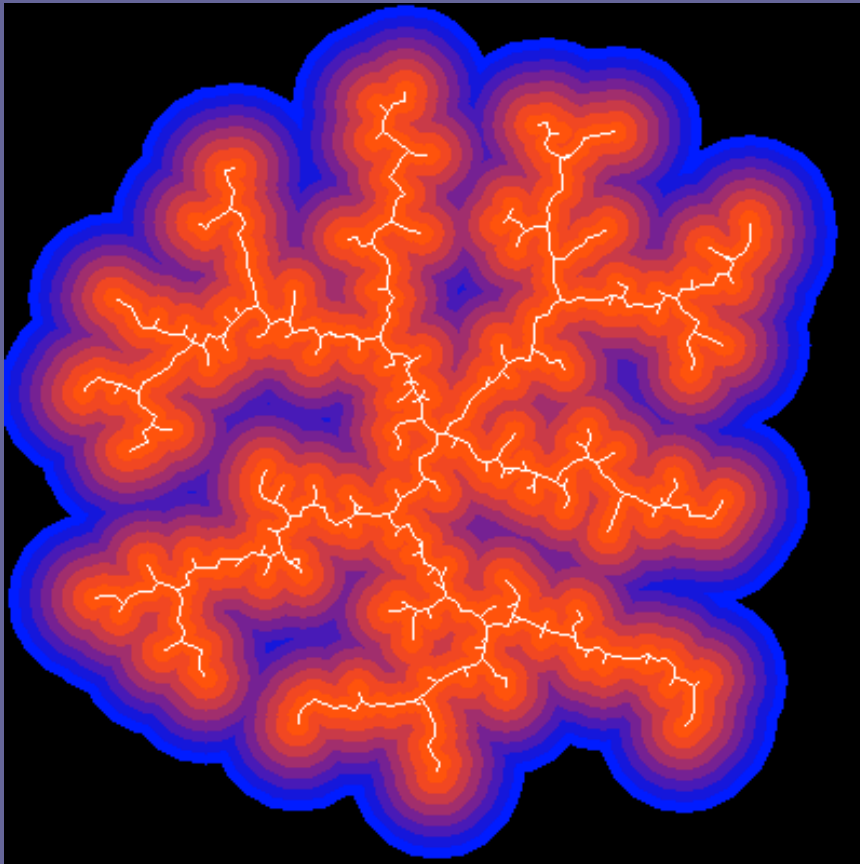
- Created a new Solver that takes steps based on the maximum and minimum values of the activity functions
 - In 2d case, takes four steps: (L_x, L_y) , (L_x, U_y) , (U_x, L_y) , and (U_x, U_y)



Multi-Action Growth (Cont.)



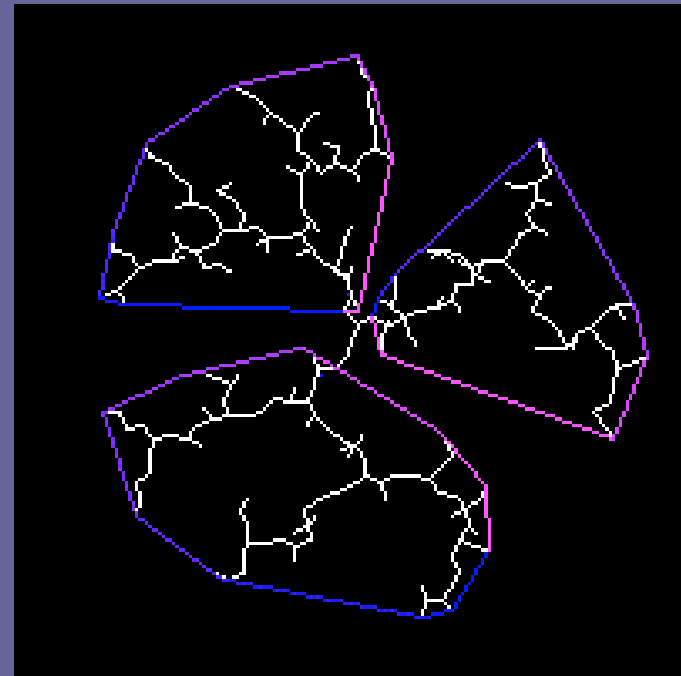
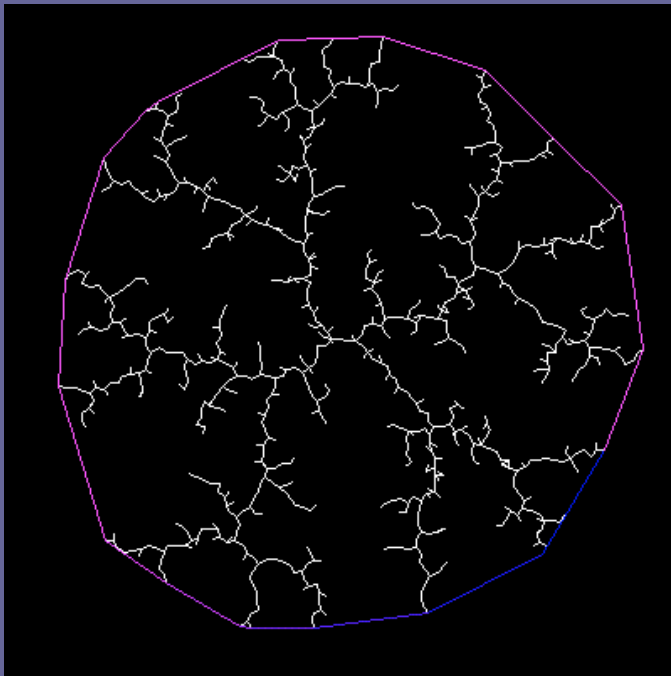
Contour Maps



- Wanted to gain a sense of the reachable area of an RRT
- Draw a set of n concentric discs, varying color, at each node in the RRT
- Clever visual, but calculating area tedious

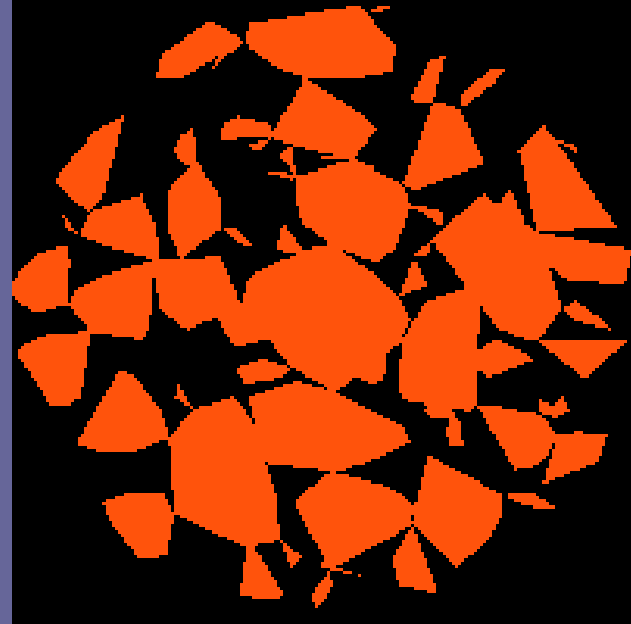
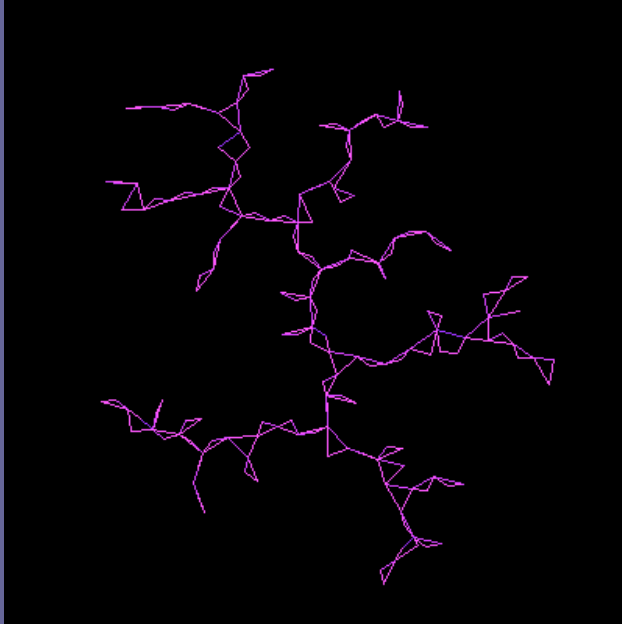
Convex Hulls

- Want an efficient algorithm to determine reachable area.
- Motivated to take the convex hull of the set of points in the RRT



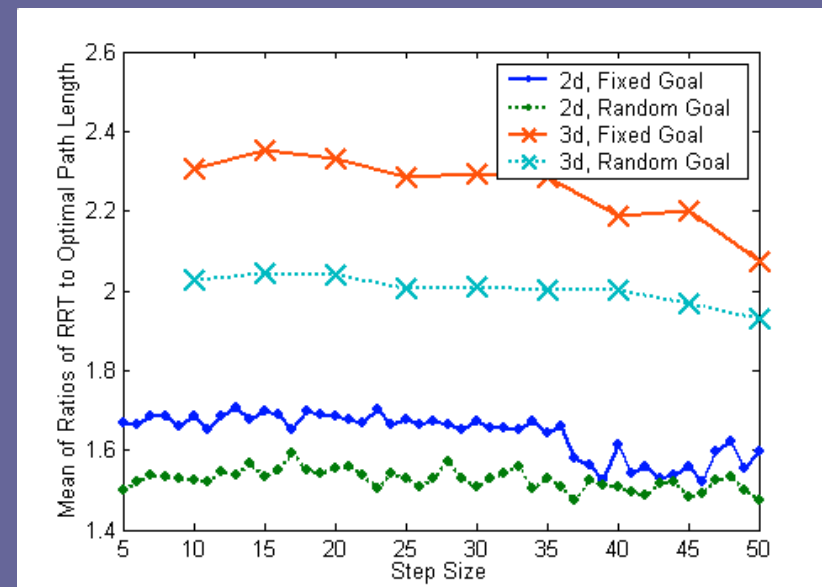
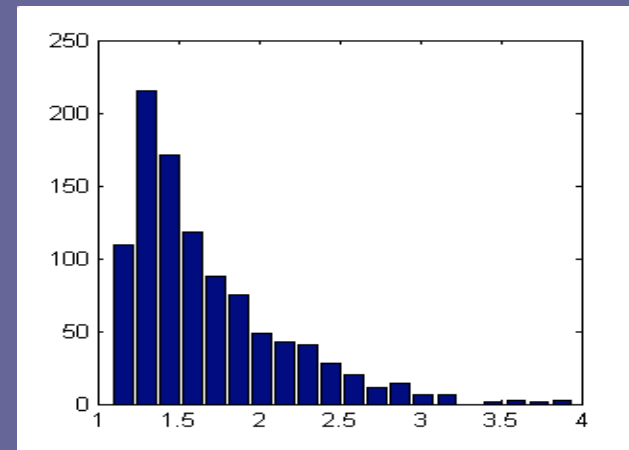
Arbitrary Precision Hull

- Similar to minimal-area hull, we wanted to compute the hull in a non-convex manner
- Group points based on depth and parent, create divisions every D deep in the tree



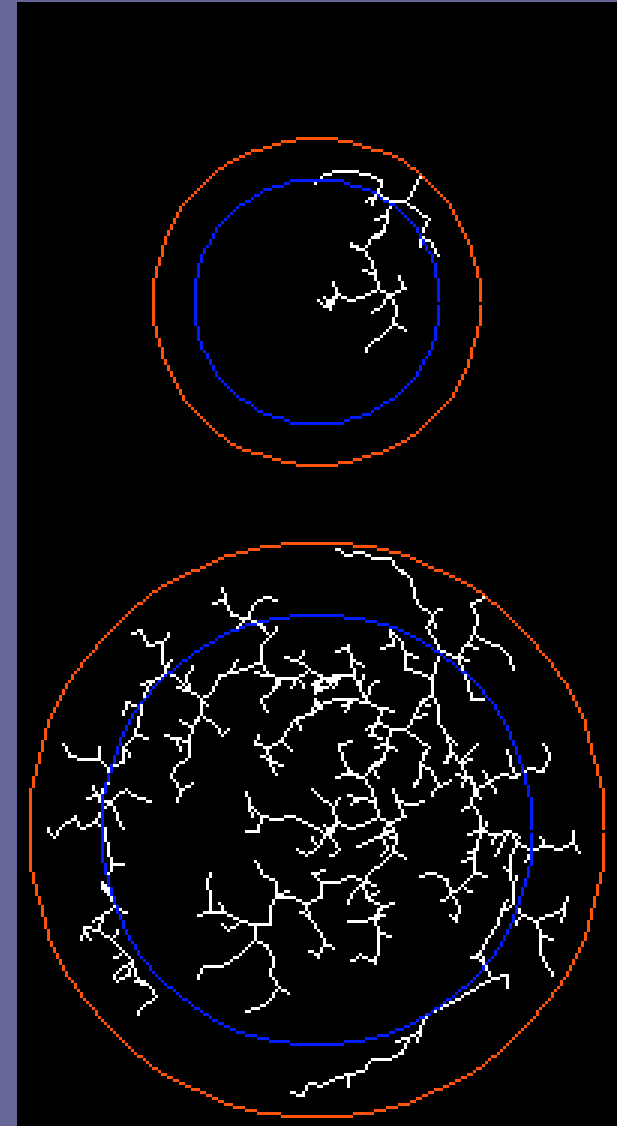
Optimal Solutions

- Question proposed: “How far off from optimal is the path an RRT finds?”
- Devised an experiment where we compared the ratio of the path the RRT found to the optimal path
- Tested with varied step size, fixed and random goal, and in 2d and 3d



Bounded RRTs

- Nearest neighbor calculation a bottleneck of the RRT
- Created a variant based on metric trees, introduced by Uhlmann (1991)
- Algorithm partitions the point set, so only a portion are queried by the nearest neighbor calculation



Conclusions

- Sampling-based planning is an effective technique for studying hybrid systems
- Developed a state-independent framework to run RRTs
- Extended the MSL to develop a tool to apply the RRT to hybrid problems
- Experimented with stair climbers in MSL
 - Biased searches toward a subgoal set

Conclusions (Cont.)

- Experimented with Rectangular HA with the MSL
 - Developed multi-action growth RRT
- Investigated reachable area of RRT
 - Arbitrary precision hull
- Studied the optimality of RRT paths
- Developed bounded RRTs

Future Work

- Nonlinear Hybrid Automata
 - RRTs applicable to nonholonomic problems
- Maneuver Automata
 - Model motion by a set of fixed maneuvers
- MSL Development – multi-state visualization
- Metric Functions – instead of Euclidean + k -factor
- Random State Selection
 - More intelligent biases
- Multiple RRTs – one in each state or transition?

Acknowledgements

- Dr. Michael Branicky, advisor
- Dr. Cenk Çavusoglu & Dr. Gultekin Ozsoyoglu, committee
- Dr. Steve LaValle et al., MSL & RRT
- Stuart Morgan