Transition-based RRT for Path Planning in Continuous Cost Spaces
(Jaillet, Cortés, and Siméon, 2008)

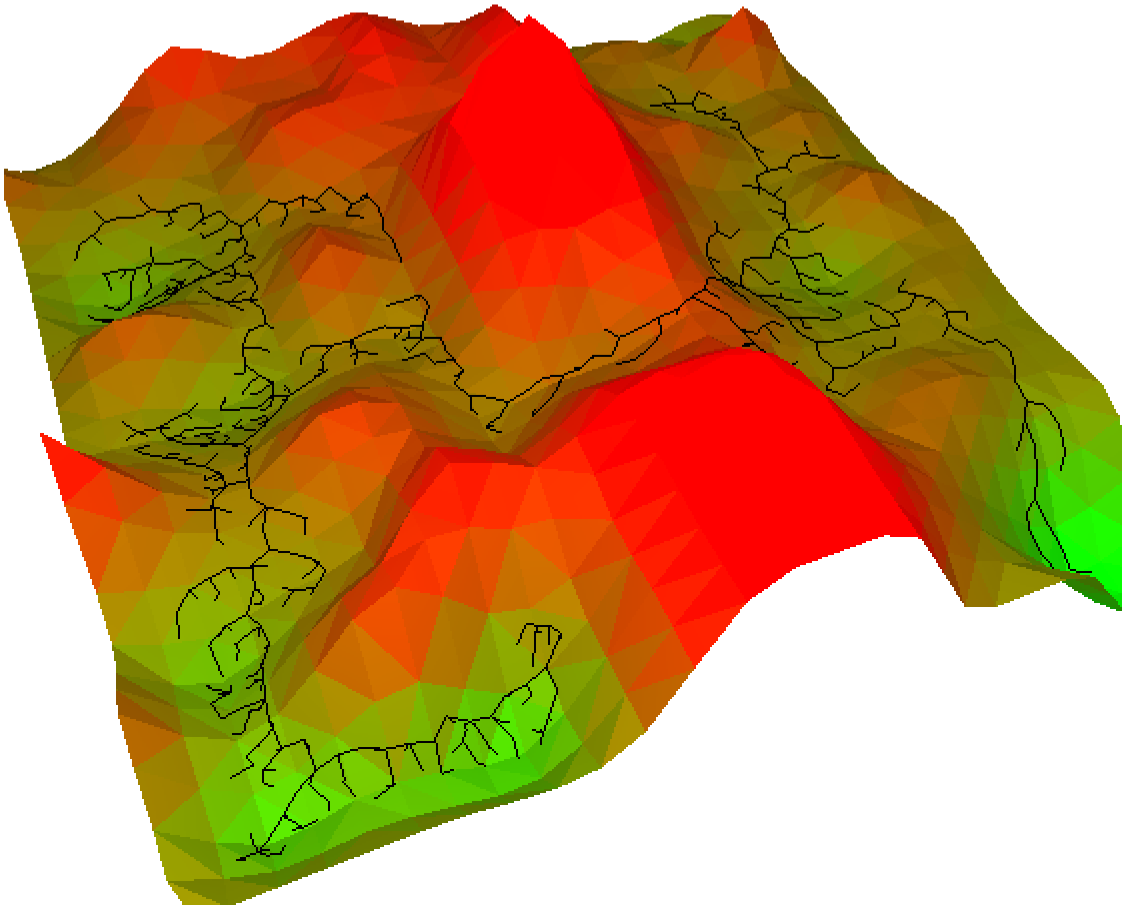
Presented by: Steven Clark

GMU CS689, Spring 2012

Outline

- Intro to T-RRT
  - Motivation: what problem are we trying to solve?
- Important algorithm components
  - *TransitionTest*
  - *MinExpandControl*
- Measuring & comparing path costs
- Results for various problems

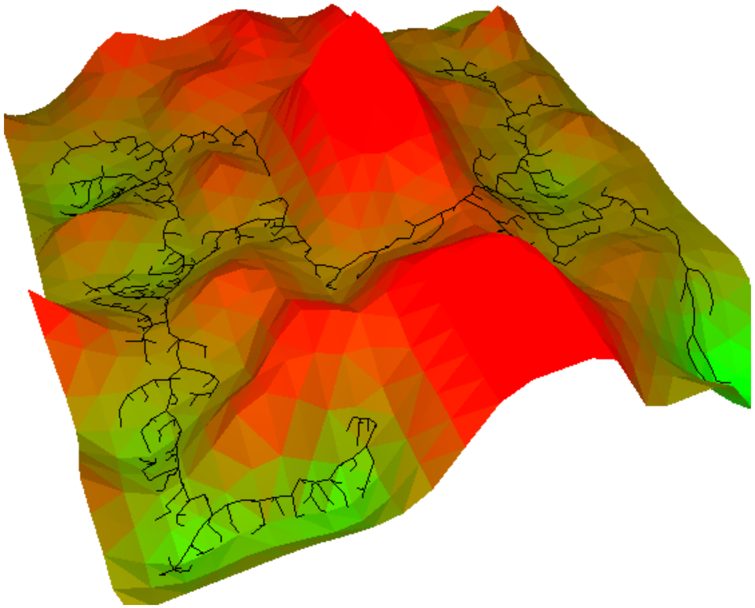
Intro to T-RRT



Exploring a “continuous cost space”

Algorithm Goals

- Efficiently explore the configuration space
- Stick to low-cost "valleys" when possible
- Don't get blocked by saddle-points



Algorithm Goals

- Try to combine:
  - Exploration strength of RRTs
  - Efficiency of stochastic optimization methods
    - (use transition tests to decide when to accept a new state)

Algorithm Pseudo-Code

Algorithm 1: Transition-based RRT

input : the configuration space CS;
the cost function $c : CS \rightarrow \mathbb{R}_*^+$;
the root q_{init} and the goal q_{goal};

output : the tree $\mathcal{T}$;

begin

```
    T ← InitTree(q_init);
    while not StopCondition(T, q_goal) do
        q_rand ← SampleConf(CS) ;
        q_near ← BestNeighbor(q_rand, T);
        if not Extend(T, q_rand, q_near, q_new) Continue;
        if TransitionTest(c(q_near), c(q_new), d_near-new)
        and MinExpandControl(T, q_near, q_rand) then
            AddNewNode(T, q_new);
            AddNewEdge(T, q_near, q_new);
```

end

Transition Test

- Probability of accepting a new configuration c_j:

Slope of cost between 2 configs

$$p_{ij} = \begin{cases} exp(-\frac{\Delta c^*_{ij}}{K * T}) & \text{if } \Delta c^*_{ij} > 0 \\ 1 & \text{otherwise.} \end{cases}$$

Always go to a lower-cost state

  - AKA “Metropolis criterion”
- K: normalizing constant
- T: “temperature” parameter
  - Controls difficulty of transition tests
  - Will change over time!

TransitionTest()

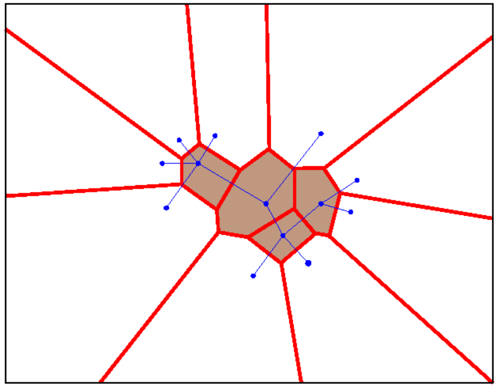
Algorithm 2: TransitionTest(c_i, c_j,d_{ij})

```
begin
    if c_j > c_max then return False;
    if c_j < c_i then return True;
    p = exp(-(c_j - c_i)/d_ij / (K*T));
    if Rand(0,1) < p then
        T = T/α;
        nFail = 0;
        return True;
    else
        if nFail > nFail_max then
            T = T*α;
            nFail = 0;
        else
            nFail = nFail + 1;
        return False;
end
```

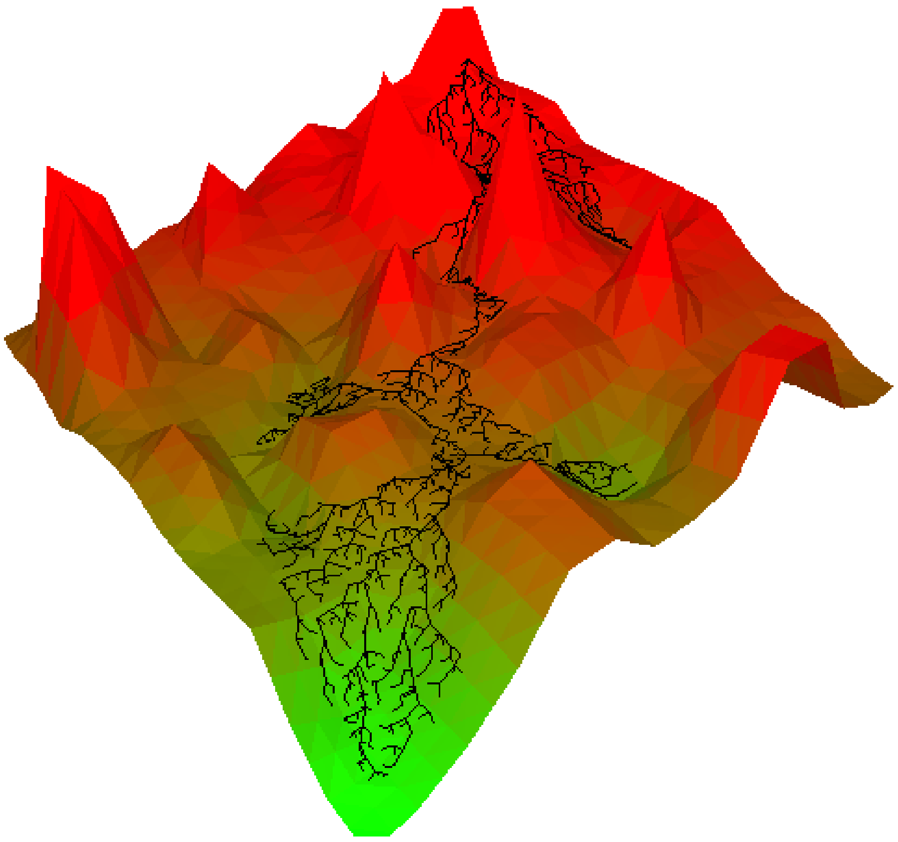
T is self-tuning!

MinExpandControl()

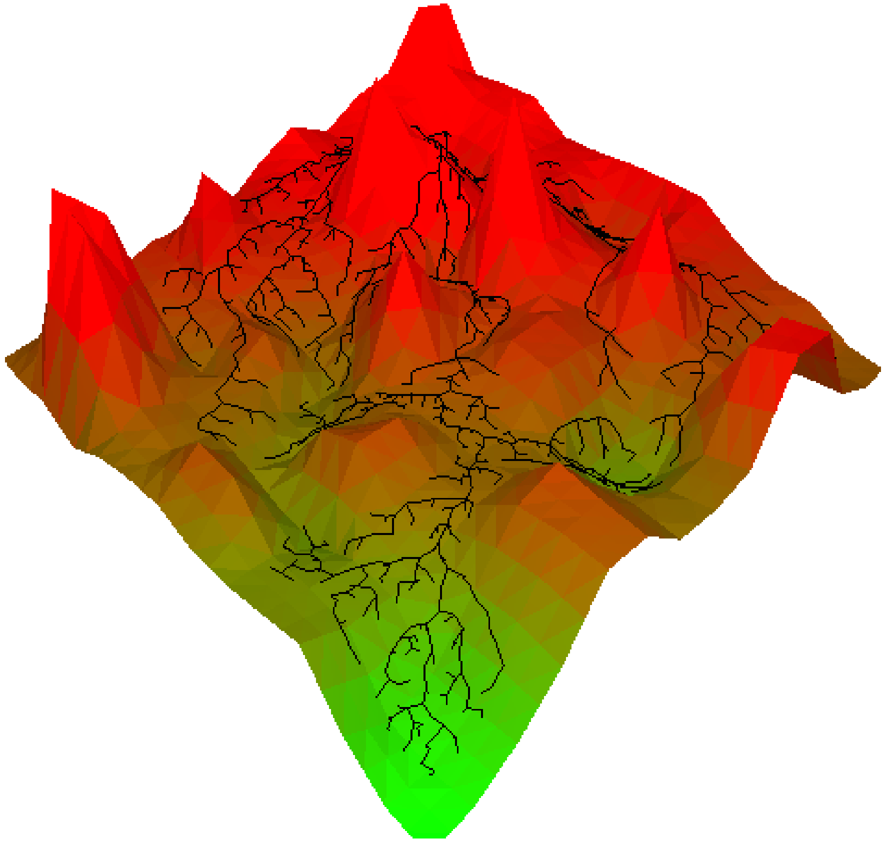
- Can classify a new state as either expansion or refinement, based on distance to nearest existing state
- MinExpandControl ensures a proper balance of expansion to refinement
  - Reject some states that are accepted by TransitionTest



Effect of MinExpandControl()



Without MinExpandControl



With MinExpandControl

Measuring Performance

- Analogous to mechanical work
  - Measure "energy" lost to work
- Work done along a continuous path:

$$W(\mathcal{P}) = \int_{s_+} \frac{\partial c_+}{\partial s} ds + \epsilon \int_s ds$$

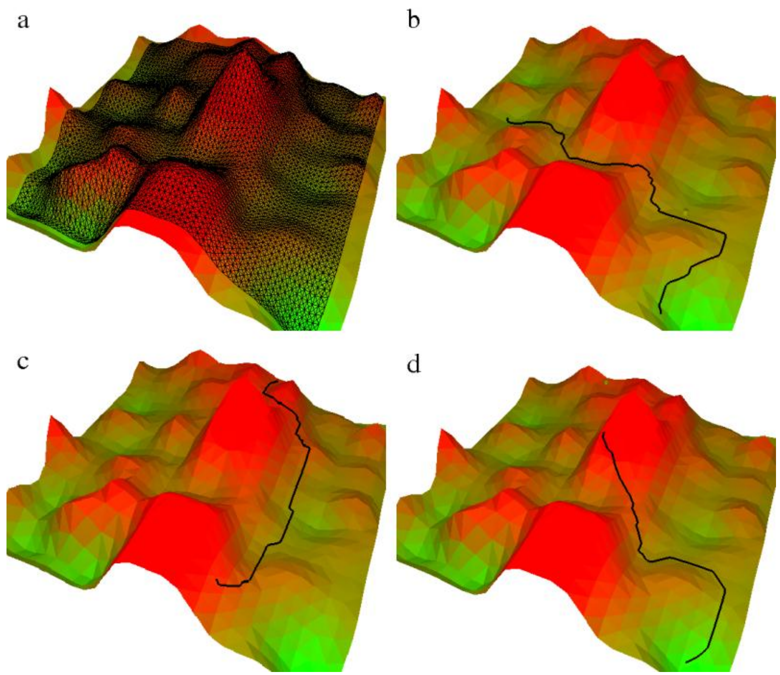
Path segments with positive slope

- Work done along a discretized path:

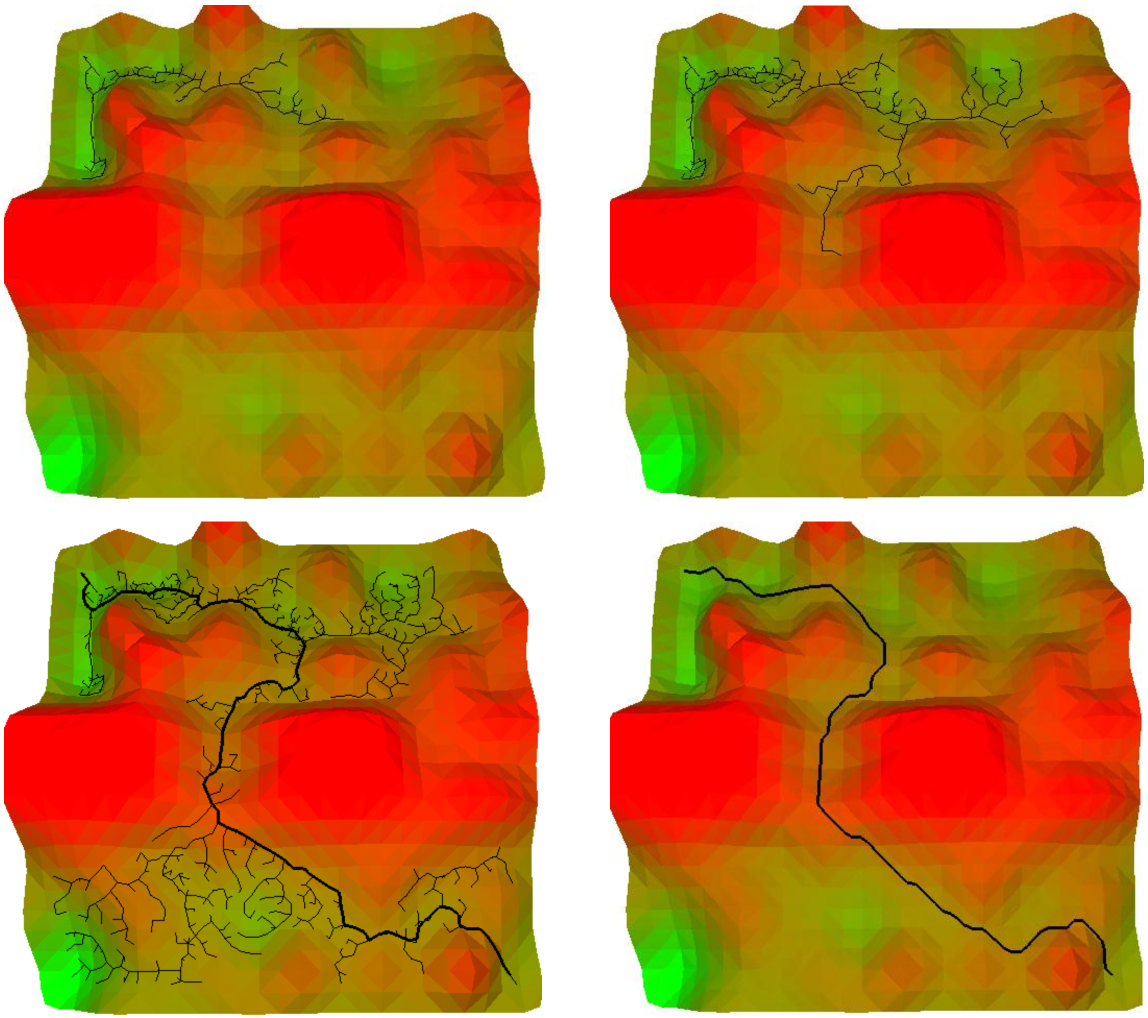
$$W(\mathcal{P}) = \sum_{i \in i_+} \left(c(q_i) - c(q_{i-1})\right) d_i + \epsilon \sum_{i \in i} d_i$$

Calculating Optimal Paths

- Using this approach, can calculate optimal paths by discretizing space and using A*
- Note: only a reference, not feasible in higher dimensions



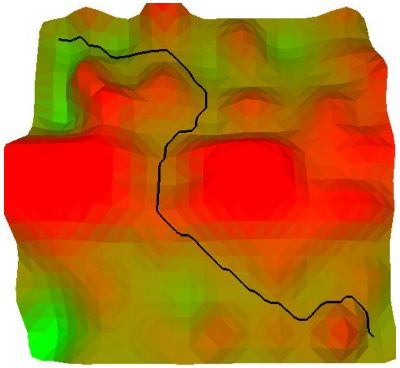
T-RRT Path vs. Optimal Path



Optimal path

RRT vs. T-RRT vs. Optimal

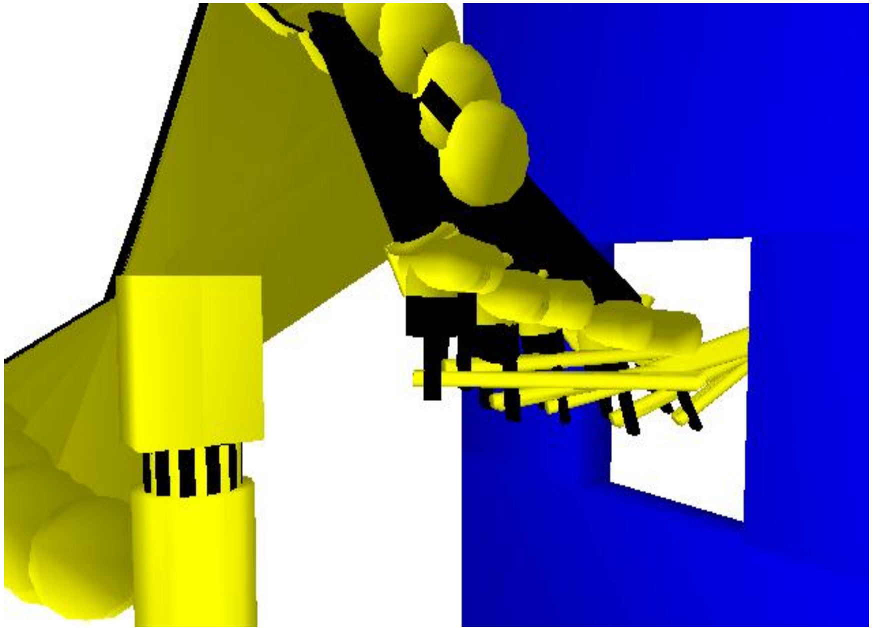
	$length$	c_{min}	c_{max}	c_{ave}	W
RRT	148	8	36	21	**32.7**
T-RRT	214	8	23	17	**19.5**
T-RRT$_{10}$	182	8	25	18	**21.9**
$\mathcal{P}^*$	178	10	23	17	**13.3**



- RRT is 2.5x worse than optimal in this scenario
- T-RRT is only 45% worse than optimal

Other Examples

- 6 DOF manipulator, extracting rod through cluttered environment
  - Cost is related to how close rod gets to obstacles
- T-RRT path is much lower work and is much more reliable

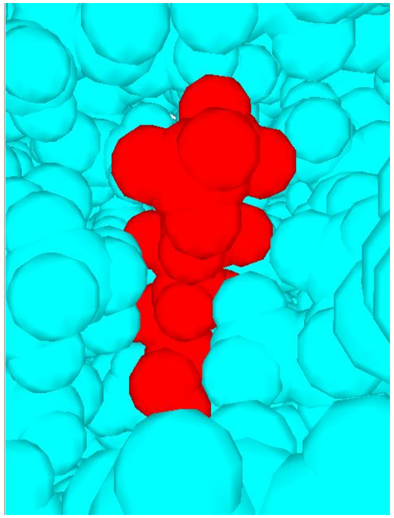


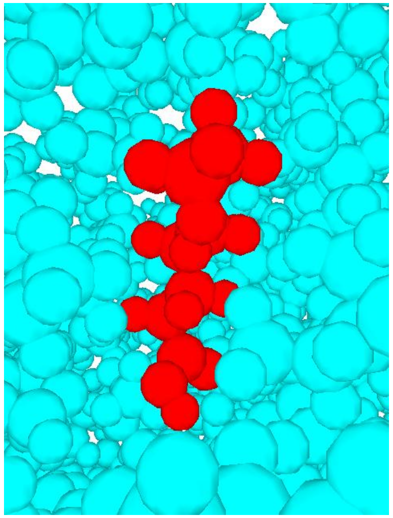
	$length$	c_{min}	c_{max}	c_{ave}	W	σ_W
RRT	45	0.1	17150	88	**3279**	3151
T-RRT	49	0.1	40.4	1	**15.2**	5.22

Other Examples

- Extracting a ligand molecule from a protein
  – Cost is related to closeness of protein & ligand
- T-RRT path is much lower work and has more clearance

	$length$	c_{min}	c_{max}	c_{ave}	W	VdW
RRT	59	0.1	2236	14	**471**	0.25
T-RRT	70	0.1	1.0	0.3	**0.3**	0.66





Conclusions

- T-RRT combines strengths of RRTs for exploration & stochastic transition tests
- Has self-tuning parameters
- General enough to apply to high-dimensional problems
- Excellent results vs. traditional RRT paths