Path Planning: An Application of the Static Hamilton Jacobi Equation

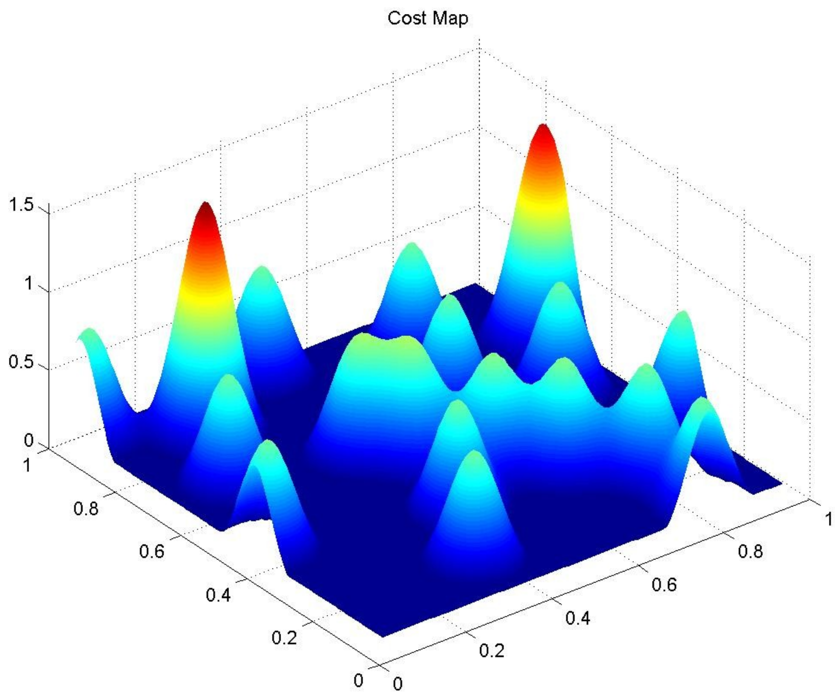
Ian Mitchell
Department of Computer Science
University of British Columbia

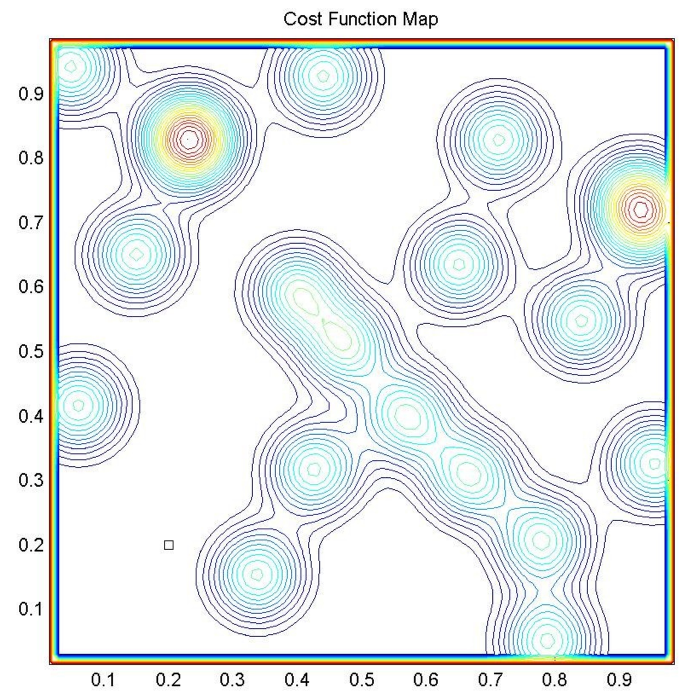
Joint work with
Ken Alton (UBC)

research supported by
the Natural Science and Engineering Research Council of Canada

Basic Path Planning

- Find the optimal path $p(s)$ to a target (or from a source)
  – No constraints on the path
- Problem data
  – Cost $c(x)$ to pass through each state in the state space
  – Set of targets or sources (provides boundary conditions)





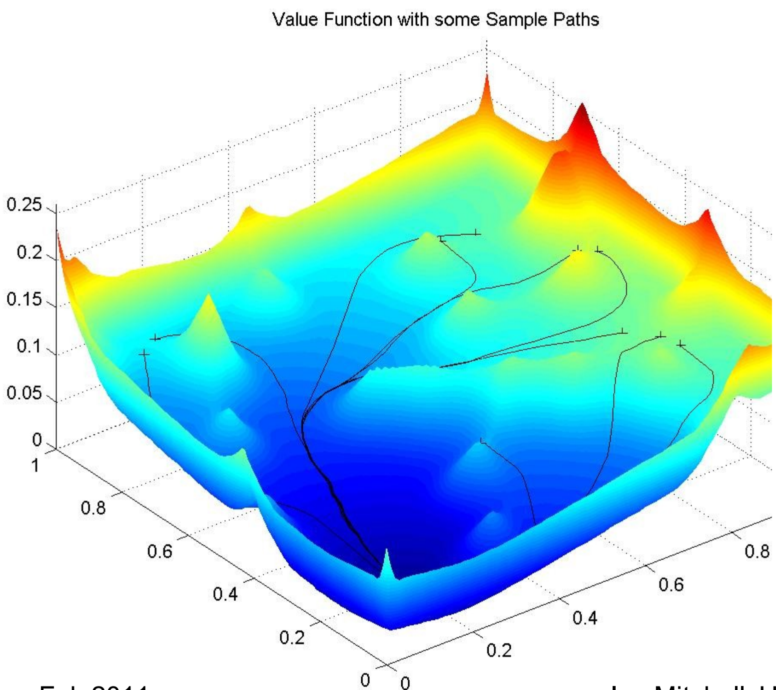
Value Function for Path Planning

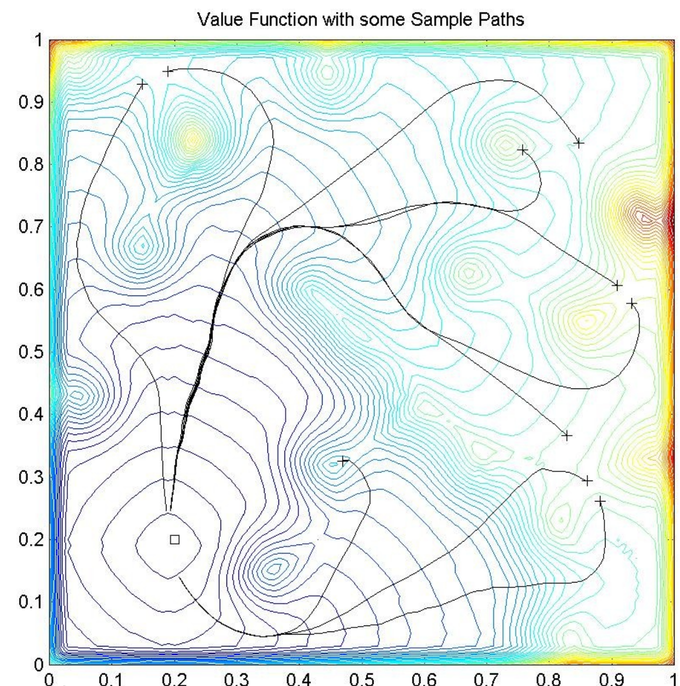
- Value function solves Eikonal equation

$$\|D_x V(x)\| = c(x)$$

- Optimal paths found by gradient descent

$$\frac{d}{ds}p(s) = \frac{D_x V(x)}{\|D_x V(x)\|}$$





Continuous Dynamic Programming

- Continuous DPP for path $p(\cdot)$

$$\vartheta(p(s)) = \min_{p(\cdot)} \left[\vartheta(p(s + \Delta s)) + \int_s^{s+\Delta s} c(p(\sigma)) d\sigma \right]$$

- Rearrange

$$\min_{p(\cdot)} \left[\frac{\vartheta(p(s)) - \vartheta(p(s + \Delta s))}{\Delta s} - \frac{\int_s^{s+\Delta s} c(p(\sigma)) d\sigma}{\Delta s} \right] = 0$$

- Take limit $\Delta s \to 0$

$$\min_{p(\cdot)} \left[-\frac{d\vartheta(p(s))}{ds} - c(p(s)) \right] = 0$$

- Set $x = p(s)$ and chain rule

$$\min_{p(\cdot)} \left[\frac{\partial \vartheta(x)}{\partial x} \frac{dp(s)}{ds} + c(x) \right] = 0$$

Static Hamilton-Jacobi PDE

- Let control be $u(s) = \frac{dp(s)}{ds}$ and observe that the only dependence on $p(\cdot)$ is u to arrive at the static HJ PDE

$$\min_u \left[D_x\vartheta(x) \cdot u + c(x)\right] = H(x, D_x\vartheta(x)) = 0$$

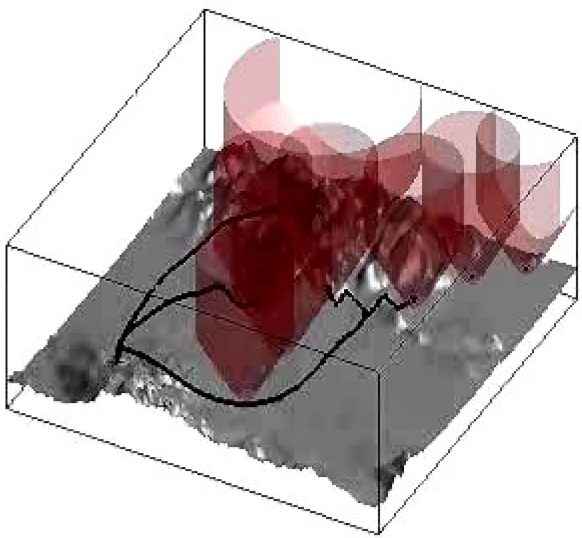
- From original problem for $x \in \mathcal{T}$ we get boundary conditions $\vartheta(x) = 0$
- If constraint on u is isotropic (eg: $\|u\|_2 \leq 1$), choose optimal control

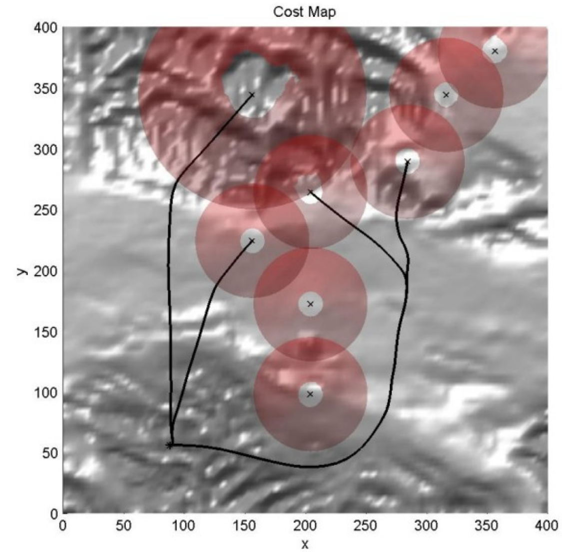
$$u(s) = \frac{D_x\vartheta(x)}{\|D_x\vartheta(x)\|_2}$$

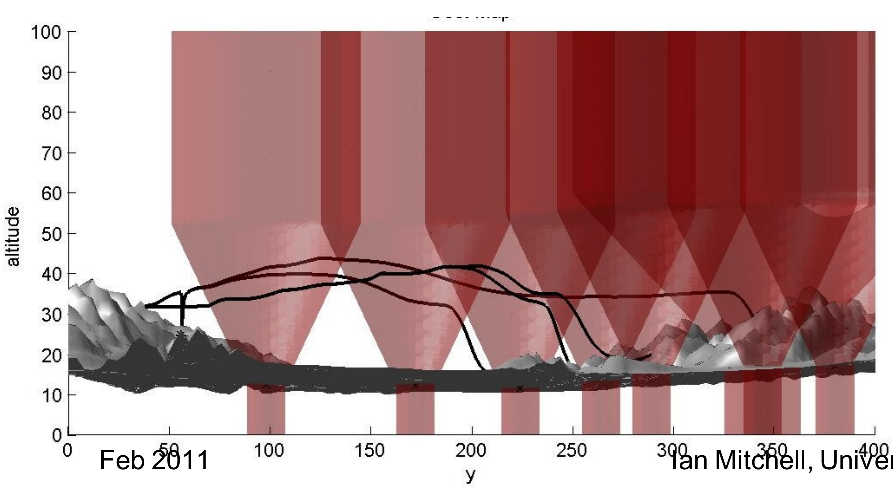
and PDE becomes the Eikonal equation

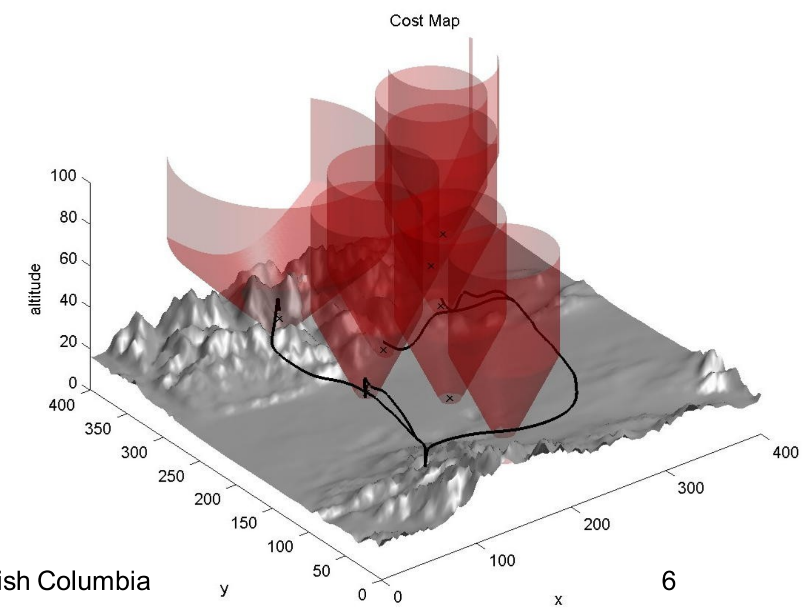
$$\begin{aligned} \|D_x\vartheta(x)\|_2 &= c(x) \quad \text{for } x \in \mathbb{R}^2 \setminus \mathcal{T} \\ \vartheta(x) &= 0 \qquad \text{for } x \in \partial\mathcal{T} \end{aligned}$$

Demanding Example? No!





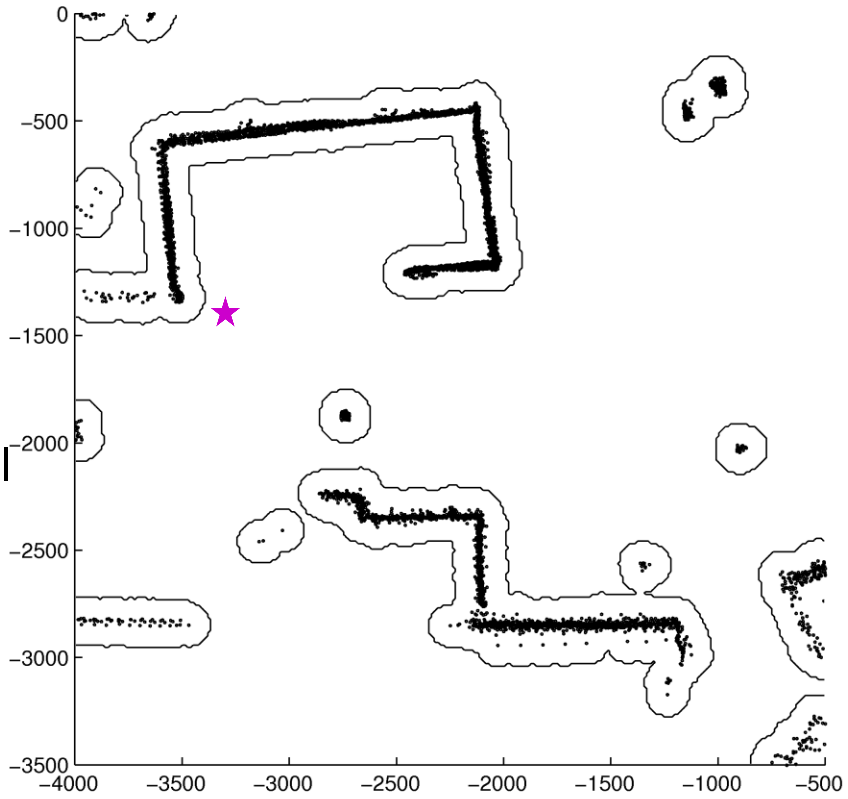




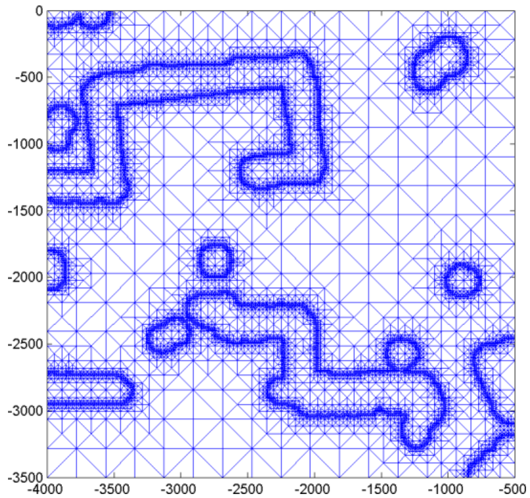
Robot Path Planning

- Find shortest path to objective while avoiding obstacles
  - Obstacle maps from laser scanner
  - Configuration space accounts for robot shape
  - Cost function essentially binary
- Value function measures cost to go
  - Solution of Eikonal equation
  - Gradient determines optimal control

typical laser scan with configuration space obstacles



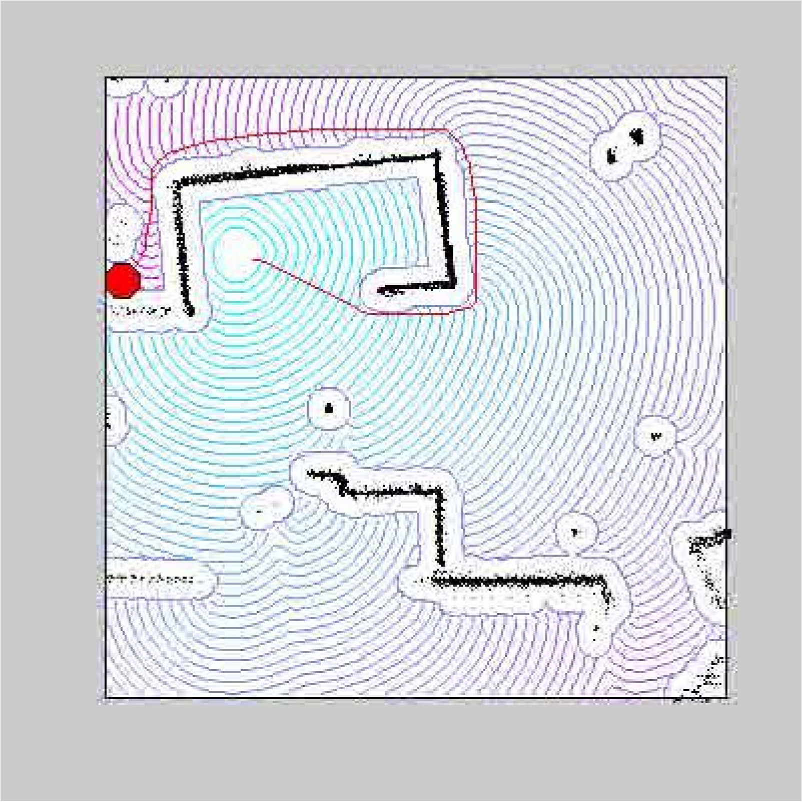
adaptive grid



Alton & Mitchell, "Optimal Path Planning under Different Norms in Continuous State Spaces," ICRA 2006

Continuous Value Function Approximation

- Contours are value function
  - Constant unit cost in free space, very high cost near obstacles
- Gradient descent to generate the path



Hamilton-Jacobi Flavours

- Time-dependent Hamilton-Jacobi used for dynamic implicit surfaces and finite horizon optimal control / differential games

$$D_t\phi(x,t) + H(x, D_x\phi(x,t)) = 0$$

  - Solution continuous but not necessarily differentiable
  - Time stepping approximation with high order accurate schemes
  - Numerical schemes have conservation law analogues
- Stationary (static) Hamilton-Jacobi used for target based cost to go and time to reach problems

$$H(x, D_x\vartheta(x)) = 0 \qquad \|D_x\vartheta(x)\| = c(x)$$

  - Solution may be discontinuous
  - Many competing algorithms, variety of speed & accuracy
  - Numerical schemes use characteristics (trajectories) of solution

Solving Static HJ PDEs

- Two methods available for using time-dependent techniques to solve the static problem
  - Iterate time-dependent version until Hamiltonian is zero
  - Transform into a front propagation problem
- Schemes designed specifically for static HJ PDEs are essentially continuous versions of value iteration from dynamic programming
  - Approximate the value at each node in terms of the values at its neighbours (in a consistent manner)
  - Details of this process define the “local update”
  - Eulerian schemes, plus a variety of semi-Lagrangian
- Result is a collection of coupled nonlinear equations for the values of all nodes in terms of all the other nodes
- Two value iteration methods for solving this collection of equations: marching and sweeping
  - Correspond to label setting and label correcting in graph algorithms

Cost Depends on…

- So far assumed that cost depends only on position
- More generally, cost could depend on position and direction of motion (eg action / input)
  - Variable dependence on position: inhomogenous cost
  - Variable dependence on direction: anisotropic cost
- Discrete graph
  - Cost is associated with edges instead of nodes
  - Dijkstra's algorithm is essentially unchanged
- Continuous space
  - Static HJ PDE no longer reduces to the Eikonal equation

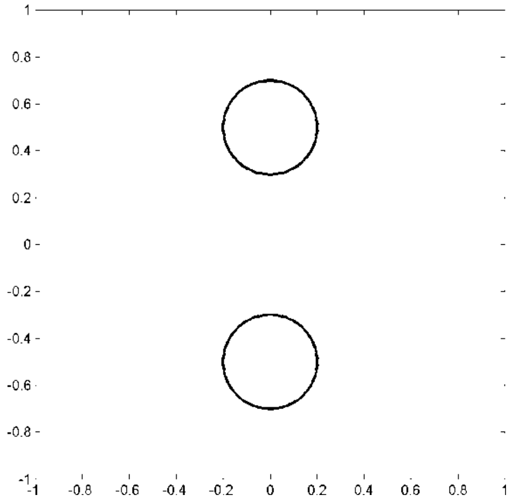
$$\min_{u \in U} \left[D_x \vartheta(x) \cdot u + c(x) \right] = 0 \quad \not\Leftrightarrow \quad \| D_x \vartheta(x) \| = c(x)$$

$$\text{when } U \text{ is not a circle / sphere}$$

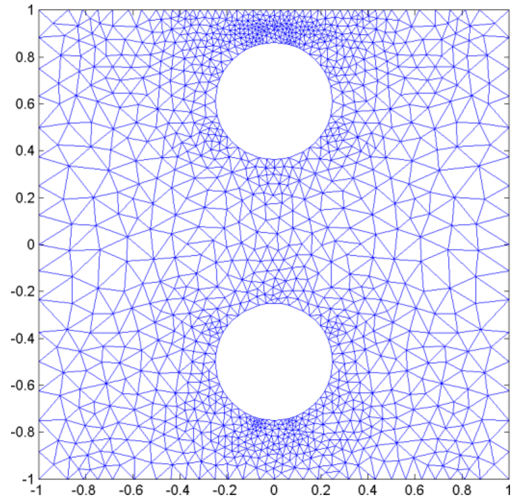
  - Gradient of ϑ may not be the optimal direction of motion
  - Isotropy is related to but stronger than holonomicity or small time local controllability

Other Static HJ Issues: Obstacles

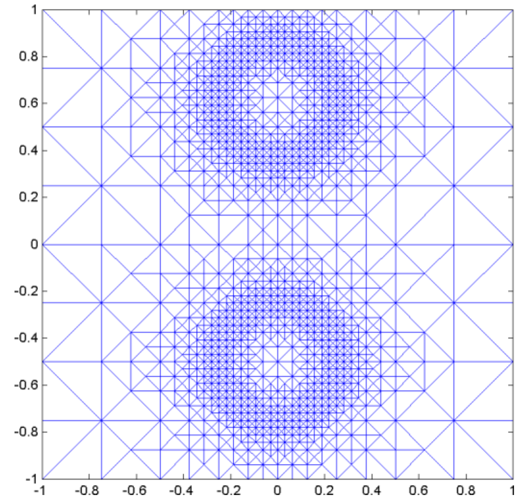
- Computational domain should not include (hard) obstacles
  - Requires “body-fitted” and often non-acute grid: straightforward in 2D, challenging in 3D, open problem in 4D+
- Alternative is to give nodes inside the obstacle a very high cost
  - Side effect: the obstacle boundary is blurred by interpolation
- Improved resolution around obstacles is possible with semi-structured adaptive meshes
  - Not trivial in higher dimensions; acute meshes may not be possible



original obstacles



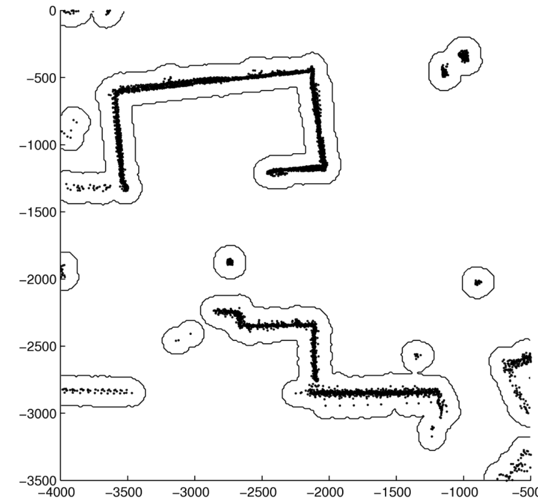
body fitted mesh



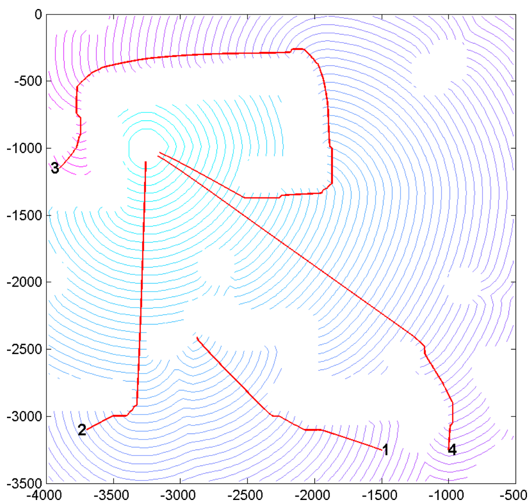
semi-structured mesh

Adaptive Meshing is Practically Important

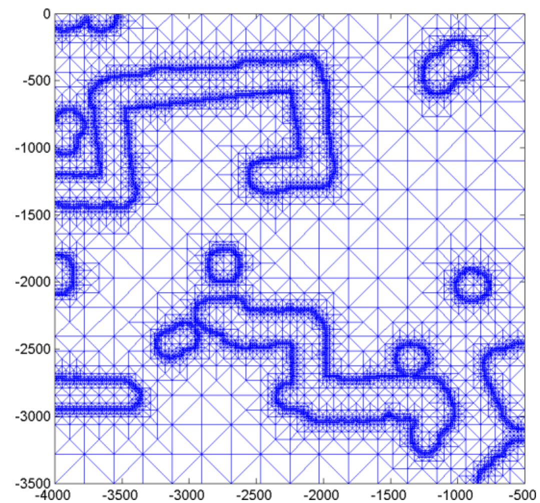
- Much of the static HJ literature involves only 2D and/or fixed Cartesian meshes with square aspect ratios
  - “Extension to variably spaced or unstructured meshes is straightforward…”
- Nontrivial path planning demands adaptive meshes
  - And configuration space meshing, and dynamic meshing, and …



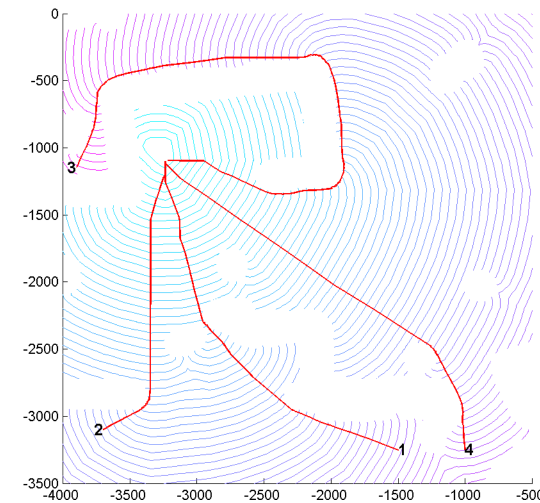
original obstacles



Cartesian mesh’s paths



adaptive mesh



adaptive mesh’s paths

Methods: Direct Time-Dependent Version

$$H(x, D_x\vartheta(x)) = 0 \text{ for } x \in \Omega \setminus \mathcal{T}$$
$$\vartheta(x) = 0 \text{ for } x \in \partial\mathcal{T}$$

- Time-dependent version: replace $\vartheta(x) \to \vartheta(t,x)$ and add temporal derivative

$$D_t\vartheta(t, x) + H(x, D_x\vartheta(t, x)) = 0$$

  - Solve until $D_t\vartheta(t,x) = 0$, so that $\vartheta(t,x) = \vartheta(x)$
- Not a good idea
  - No reason to believe that $D_t\vartheta(t,x) \to 0$ in general
  - In limit $t \to \infty$, there is no guarantee that $\vartheta(t,x)$ remains continuous, so numerical methods may fail

Transform Static to Time-Dependent HJ

Create implicit surface definition of T

$$\phi(x,0) \begin{cases} \leq 0, x \in T; \\ = 0, x \in \partial T; \\ \geq 0, x \in \mathbb{R}^d \setminus T. \end{cases}$$

Under assumption $D_x\phi(x,0) \cdot p \neq 0$ on ∂T, make change of variables

$$D_x\vartheta(x) \leftarrow \frac{D_x\phi(x,t)}{D_t\phi(x,t)}$$

and get toolbox appropriate PDE

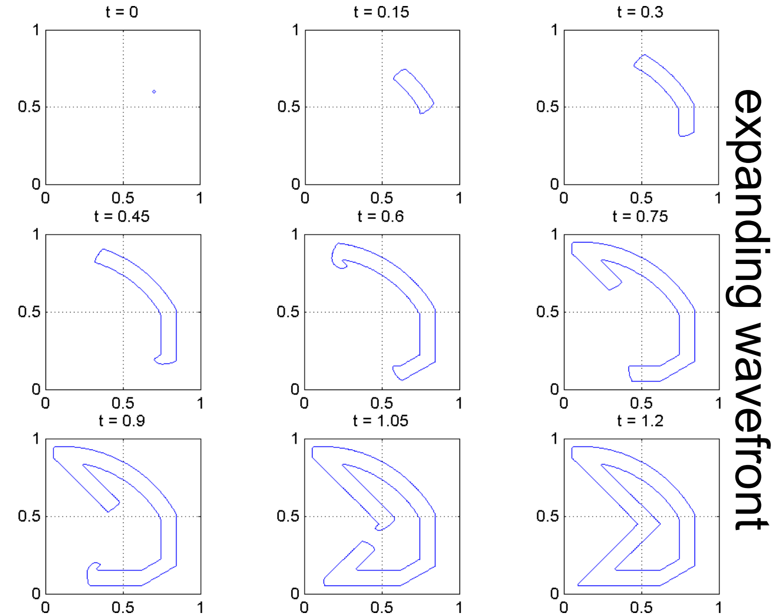
$$D_t\phi(x,t) + \min_{p \in \mathbb{S}^1} \frac{D_x\phi(x,t) \cdot p}{\ell(x,p)} = 0.$$

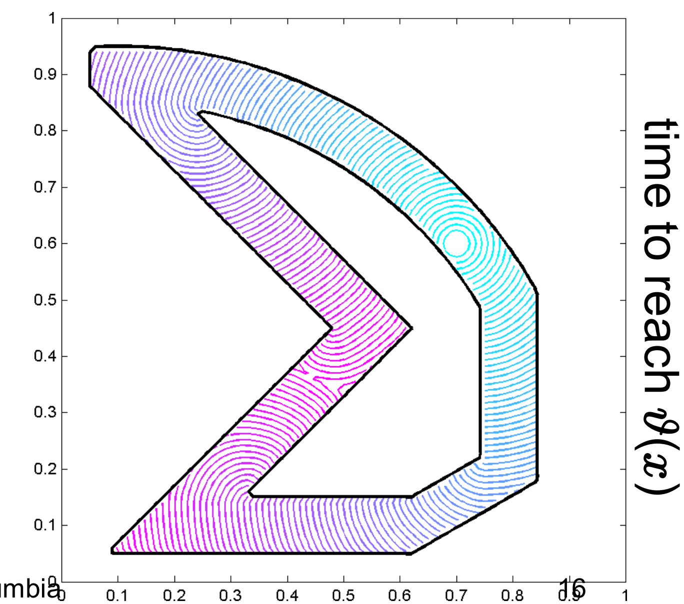
After solving, set ϑ to be crossing time

$$\vartheta(x) = \{t \mid \phi(x,t) = 0\}.$$

Methods: Time-Dependent Transform

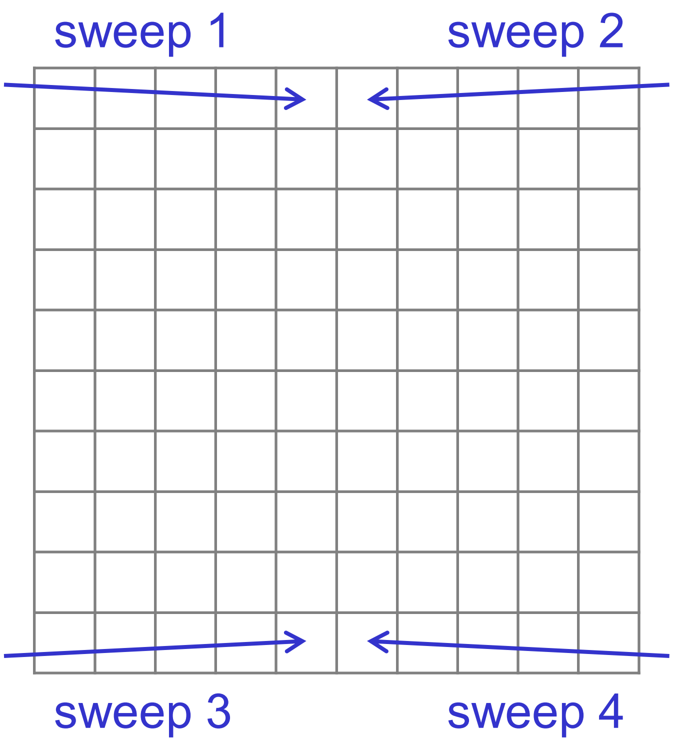
- Equivalent wavefront propagation problem [Osher 93]
- Pros:
  - Implicit surface function for wavefront is always continuous
  - Handles anisotropy
  - High order accuracy schemes available on uniform Cartesian grid
  - Subgrid resolution of obstacles through implicit surface representation
  - ToolboxLS code is available
- Cons:
  - CFL requires many timesteps
  - Computation over entire grid at each timestep





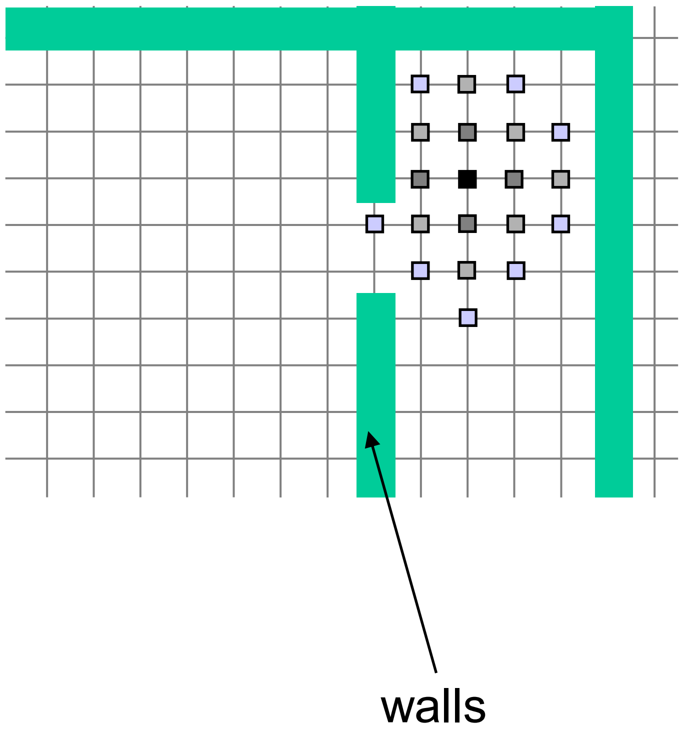
Methods: Fast Sweeping

- Gauss-Seidel iteration through the grid
  - For a particular node, use a consistent update (same as fast marching)
  - Several different node orderings are used in the hope of quickly propagating information along characteristics
  - Zhao, Qian, Zhang, Tsai, Osher, Chang, Kao, …
- Pros:
  - Easy to implement
  - handles anisotropy, nonconvexity, obtuse unstructured grids
- Cons:
  - Multiple sweeps required for convergence



Methods: Fast Marching / Ordered Upwind

- Dijkstra's algorithm with a consistent node update formula
  - Tsitsiklis, Sethian, Kimmel, Vladimirsky, …
- Pros:
  - Efficient, single pass
  - Isotropic case relatively easy to implement
- Cons:
  - Random memory access pattern
  - No advantage from accurate initial guess
  - Requires causality relationship between node values
  - Anisotropic case trickier to implement



More General Anisotropic Cost / Speed

- Dirichlet problem for a static Hamilton-Jacobi PDE:

$$H(x, Du(x)) = 0, \qquad x \in \Omega$$

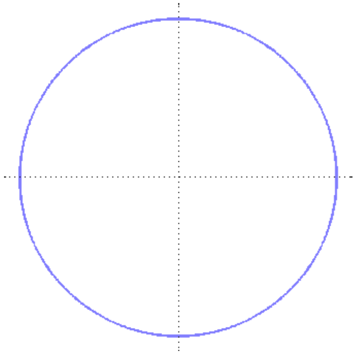
$$u(x) = g(x), \quad x \in \partial\Omega$$

- Control-theoretic Hamiltonian:

$$H(x, q) = \max_{a \in \mathcal{A}} [(-q \cdot a) f(x, a)] - 1$$

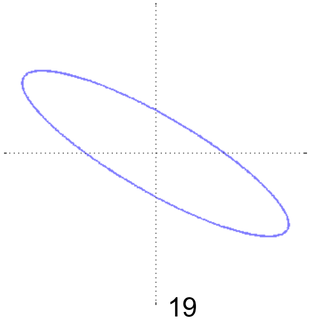
- Unit vector controls:

$$\mathcal{A} = \{a \in \mathbb{R}^d \mid \|a\| = 1\}$$



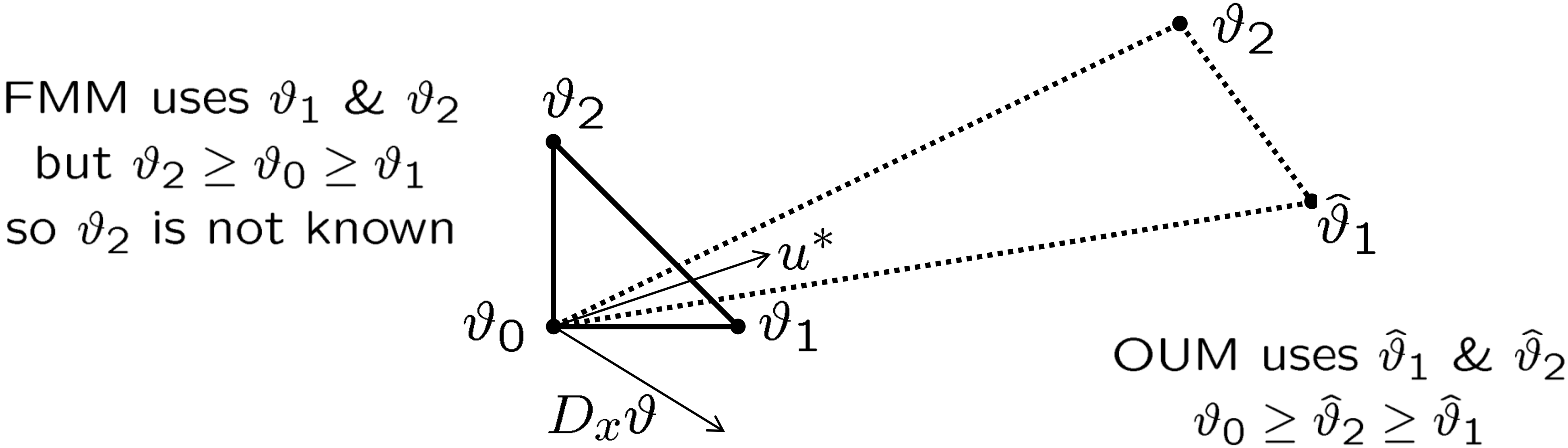
- Speed profile:

$$\mathcal{A}_f(x) = \{a f(x, a) \mid a \in \mathbb{R}^d \text{ and } \|a\| \leq 1\}$$



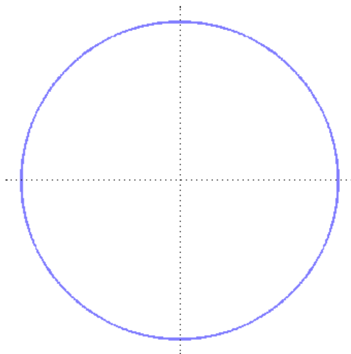
Anisotropy Leads to Causality Problems

- To compute the value at a node, we look back along the optimal trajectory ("characteristic"), which may not be the gradient
- Nodes in the simplex containing the characteristic may have value greater than the current node
  - Under Dijkstra's algorithm / FMM, only values less than the current node are known to be correct
- Ordered upwind extension of FMM searches a larger set of simplices to find one whose values are all known
- However, for some anisotropies and grids, regular FMM works

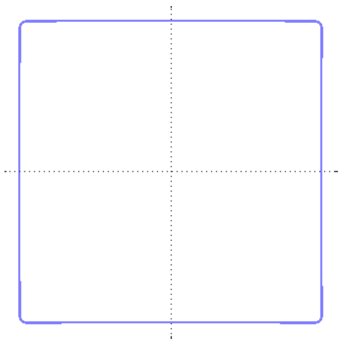


Speed Profiles

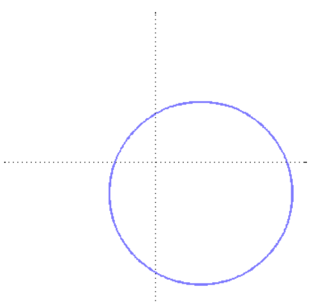
- To ensure continuity of the value function, origin must be in the interior (small time locally controllable)

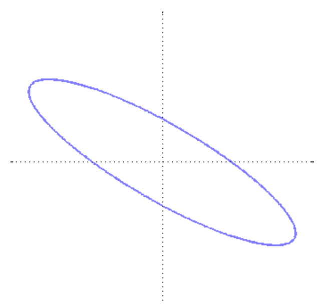


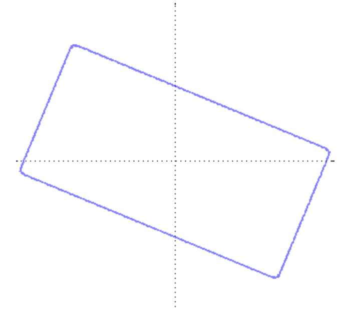
- To use Eikonal solvers, speed profile must be a circle / sphere at each point



- On an orthogonal grid, FMM will still work for axis-aligned anisotropies
- For more general anisotropies, OUM or fast sweeping methods are required



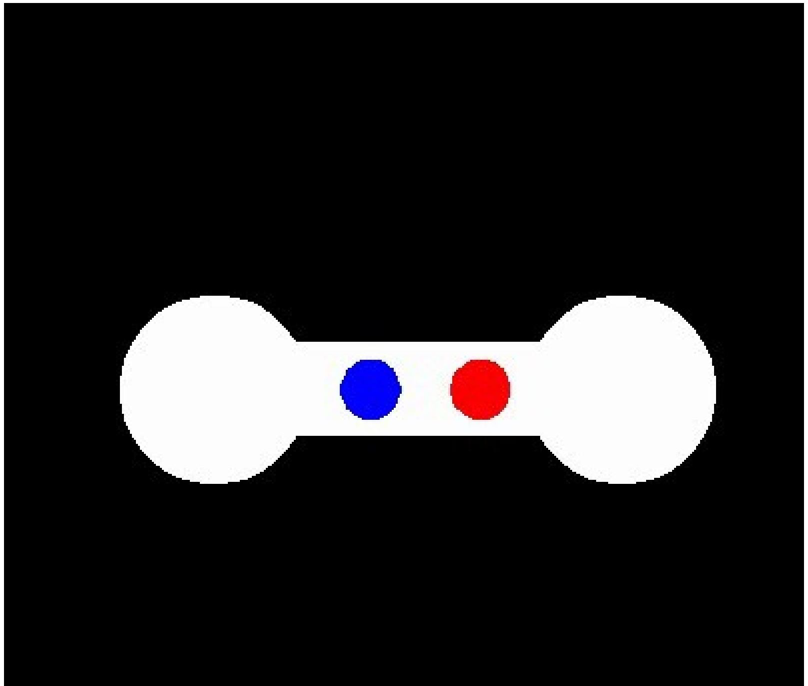




FMM for Axis-Aligned Anisotropies

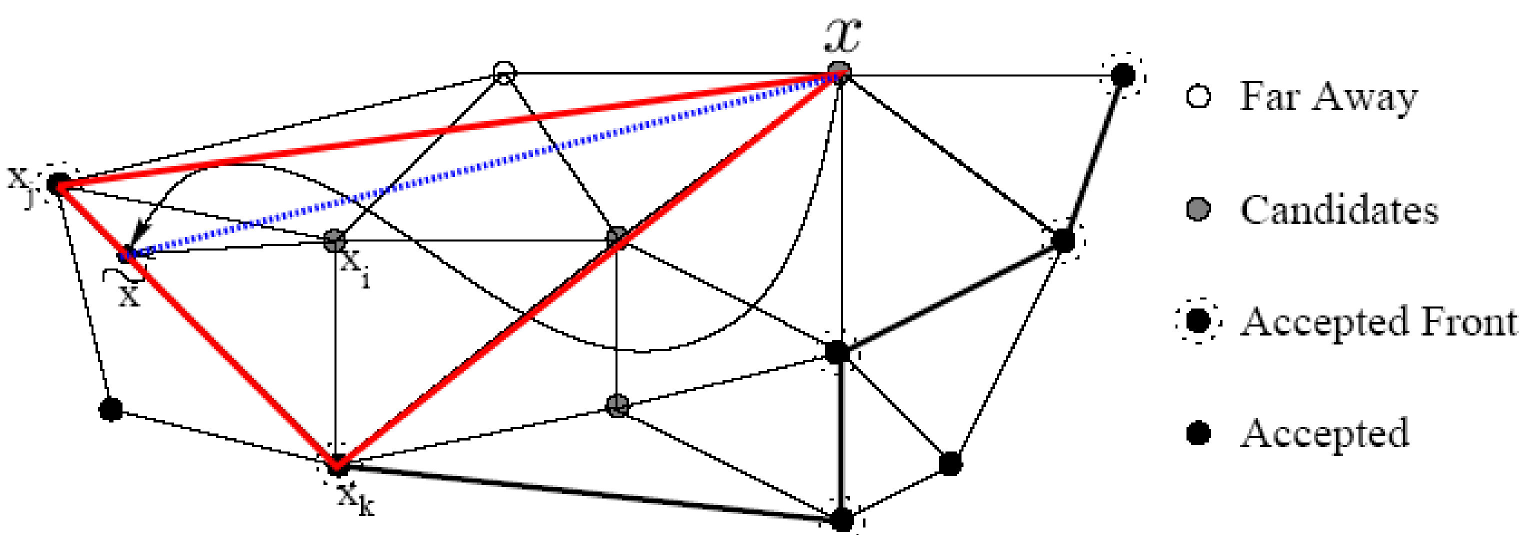
- FMM can be used on an orthogonal grid for Hamiltonians satisfying *strict one-sided monotonicity*
  - Related to "Osher's criterion" but does not require differentiability
- Alton & Mitchell, SINUM 2008
- Example: two robots moving in the plane

$$\left\| \left(\left\| \left(\frac{\partial \vartheta(x)}{\partial x_1}, \frac{\partial \vartheta(x)}{\partial x_2} \right) \right\|_2, \left(\frac{\partial \vartheta(x)}{\partial x_3}, \frac{\partial \vartheta(x)}{\partial x_4} \right) \right\|_2 \right\|_1 = c(x).$$



Ordered Upwind Method (OUM)

- Extension of FMM to solve problems with general convex speed profiles in O(N log N)
- Update() looks beyond immediate neighbors to use virtual simplices that include nodes within $h\Upsilon$,
  - Anisotropy coefficient Υ is ratio of fastest to slowest speed
- Search for such neighbours occurs only on the front of newly accepted nodes (Accepted Front OUM / AFOUM)

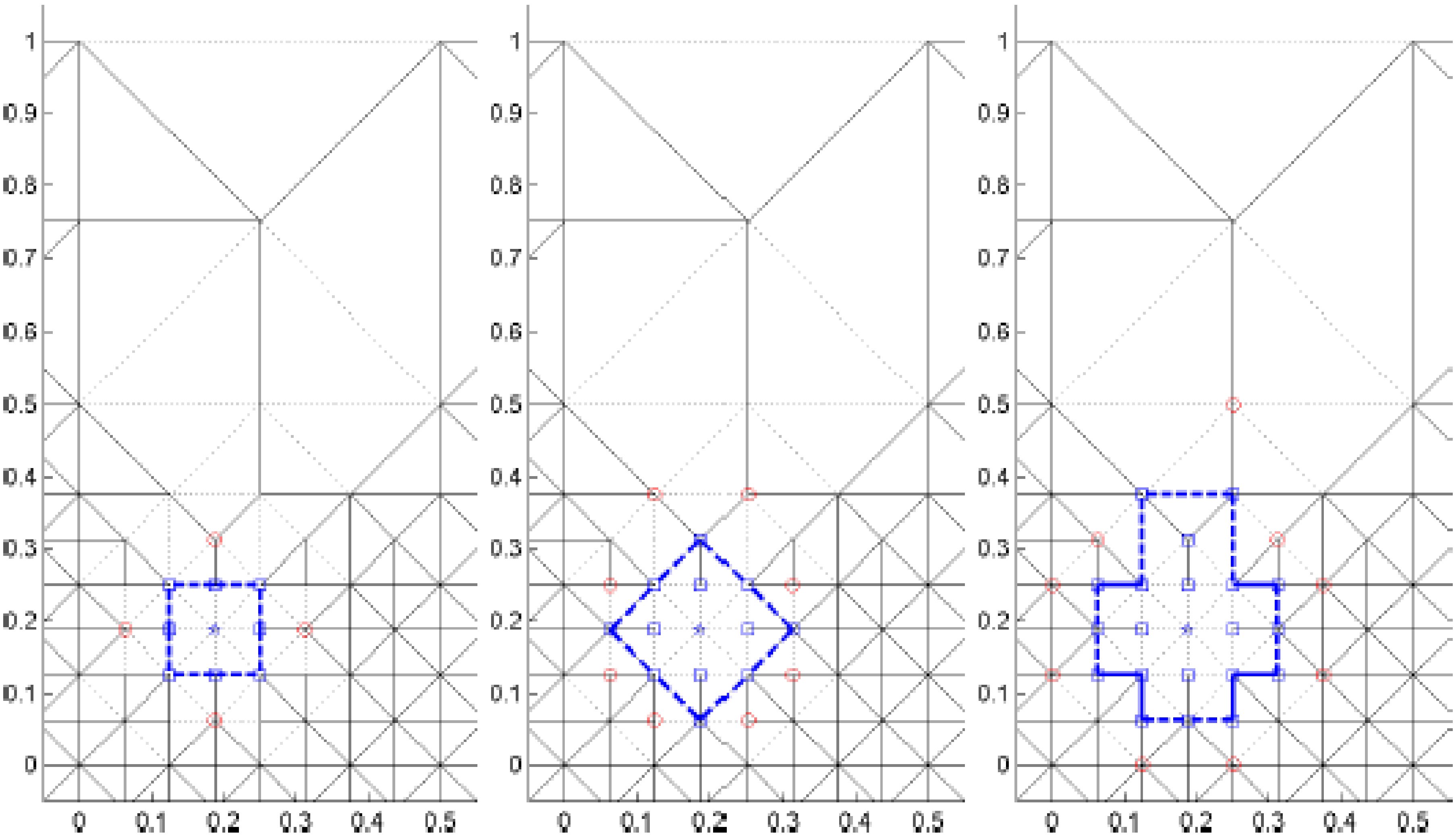


[Sethian & Vladimirsky, SINUM, 2003]

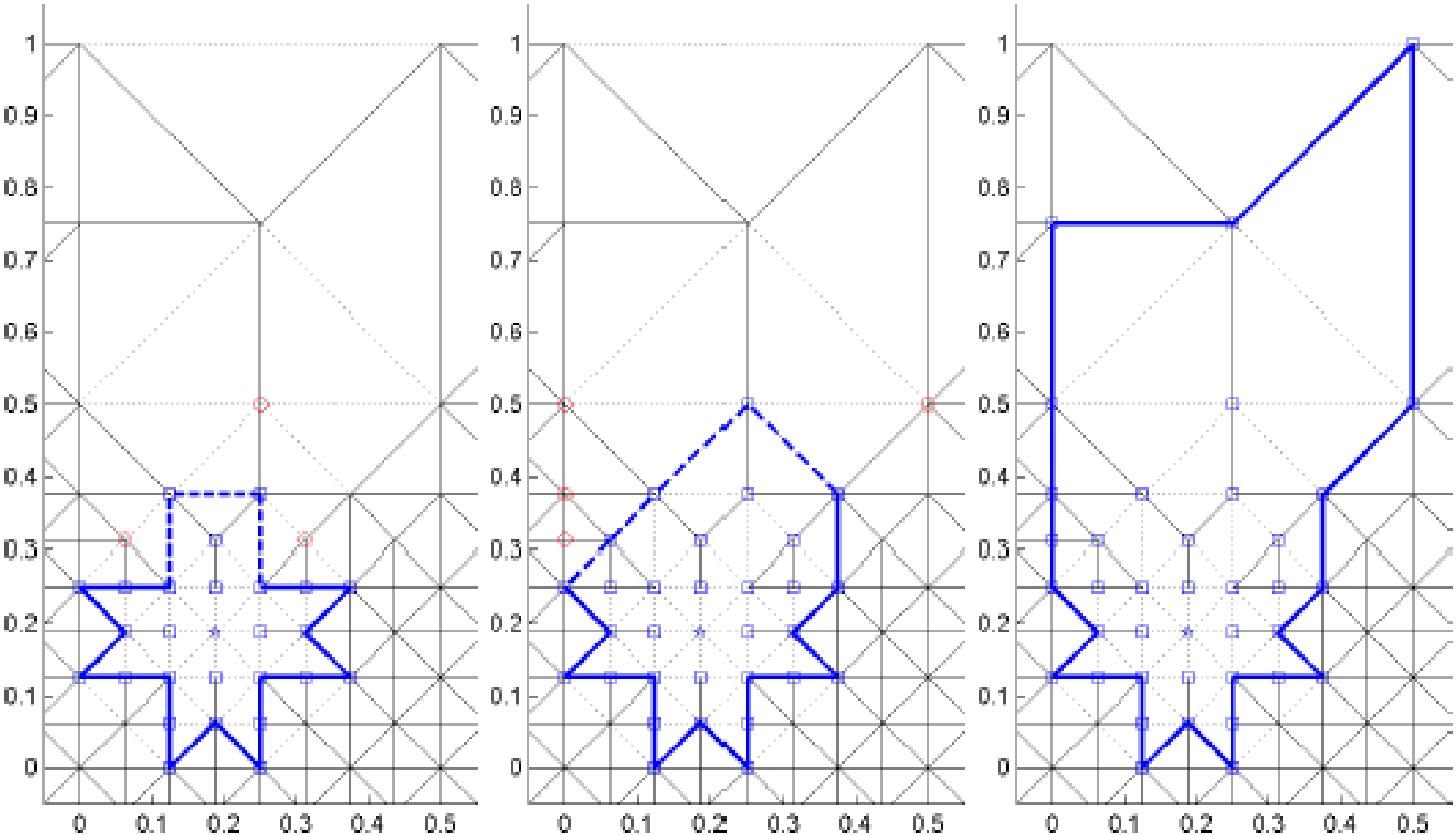
Monotone Acceptance OUM (MAOUM)

- Like AFOUM
  - extension of FMM to solve problems with general convex speed profiles in $\mathcal{O}$(N log N)
- Unlike AFOUM
  - Dijkstra-like algorithm: computes solution in order of nondecreasing value
  - Standard convergence proof [Barles & Souganidis, 1991]
  - Simple conversion to a Dial-like algorithm that sorts and accepts solution values using buckets
  - Stencil size adjusts to the local level of grid refinement
  - No accepted front
  - Initial pass through grid to generate stencils based on tests that can be applied to each potential face of the stencil
  - Must store stencils
- Alton & Mitchell, submitted to J. Scientific Computing

Stencil Generation Algorithm

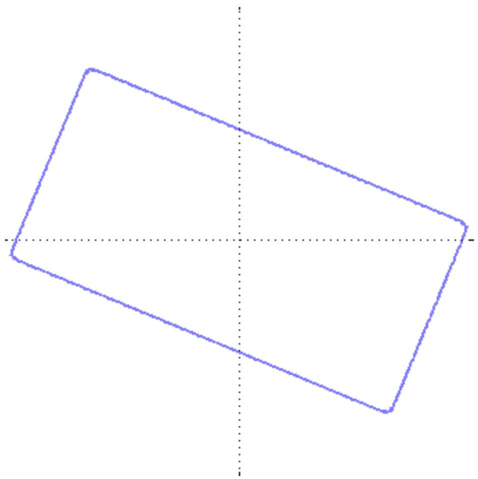


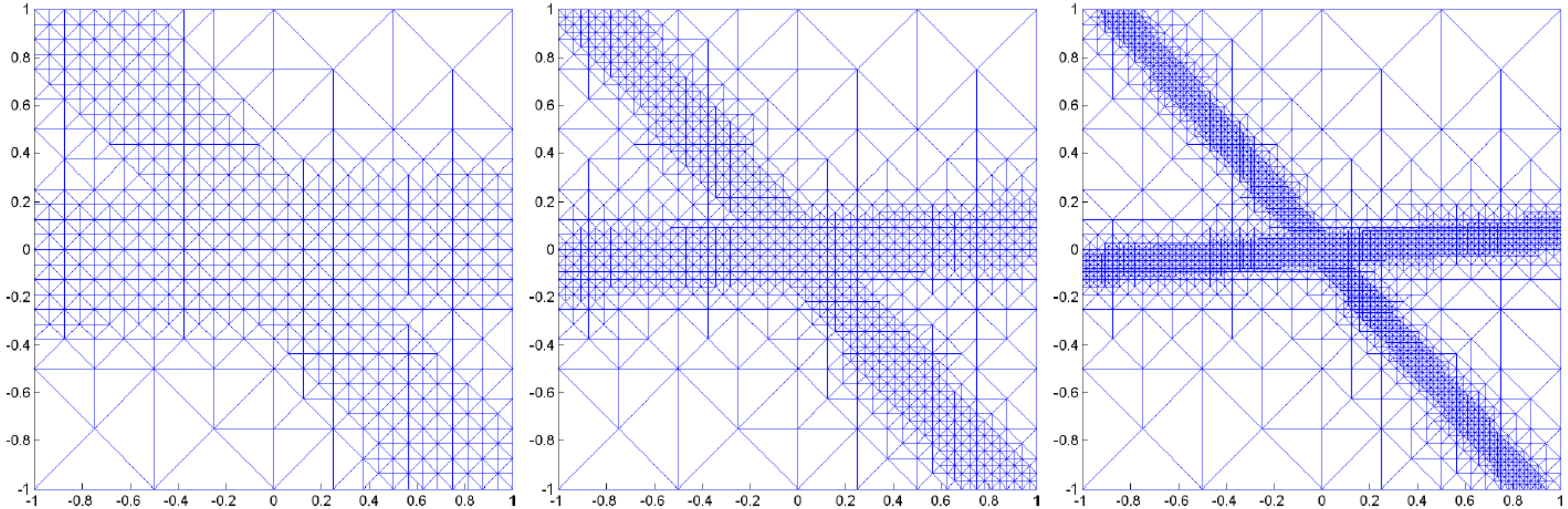
Stencil Generation (continued)



Experiment: Rectangular Speed Profile

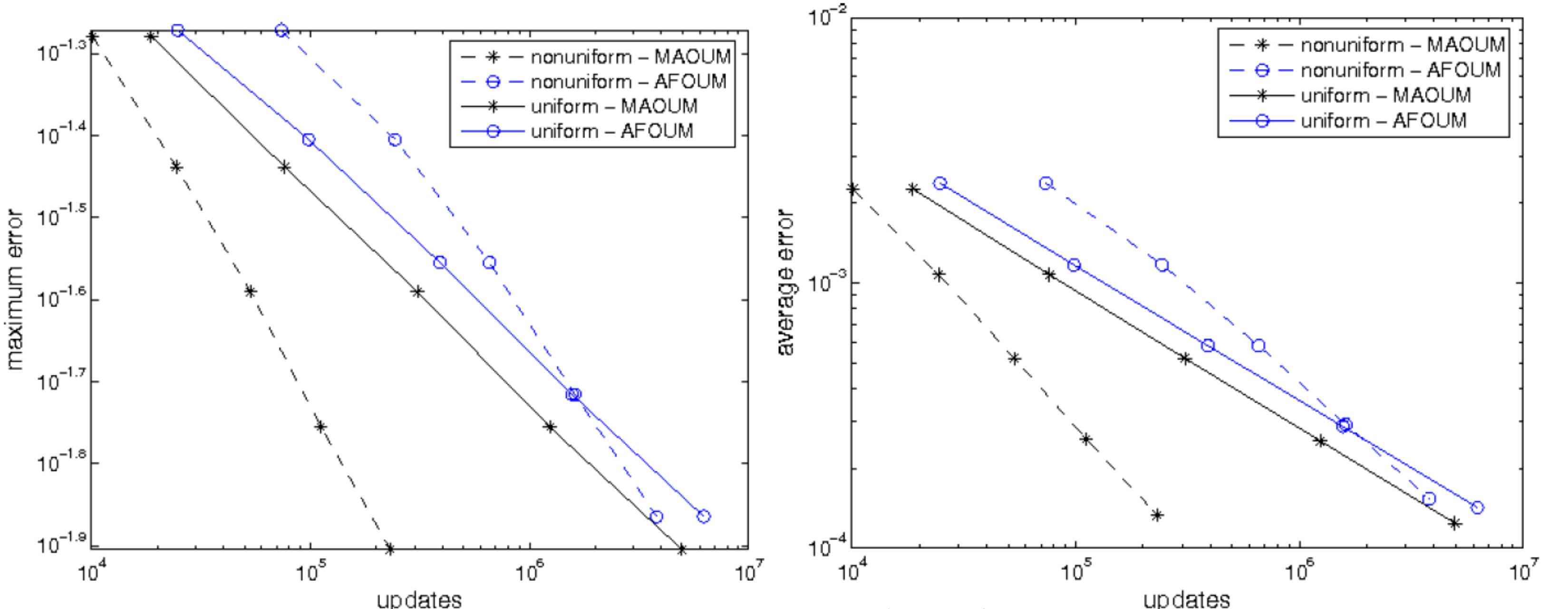
- Homogeneous speed profile
- Boundary condition specified at origin
- Grid refined where solution and characteristics are highly curved





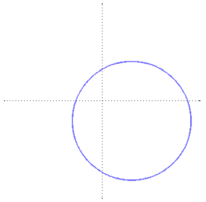
Results: Rectangular Speed Profile

- MAOUM and AFOUM on uniform and nonuniform grids
- Maximum and average error versus updates
- Nonuniform grid has better error convergence rate for both algorithms than nonuniform grid
- MAOUM on nonuniform grid has smallest error



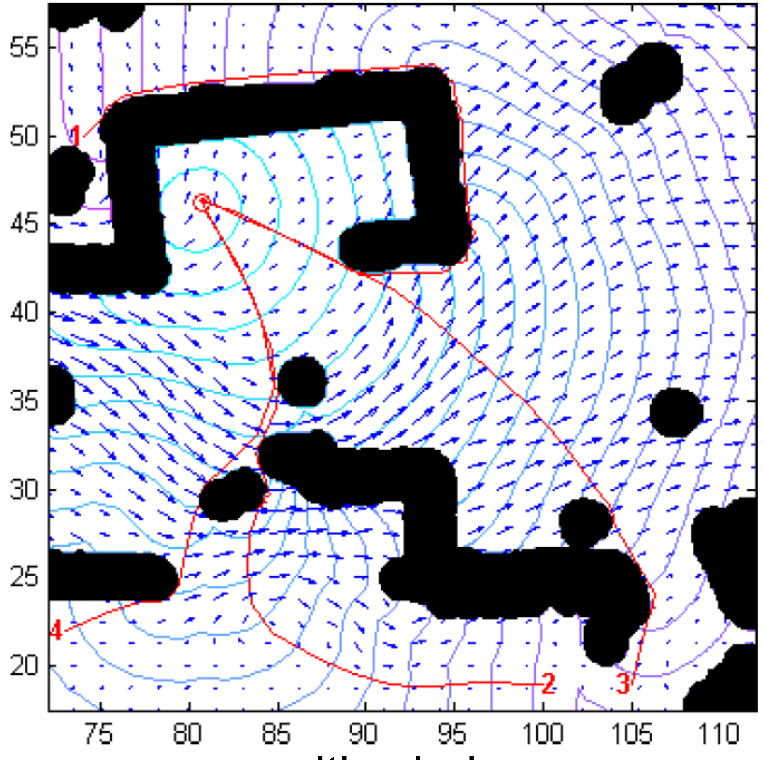
Example: Robot Path Planning

- Robot wants to reach goal in minimal time avoiding obstacles and fighting a fierce wind
- Solved with new ordered upwind scheme: Monotone Acceptance OUM
  - Alton & Mitchell, submitted J. Scientific Computing

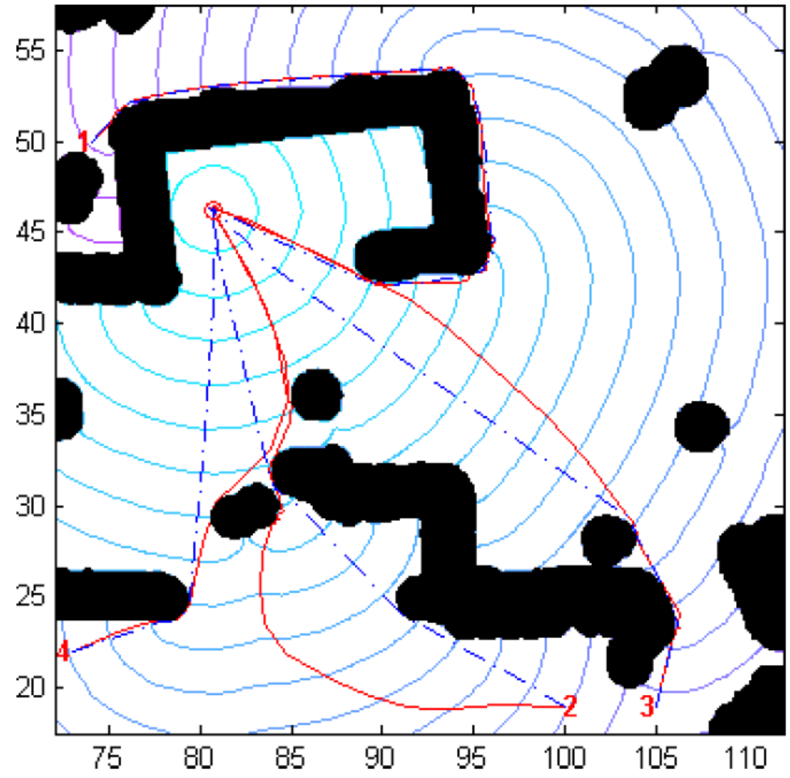


speed profile

$$\mathcal{A}_f(x) = \{y \mid \|y - \vec{f}_w(x)\| \le f_r\}$$



with wind



with and without wind

Path Planning: An Application of the Static Hamilton Jacobi Equation

For more information contact

Ian Mitchell

Department of Computer Science

The University of British Columbia

mitchell@cs.ubc.ca

http://www.cs.ubc.ca/~mitchell