# CPSC 542D: Level Set Methods

## Dynamic Implicit Surfaces and the Hamilton-Jacobi Equation

## or

## What Water Simulation, Robot Path Planning and Aircraft Collision Avoidance Have in Common

Ian Mitchell

Department of Computer Science

The University of British Columbia

research supported by

National Science and Engineering Research Council of Canada

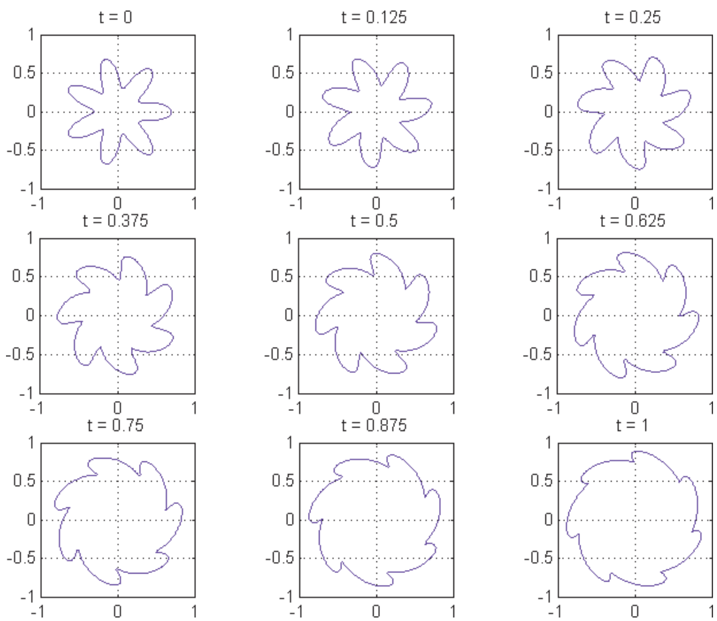
ONR Computational Methods for Collaborative Control MURI (N00014-02-1-0729)

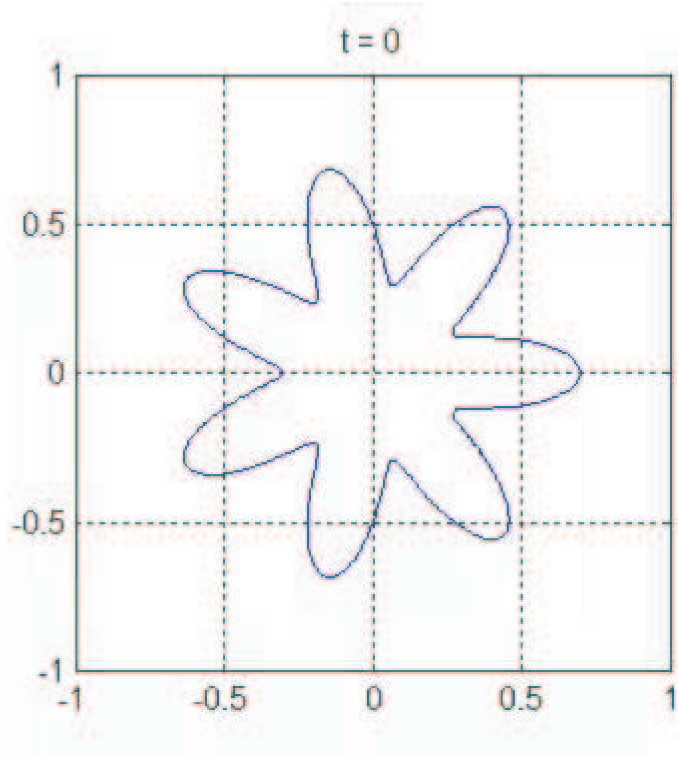
# Outline

- Part 1: Dynamic implicit surfaces
  - Application: Free surface fluid simulation
  - Challenge: Volume conservation
- Part 1-2: Optimal control
  - Application: Reach sets for control verification
  - Application: Filtering pilot commands for safety
- Part 2: The Stationary HJ PDE
  - Application: Robotic path planning

# Dynamic Interfaces

- How do you represent an evolving interface?



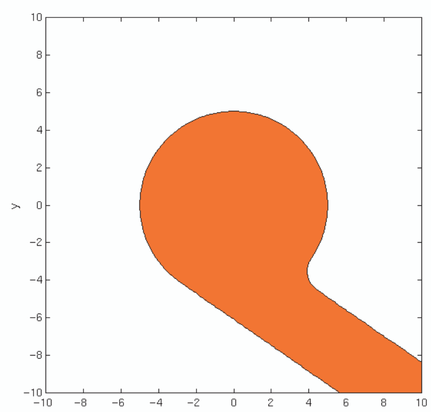
# Implicit Surface Functions

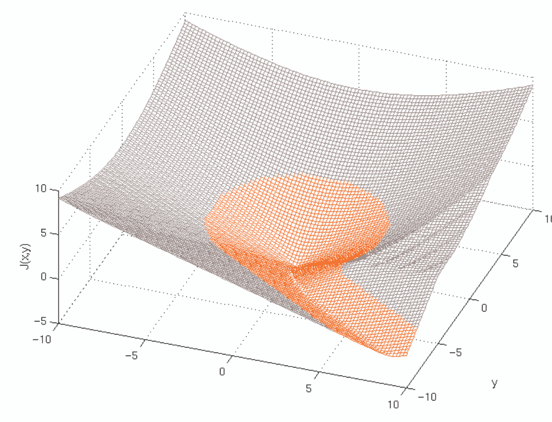
- Surface $S(t)$ and/or set $G(t)$ are defined implicitly by an isosurface of a scalar function $\phi(x,t)$, with several benefits
  - State space dimension does not matter conceptually
  - Surfaces automatically merge and/or separate
  - Geometric quantities are easy to calculate

$$\phi : \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}$$

$$\mathcal{G}(t) = \{x \in \mathbb{R}^n \mid \phi(x,t) \leq 0\}$$

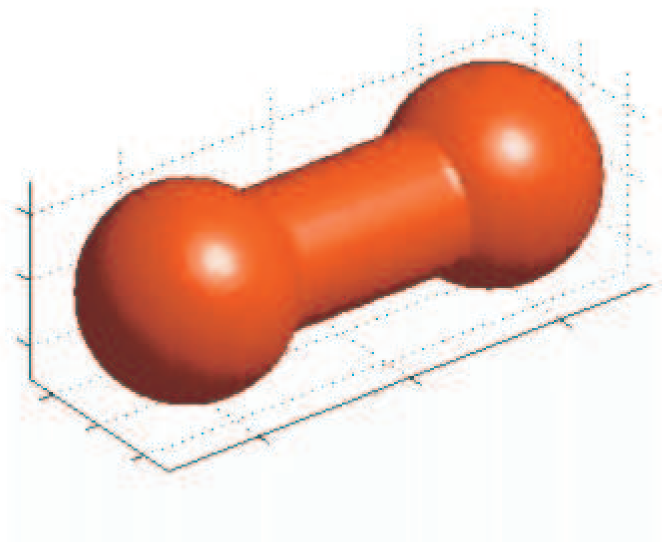
$$\mathcal{S}(t) = \partial\mathcal{G}(t) = \{x \in \mathbb{R}^n \mid \phi(x,t) = 0\}$$



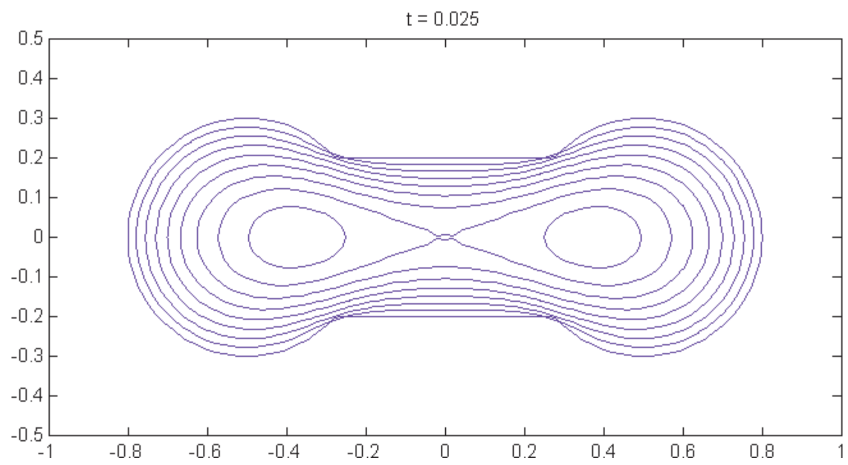



# Implicit Surface Benefits

- What about three dimensions?
- What about changing topology?



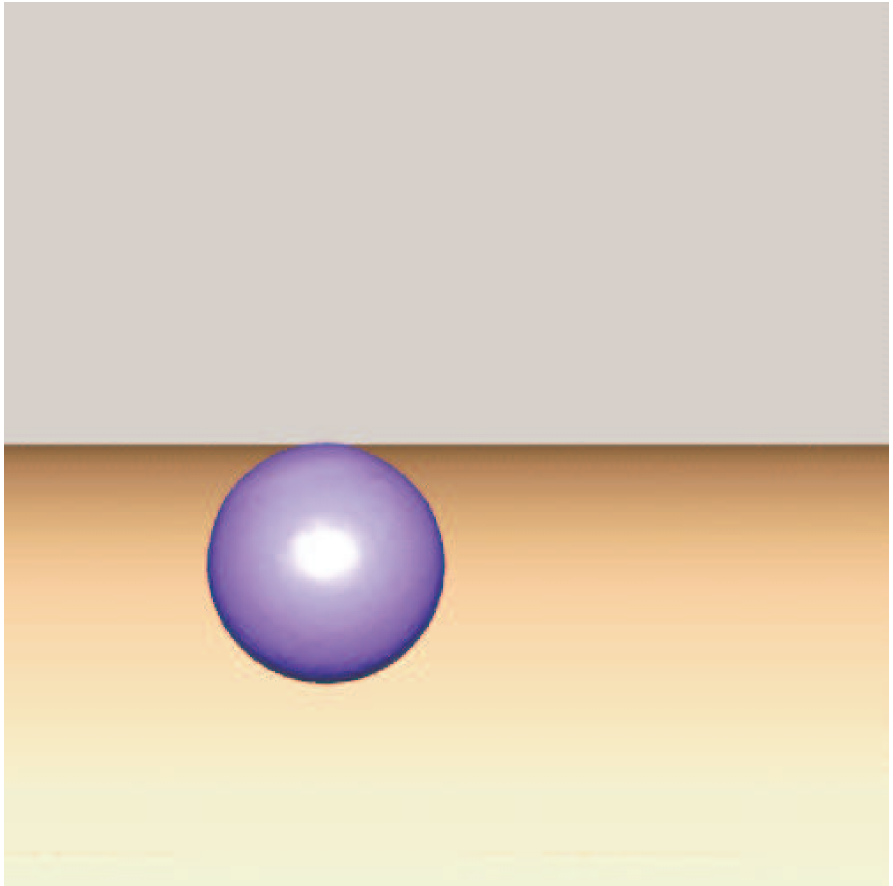
shrinking dumbbell




contour slice through midplane

# Level Set Methods and Boundaries

- Level sets are just one method of tracking interfaces
  - Underlying theory: Hamilton-Jacobi equations
- Advantages
  - Geometric information easy to extract
  - Handles merging and breaking interfaces
  - Easy to implement in 3D
- Disadvantages
  - Volume loss



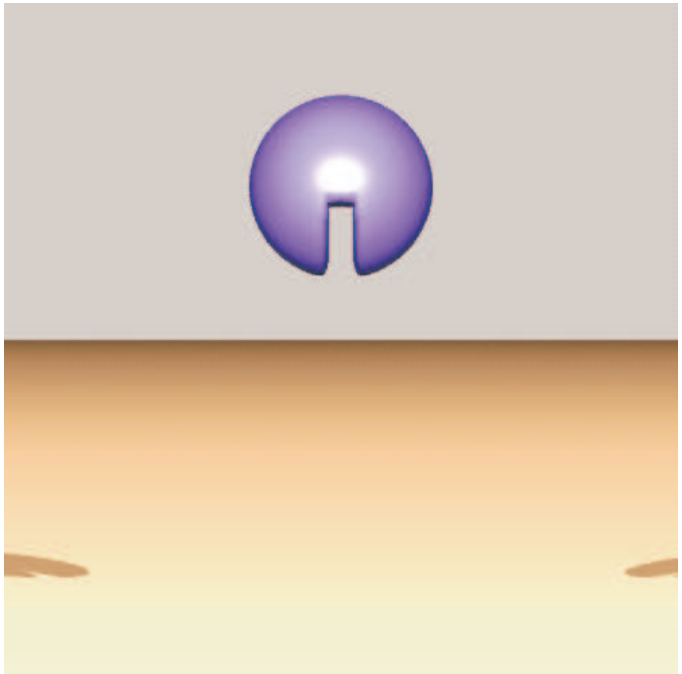
# Application: Animating Fluids

- State of the art evolving interface
  - Merging and separating surfaces
  - Smooth simulation and rendering of fluid and container
  - Plausible water motion

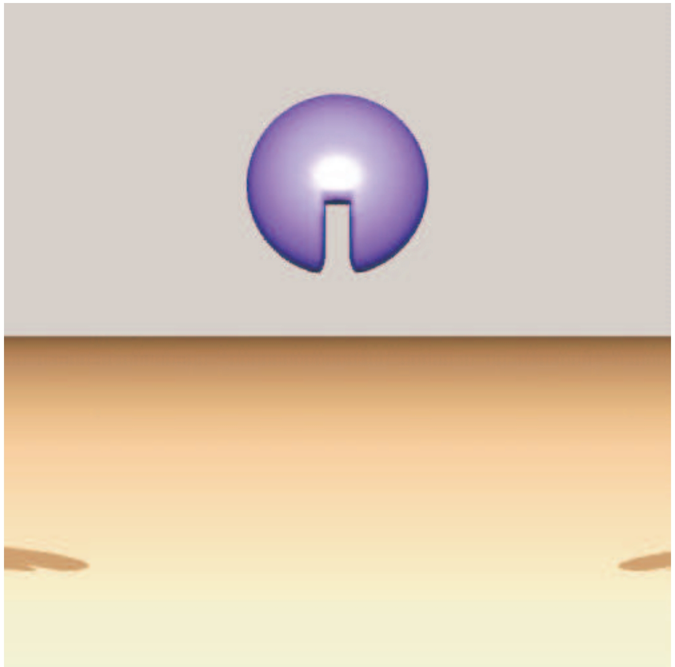
# Notched Sphere

- 3D version of Zalesak's disk
- $100^3$ grid, notch width 5, roughly 64 particles per cell

Level Set Only



Particle Level Set

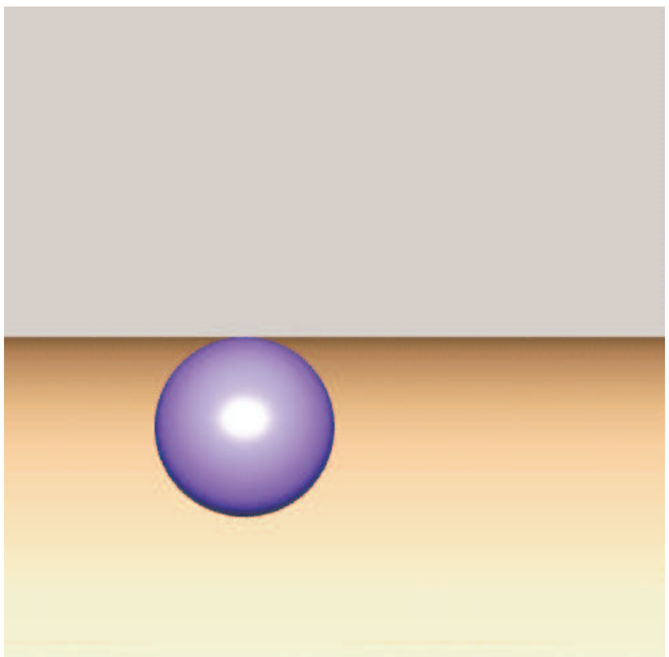


Rendering by Sou Cheng Choi

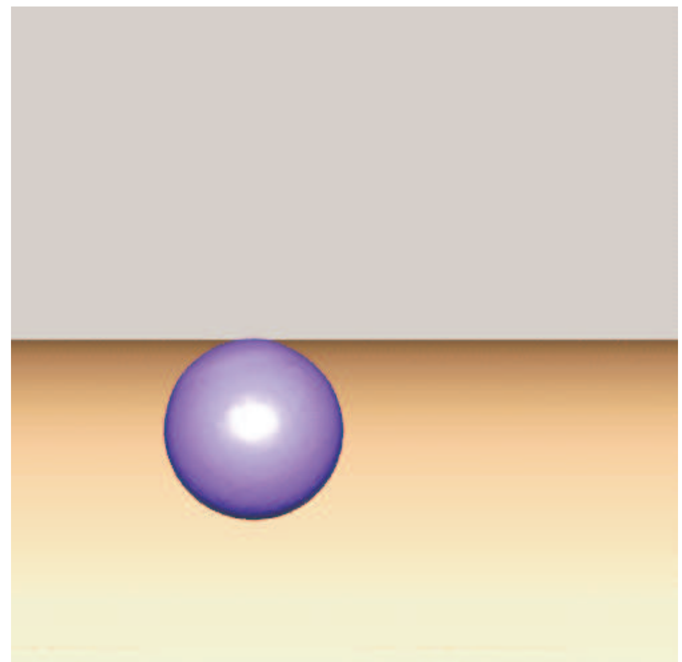
# Pushing the Limits

- Fully 3D vortex stretch of sphere (vortex in x-y and x-z planes)
  - $100^3$ grid, error is evaluated by time reversing the flow
  - [LeVeque, 1996]

Level Set Only



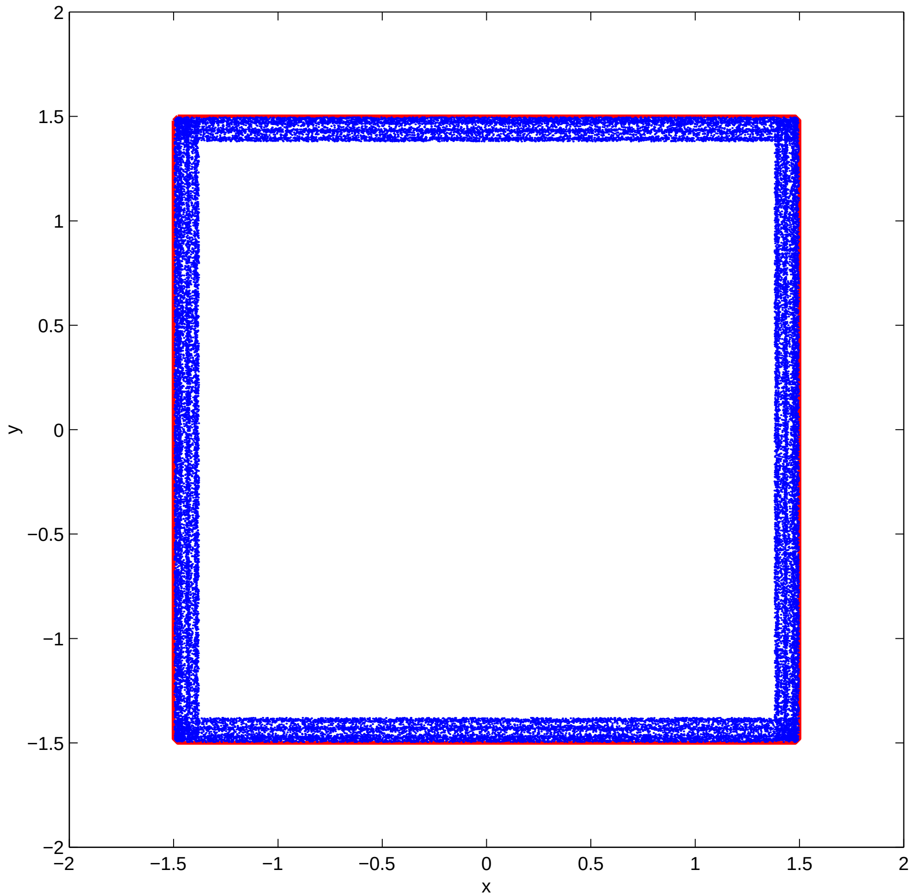
Particle Level Set

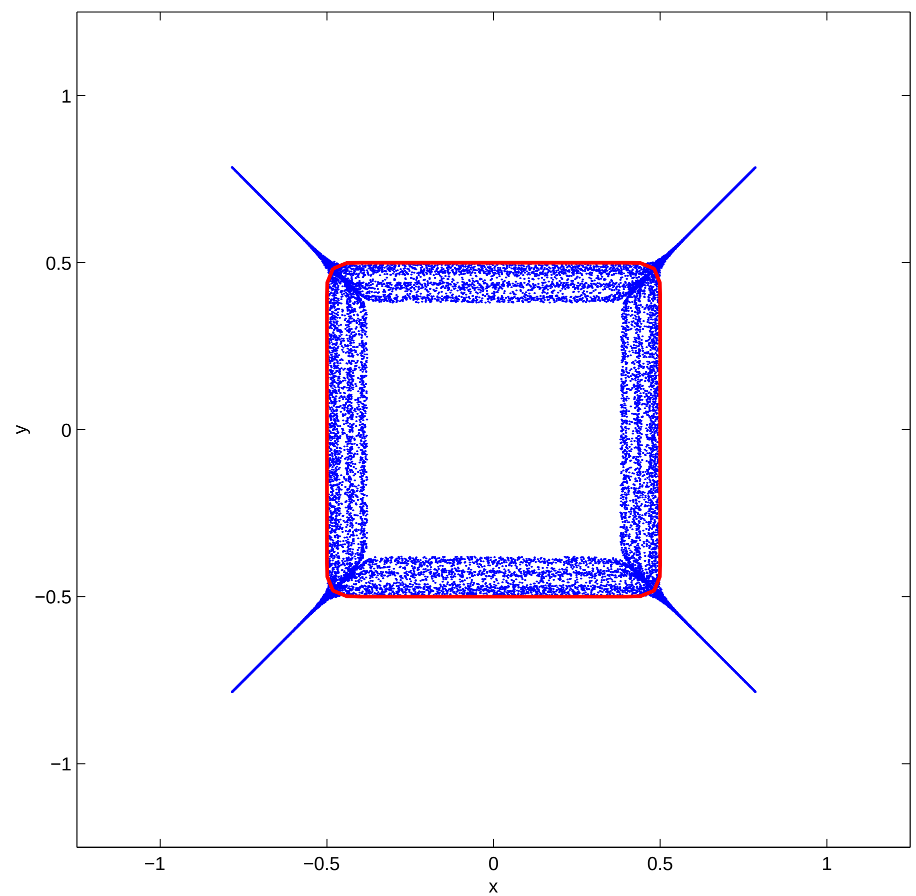


Rendering by Sou Cheng Choi

# Not Finished Yet

- Reports of dubious repeatability.
- What about shocks?  Particle methods fail.



# Outline

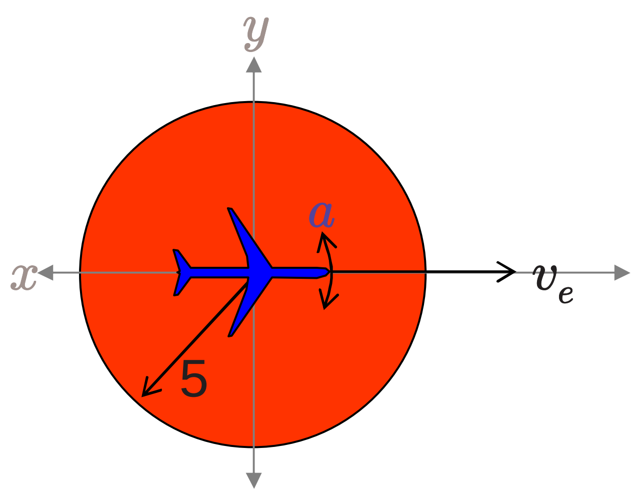
- Part 1: Dynamic implicit surfaces
  - Application: Free surface fluid simulation
  - Challenge: Volume conservation
- Part 1-2: Optimal control
  - Application: Reach sets for control verification
  - Application: Filtering pilot commands for safety
- Part 2: The Stationary HJ PDE
  - Application: Robotic path planning

# Game of Two Identical Vehicles

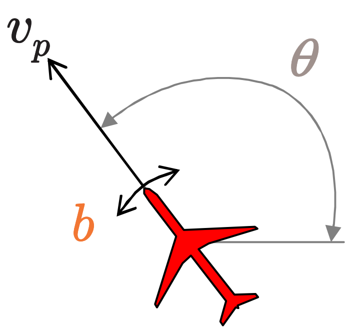
- Classical collision avoidance example
  - Collision occurs if vehicles get within five units of one another
  - Evader chooses turn rate $|a| \leq 1$ to avoid collision
  - Pursuer chooses turn rate $|b| \leq 1$ to cause collision
  - Fixed equal velocity $v_e = v_p = 5$

dynamics (pursuer)

$$\frac{d}{dt}\begin{bmatrix} x_p \\ y_p \\ \theta_p \end{bmatrix} = \begin{bmatrix} v_p \cos \theta_p \\ v_p \sin \theta_p \\ b \end{bmatrix}$$



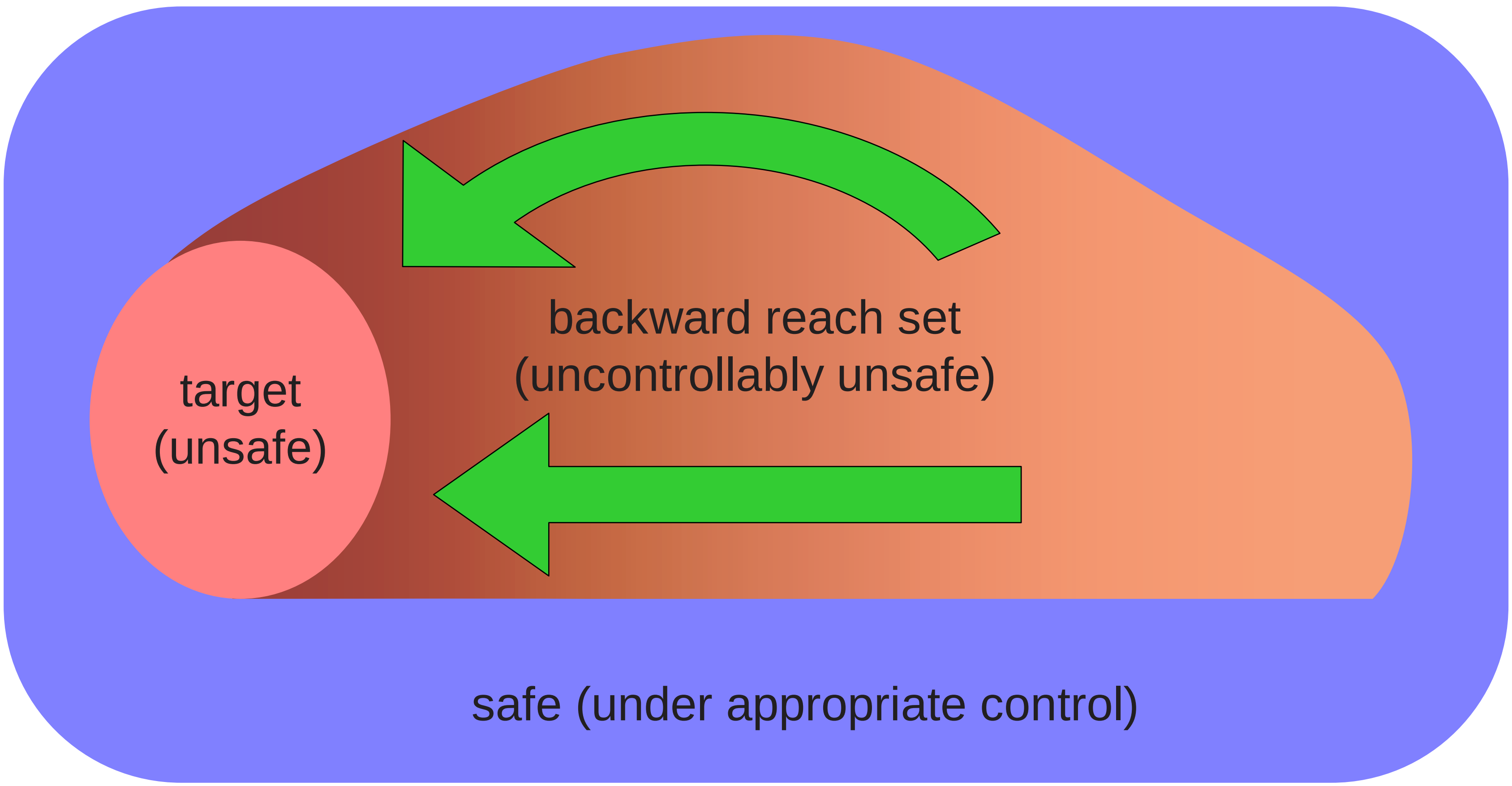

evader aircraft (control)




pursuer aircraft (disturbance)

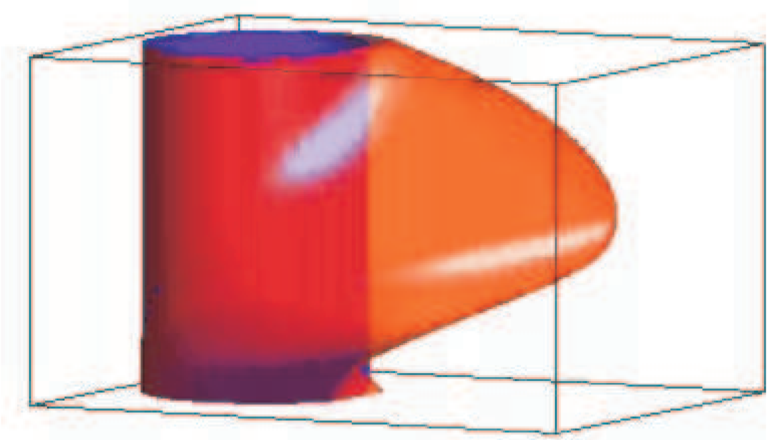
# Reachable Sets: What and Why?

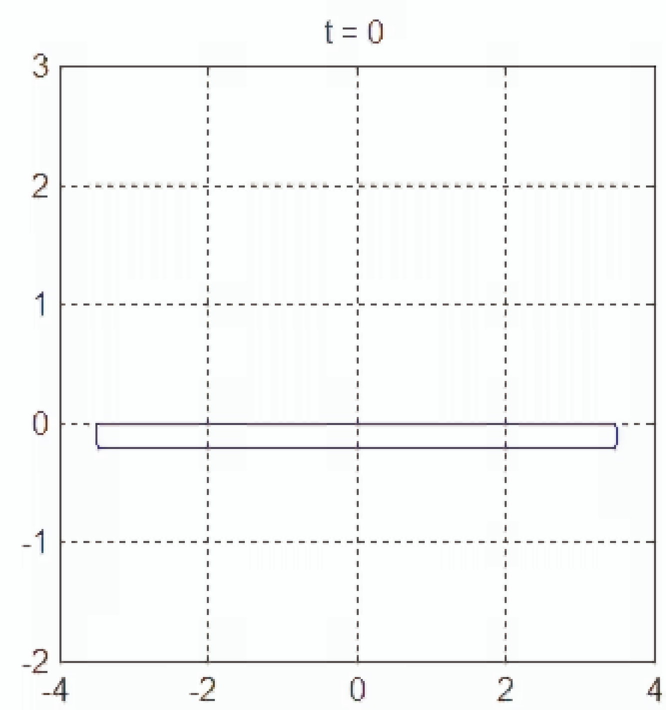
- One application: safety analysis
  - What states are doomed to become unsafe?
  - What states are safe given an appropriate control strategy?



# Calculating Reach Sets

- Two primary challenges
  - How to represent set of reachable states
  - How to evolve set according to dynamics
- Discrete systems $x_{k+1} = \delta(x_k)$
  - Enumerate trajectories and states
  - Efficient representations: Binary Decision Diagrams
- Continuous systems $dx/dt = f(x)$?



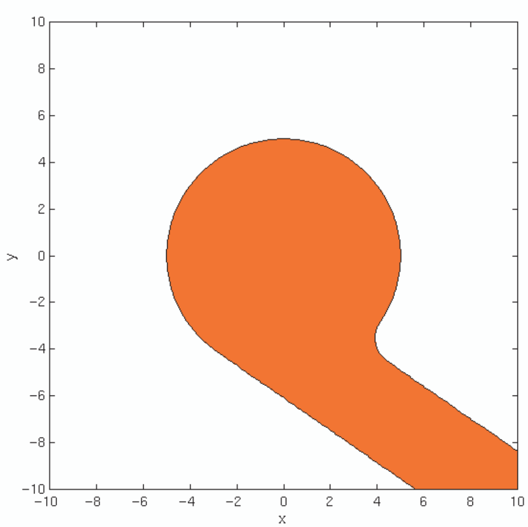


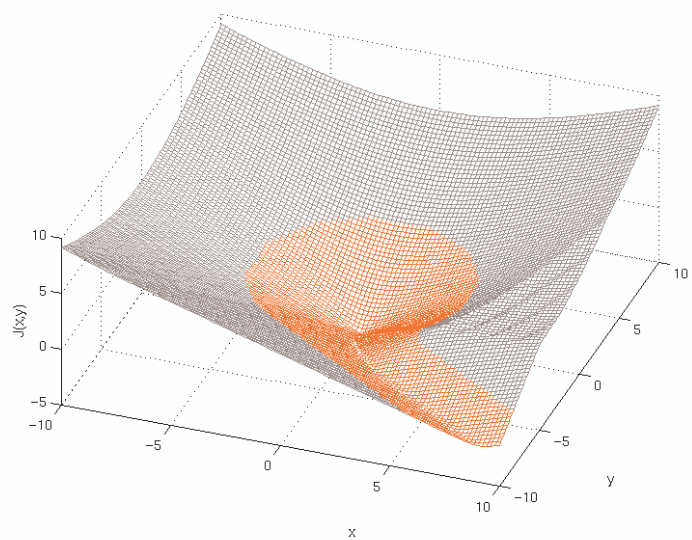
# Implicit Surface Functions

- Set $G(t)$ is defined implicitly by an isosurface of a scalar function $\phi(x,t)$, with several benefits
  - State space dimension does not matter conceptually
  - Surfaces automatically merge and/or separate
  - Geometric quantities are easy to calculate

$$\phi : \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}$$

$$\mathcal{G}(t) = \{x \in \mathbb{R}^n \mid \phi(x,t) \leq 0\}$$

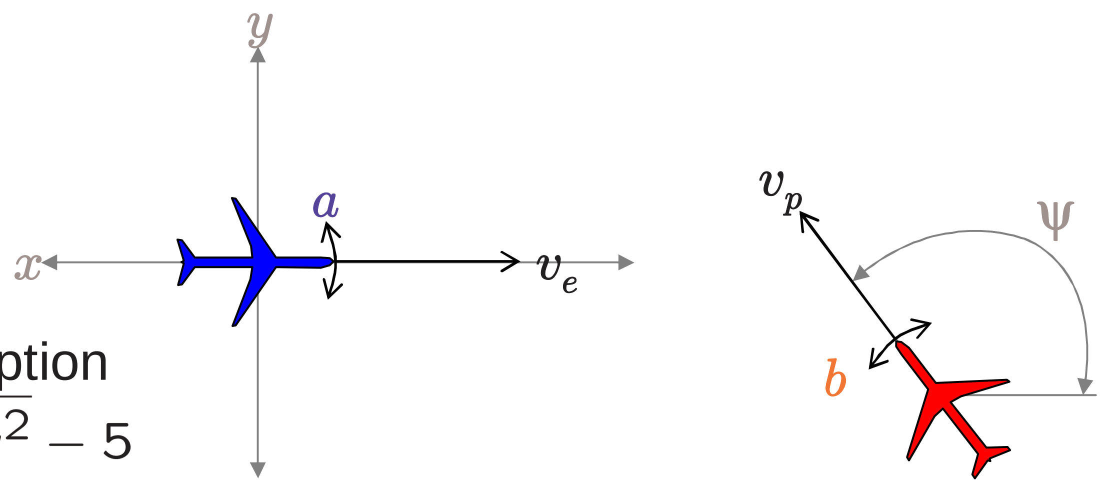


# Collision Avoidance Computation

- Work in relative coordinates with evader fixed at origin
  - State variables are now relative planar location ($x$,$y$) and relative heading $\psi$

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ \psi \end{bmatrix} = \begin{bmatrix} -v_e + v_p \cos\psi - ay \\ v_p \sin\psi - ax \\ b - a \end{bmatrix}$$




target set description

$$h(x) = \sqrt{x^2 + y^2} - 5$$

evader aircraft (control) pursuer aircraft (disturbance)

# Evolving Reachable Sets

- Modified Hamilton-Jacobi partial differential equation

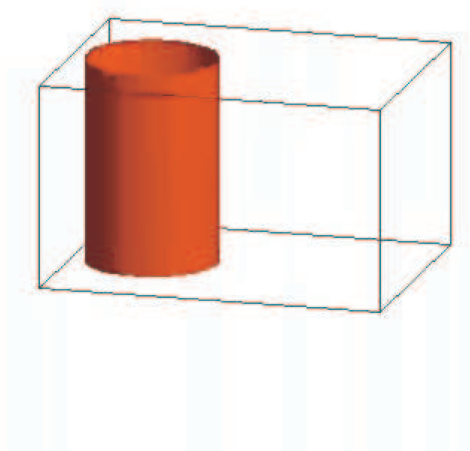
$$D_t\phi(x,t) + \min\left[0, H(x, D_x\phi(x,t))\right] = 0$$

$$\text{with Hamiltonian}: \quad H(x,p) = \max_{a\in\mathcal{A}} \min_{b\in\mathcal{B}} f(x,a,b)\cdot p$$

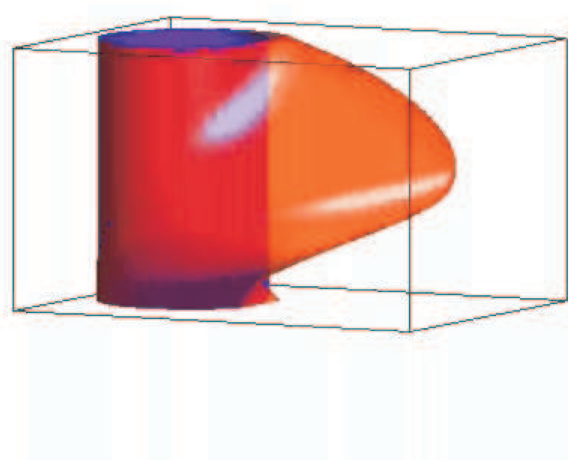
$$\text{and terminal conditions}: \quad \phi(x,0) = h(x)$$

$$\text{where} \qquad G(0) = \{x \in \mathbb{R}^n \mid h(x) \leq 0\}$$

$$\text{and} \qquad \dot{x} = f(x,a,b)$$



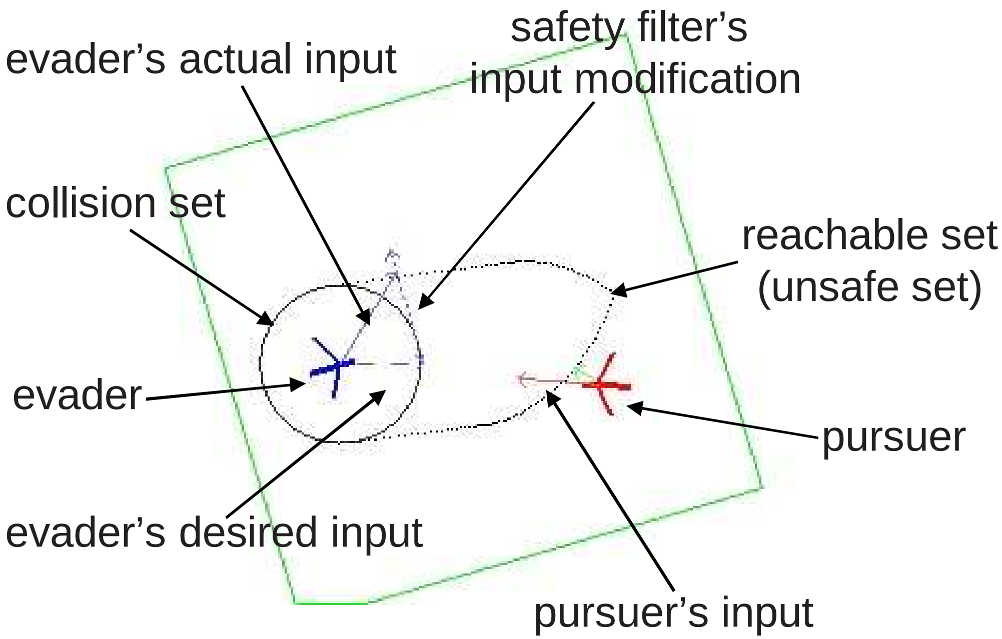
growth of reachable set

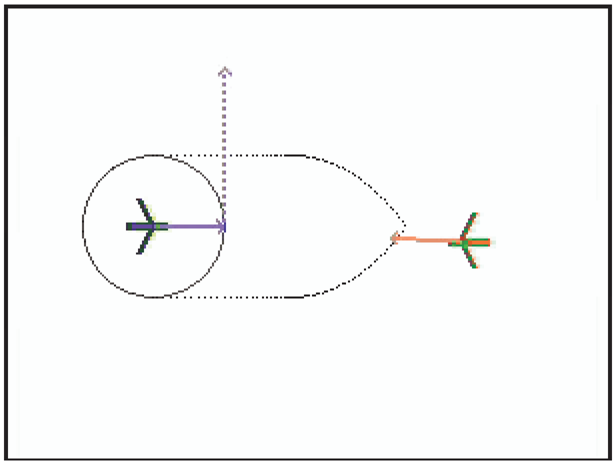


final reachable set

# Application: Softwalls for Aircraft Safety

- Use reachable sets to guarantee safety
- Basic Rules
  - Pursuer: turn to head toward evader
  - Evader: turn to head east
- Evader's input is filtered to guarantee that pursuer does not enter the reachable set

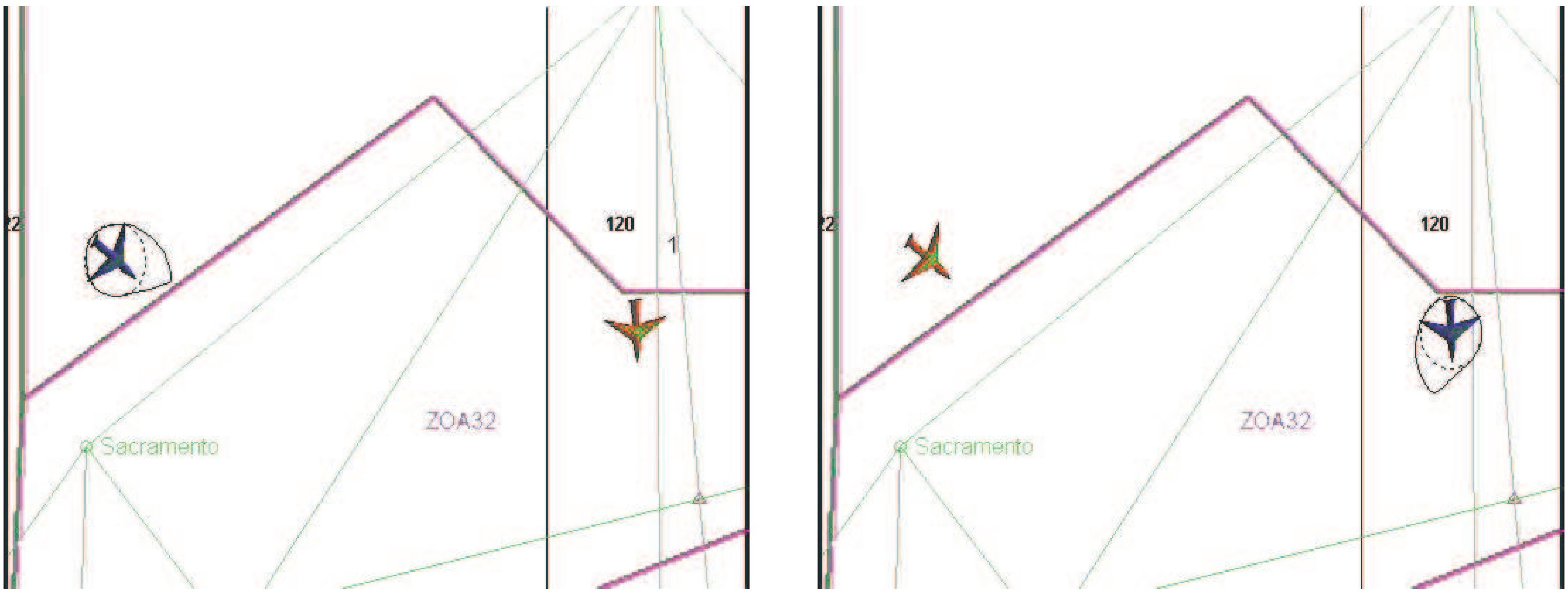


joint work with Edward Lee & Adam Cataldo

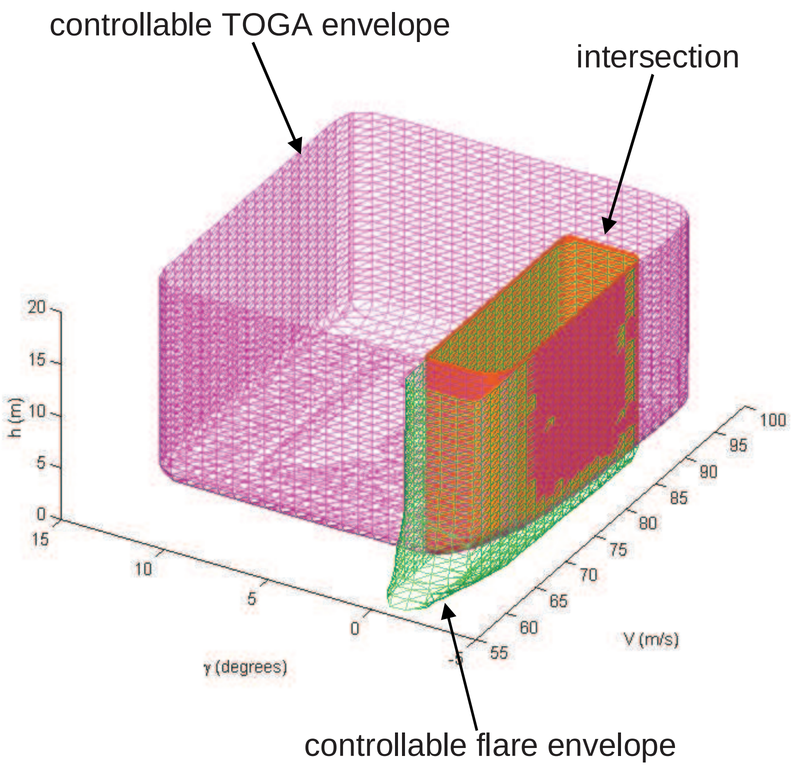
# Application: Collision Alert for ATC

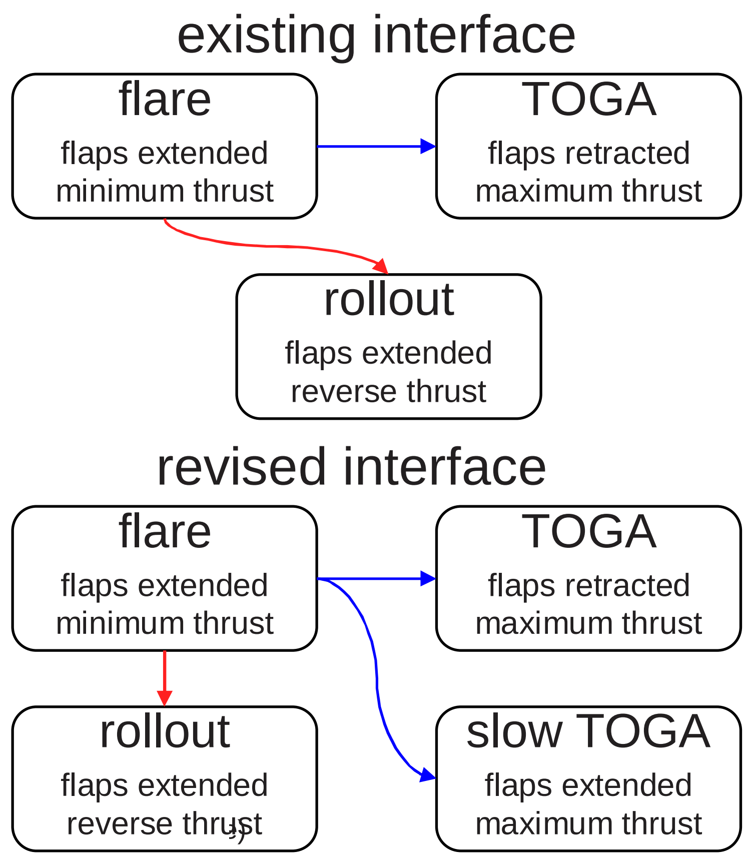
- Use reachable set to detect potential collisions and warn Air Traffic Control (ATC)
  - Find aircraft pairs in ETMS database whose flight plans intersect
  - Check whether either aircraft is in the other's collision region
  - If so, examine ETMS data to see if aircraft path is deviated
  - One hour sample in Oakland center's airspace—
    - 1590 pairs, 1555 no conflict, 25 detected conflicts, 2 false alerts



# Application: Cockpit Display Analysis

- Controllable flight envelopes for landing and Take Off / Go Around (TOGA) maneuvers may not be the same
- Pilot's cockpit display may not contain sufficient information to distinguish whether TOGA can be initiated



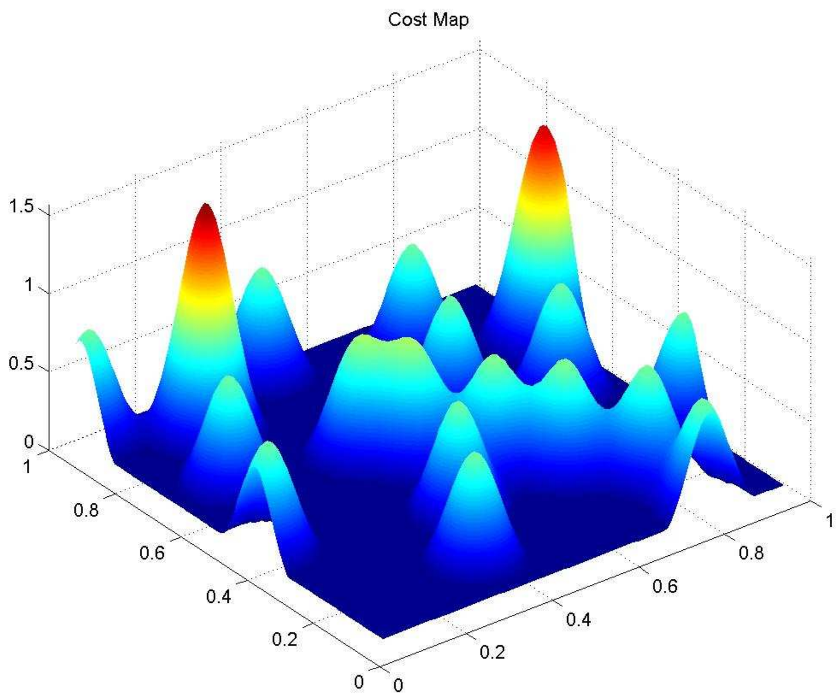



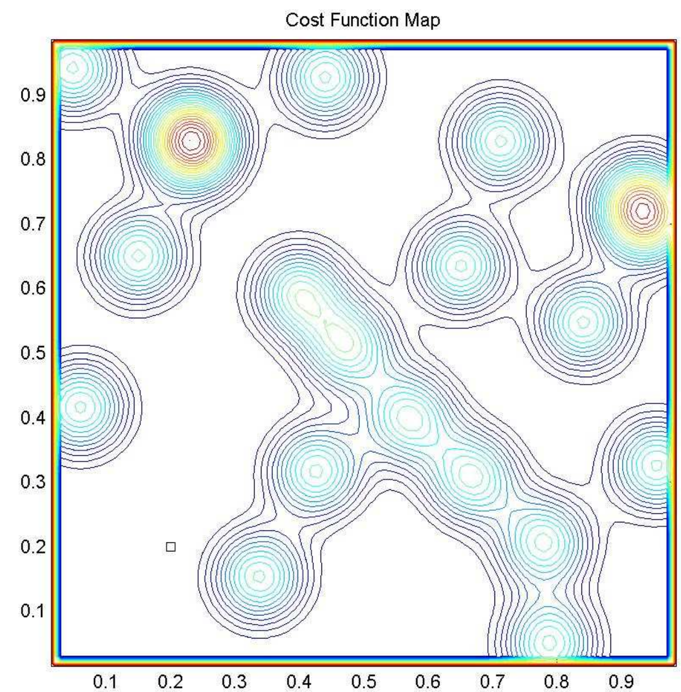
# Outline

- Part 1: Dynamic implicit surfaces
  - Application: Free surface fluid simulation
  - Challenge: Volume conservation
- Part 1-2: Optimal control
  - Application: Reach sets for control verification
  - Application: Filtering pilot commands for safety
- Part 2: The Stationary HJ PDE
  - Application: Robotic path planning

# Basic Path Planning

- Find the optimal path $p(s)$ to a target (or from a source)
- Inputs
  - Cost to pass through each state in the state space
  - Set of targets or sources (provides boundary conditions)



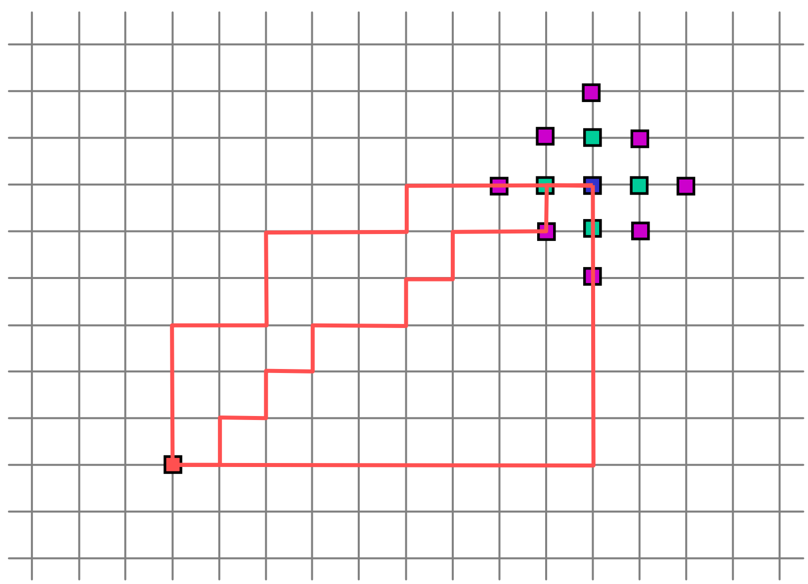
# Dynamic Programming Principle

$$V(x) = \min_{y \in N(x)} [V(y) + c(y \to x)]$$

- Value function $V(x)$ is “cost to go” from $x$ to the nearest target
- $V(x)$ at a point $x$ is the minimum over all points $y$ in the neighborhood $N(x)$ of the sum of
  - the cost $V(y)$ at point $y$
  - the cost $c(y \to x)$ to travel from $y$ to $x$
- Dynamic programming applies if
  - Costs are additive
  - Subsets of feasible paths are themselves feasible
  - Concatenations of feasible paths are feasible

# Dijkstra's Method

- Solution of dynamic programming on a discrete graph
  1. Set all interior nodes to a dummy value infinity $\infty$
  2. For all boundary nodes $\boldsymbol{x}$ and all $\boldsymbol{y} \in \boldsymbol{N(x)}$ approximate $\boldsymbol{V}(\boldsymbol{y})$ by DPP
  3. Sort all interior nodes with finite values in a list
  4. Pop node $\boldsymbol{x}$ with minimum value from the list and update $\boldsymbol{V}(\boldsymbol{y})$ by DPP for all $\boldsymbol{y} \in \boldsymbol{N(x)}$
  5. Repeat from (3) until all nodes have been popped



Constant cost map $\boldsymbol{c}(\boldsymbol{y} \rightarrow \boldsymbol{x}) = 1$

- Boundary node $\boldsymbol{V}(\boldsymbol{x}) = 0$
- First Neighbors $\boldsymbol{V}(\boldsymbol{x}) = 1$
- Second Neighbors $\boldsymbol{V}(\boldsymbol{x}) = 2$

- Distant node $\boldsymbol{V}(\boldsymbol{y}) = 15$

Optimal path?

# Eikonal Equation

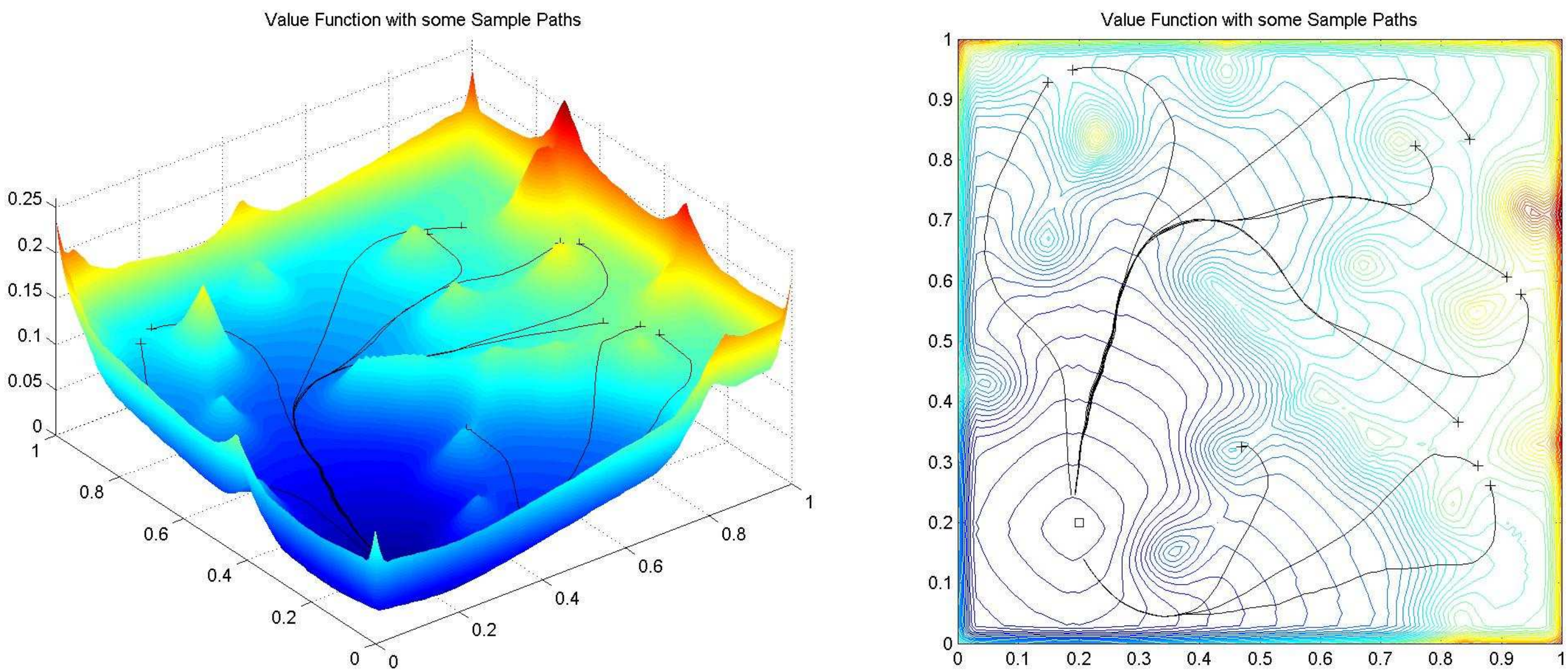
$$\|\nabla V(x)\| = c(x)$$

- Value function is viscosity solution of Eikonal equation
- Dynamic Programming Principle applies to Eikonal Equation
- Fast Marching Method: a continuous Dijkstra's algorithm
  - Node update equation is consistent with continuous PDE (and numerically stable)
  - Nodes are dynamically ordered so that each is visited a constant number of times

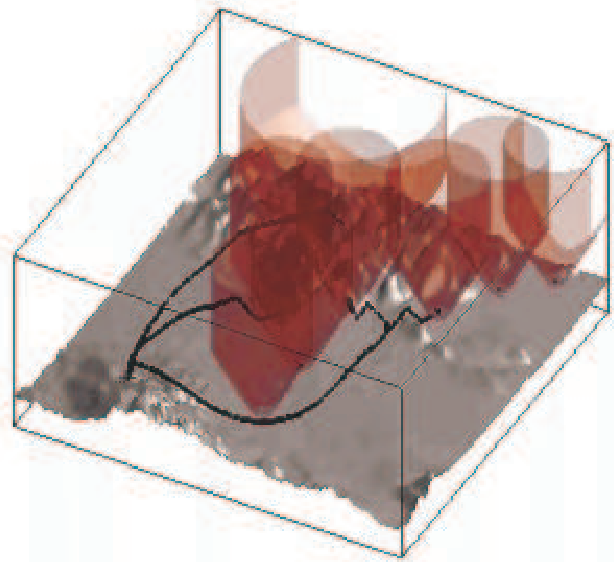
# Path Generation

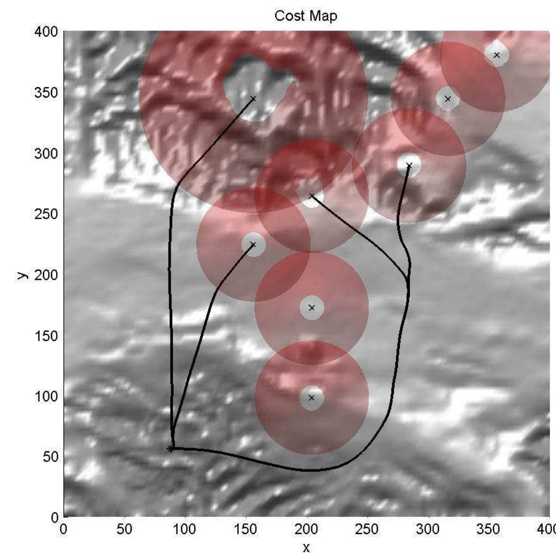
- Optimal path $p(s)$ is found by gradient descent
  - Value function $V(x)$ has no local minima, so paths will always terminate at a target

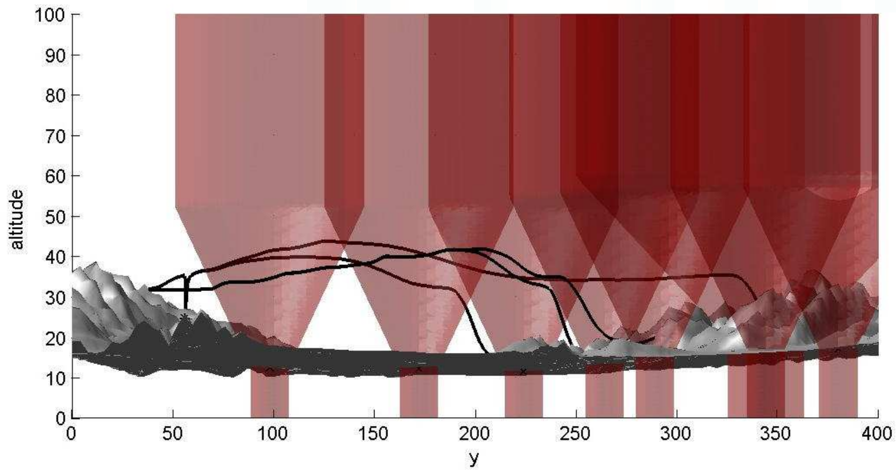
$$\frac{dp}{ds} = \frac{\nabla V(x)}{\|\nabla V(x)\|}$$



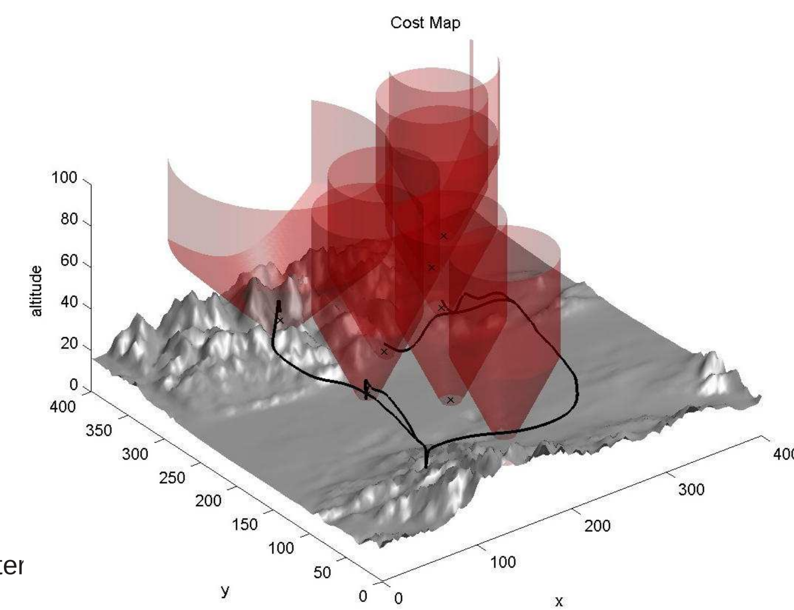
# Demanding Example?  No!





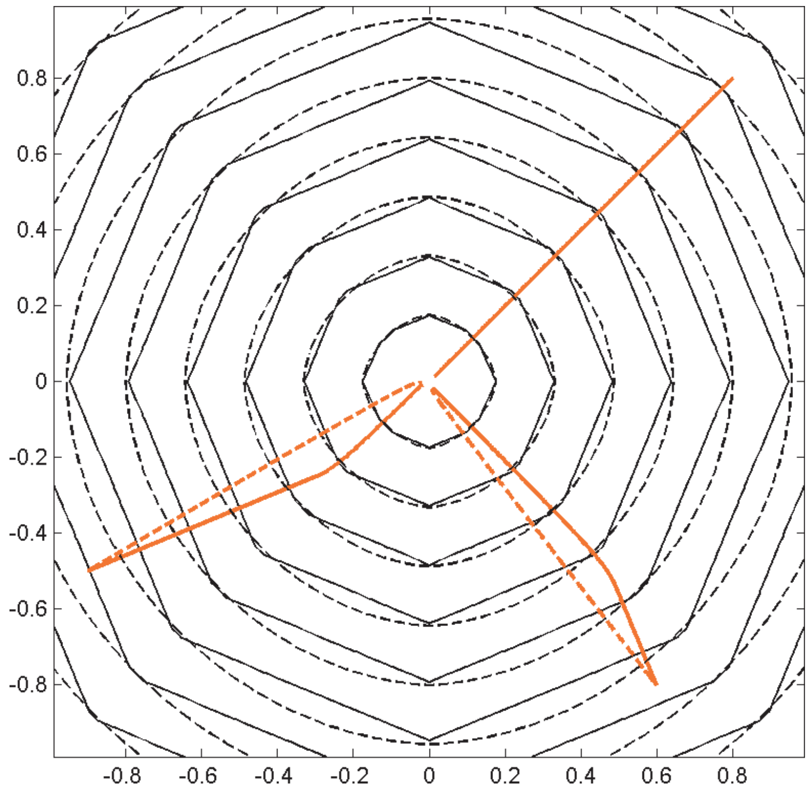




C Computer



# Treating the Continuous State Space

- Dijkstra's algorithm finds paths through discrete grid
  - Will not find some optimal paths even as grid is refined
- Fast marching method is consistent with continuous state space
  - Simple modification to node update equation

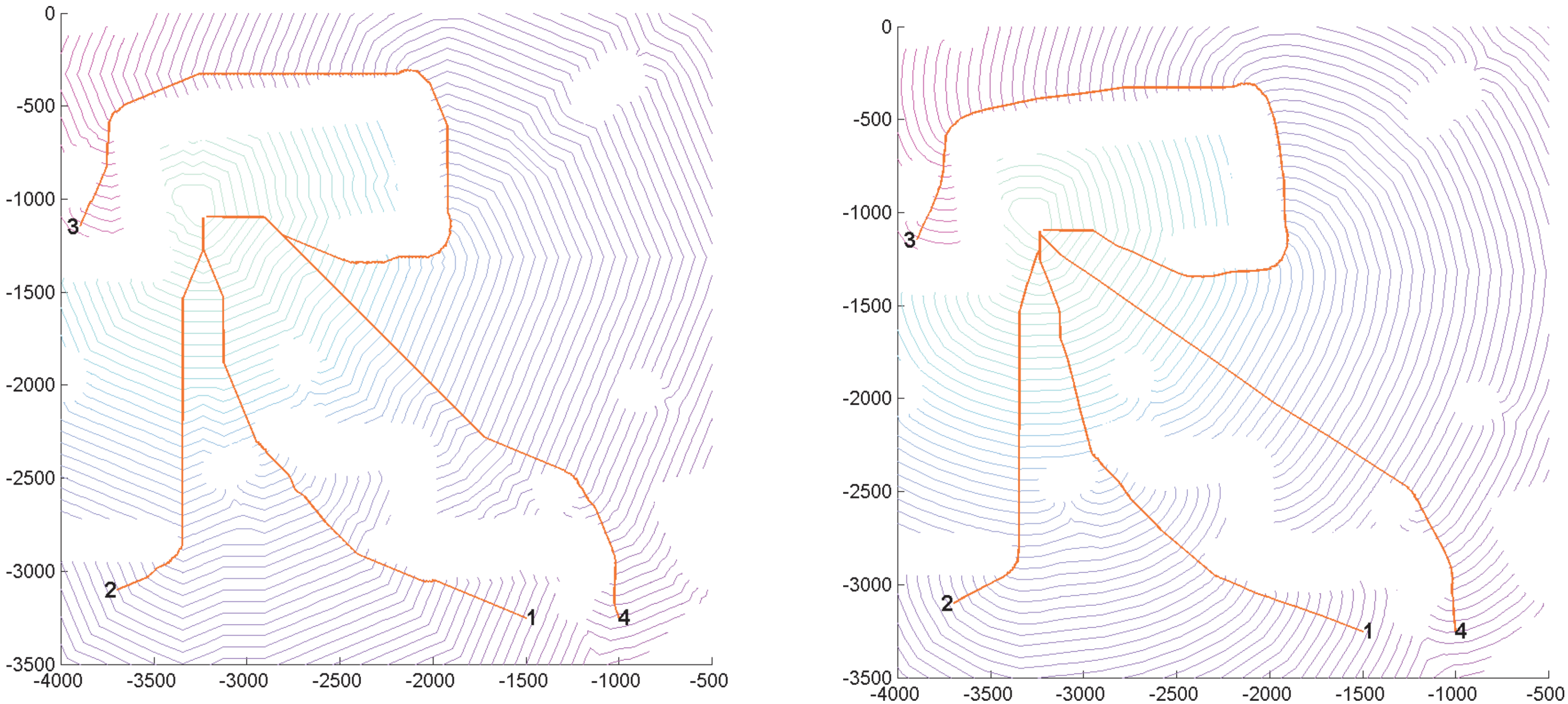



discrete Dijkstra's algorithm
(8 neighbors)

continuous fast
marching method

# Treating the Continuous State Space

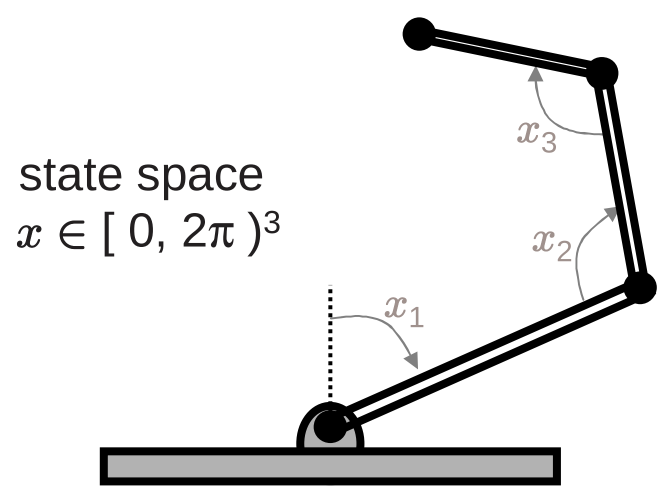
- Even when interpolation is used to extend discrete Dijkstra solution to the whole domain, trajectories tend to travel along grid edges




discrete Dijkstra's algorithm (8 neighbors) continuous fast marching method

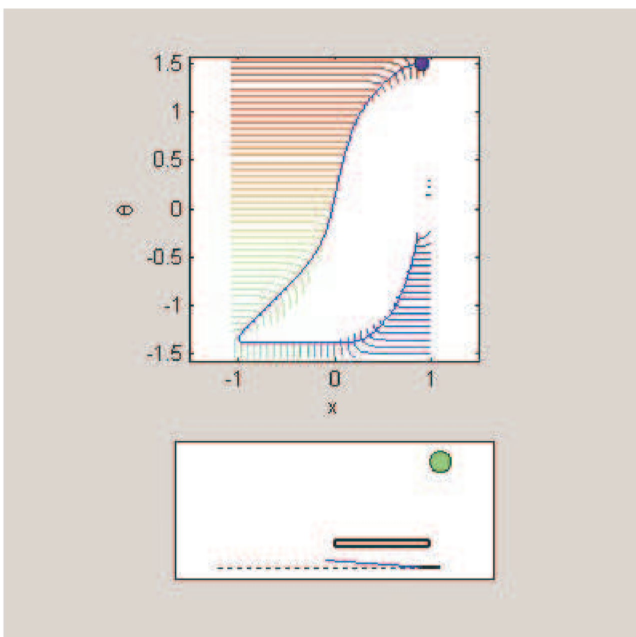
# Why the Euclidean Norm?

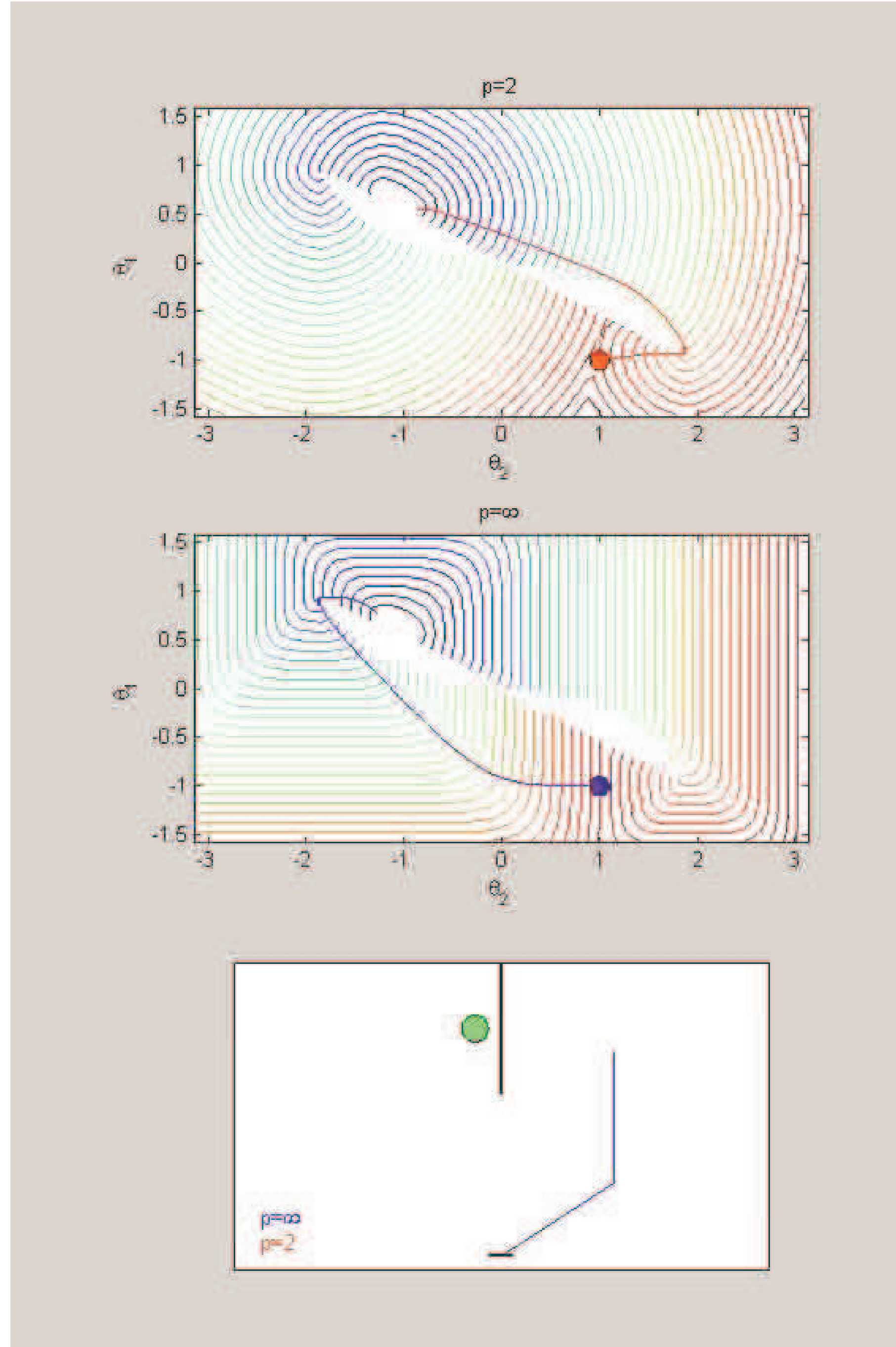
- We have thus far assumed $||\cdot||_2$ bound, but it is not always best
- For example: robot arm with joint angle state space
  - All joints may move independently at maximum speed: $||\cdot||_\infty$
  - Total power drawn by all joints is bounded: $||\cdot||_1$
- If action is bounded in $||\cdot||_p$, then value function is solution of "Eikonal" equation $||\vartheta(x)||_{p^*} = c(x)$ in the dual norm $p^*$
  - $p = 1$ and $p = \infty$ are duals, and $p = 2$ is its own dual
- Straightforward to derive update equations for $p = 1$, $p = \infty$



# Infinity Norm

- Paths may be very different when bounded in other norms
- Right: optimal trajectory of two joint arm under $||\cdot||_2$ (red) and $||\cdot||_\infty$ (blue)
- Below: one joint and slider arm under $||\cdot||_\infty$



# Mixtures of Norms: Multiple Vehicles

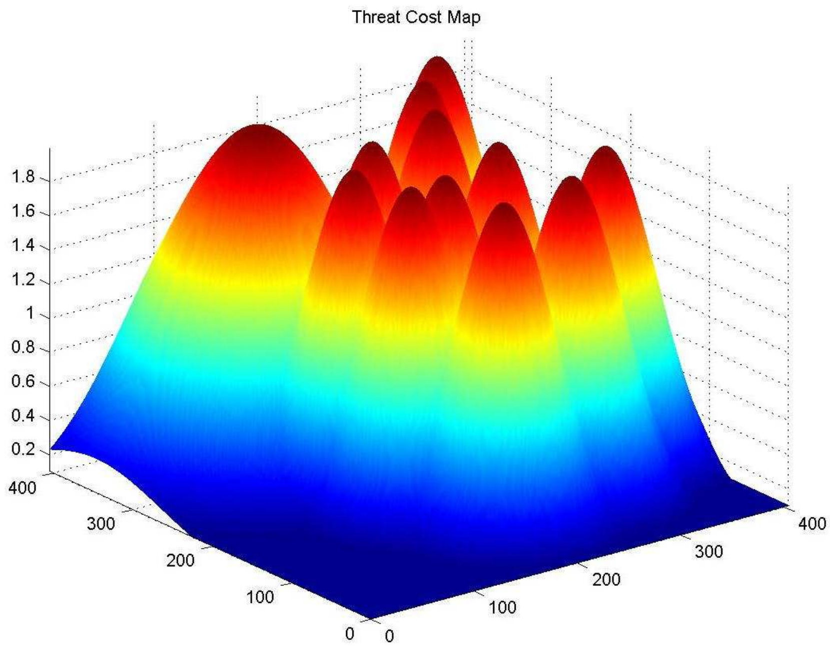
- May even be situations where action norm bounds are mixed
  - Red robot starts on right, may move any direction in 2D
  - Blue robot starts on left, constrained to 1D circular path
  - Cost encodes black obstacles and collision states
  - 2D robot action constrained in $||\cdot||_2$ and combined action in $||\cdot||_\infty$

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

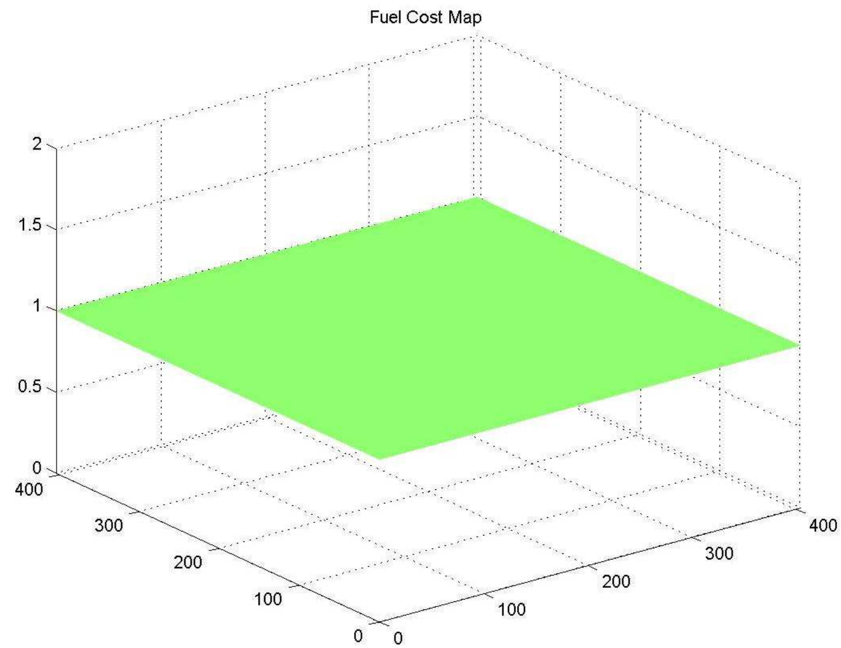
# Constrained Path Planning

- Input includes multiple cost functions $c_i(x)$
- Possible goals:
  - Find feasible paths given bounds on each cost
  - Optimize one cost subject to bounds on the others
  - Given a feasible/optimal path, determine marginals of the constraining costs

Variable cost (eg threat level)

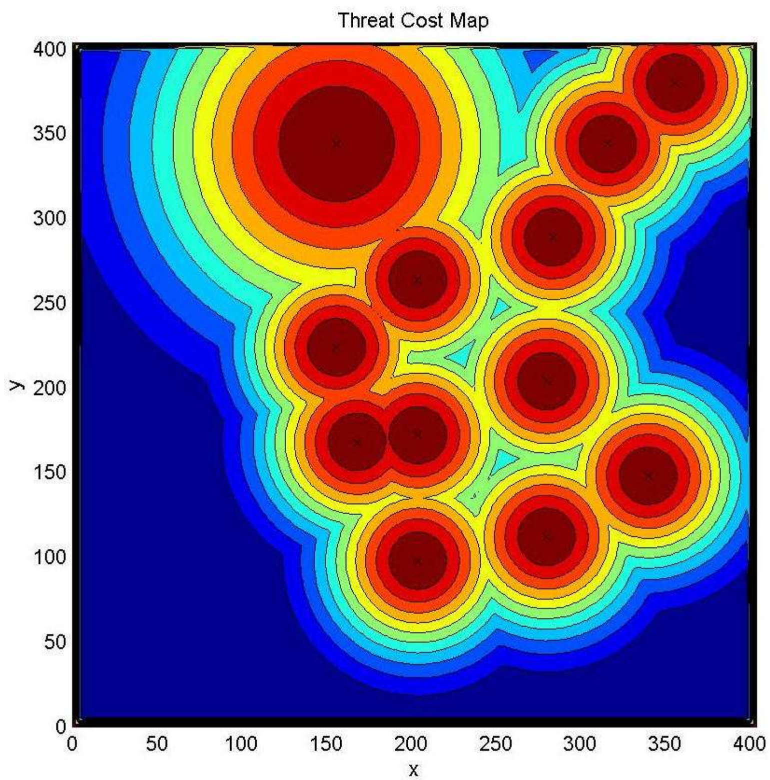



Constant cost (eg fuel)

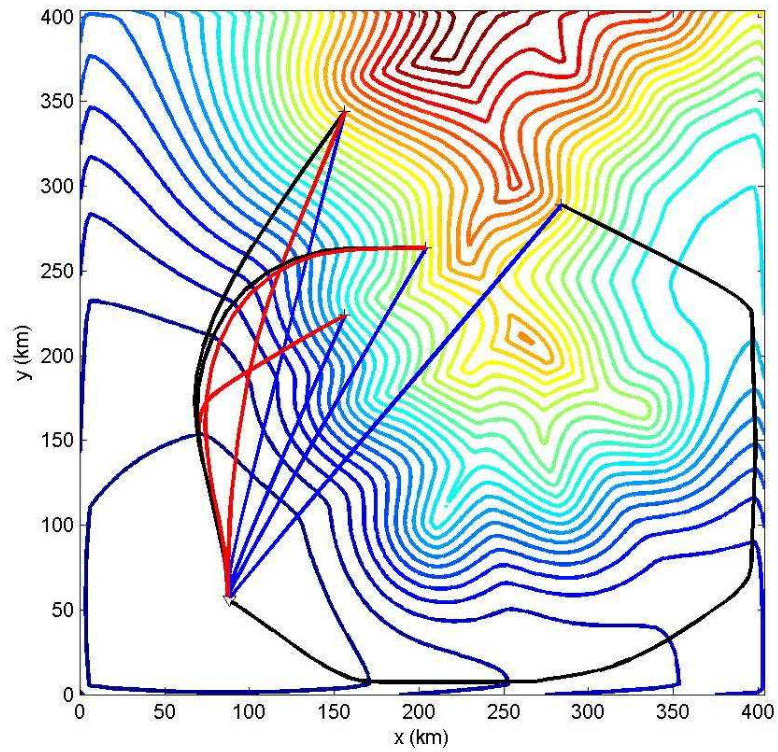


# Constrained Example

- Plan path to selected sites
  - Threat cost function is maximum of individual threats
- For each target, plan 3 paths
  - minimum threat, minimum fuel, minimum threat (with fuel ≤ 300)




threat cost




Paths (on value function)