

Lecture 7:

Potential Fields and

Model Predictive Control

CS 344R: Robotics

Benjamin Kuipers

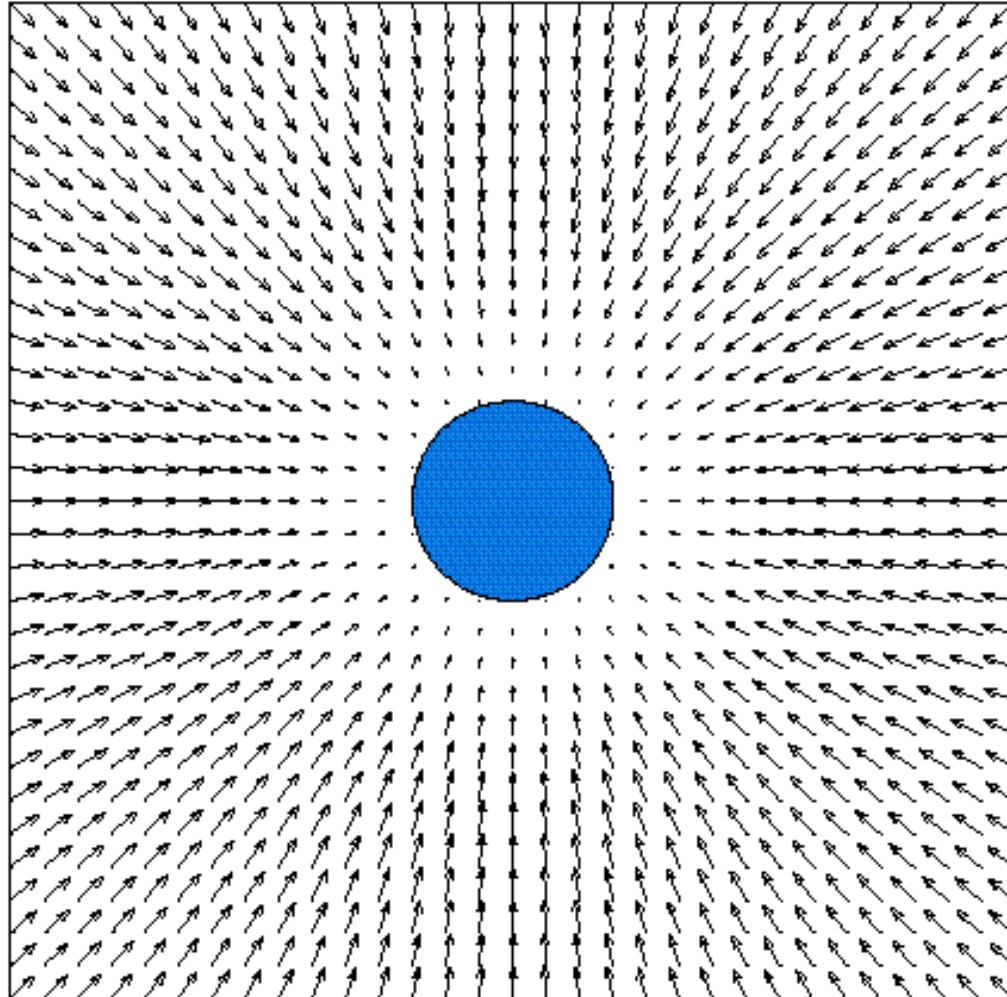
Potential Fields

- Oussama Khatib, 1986.

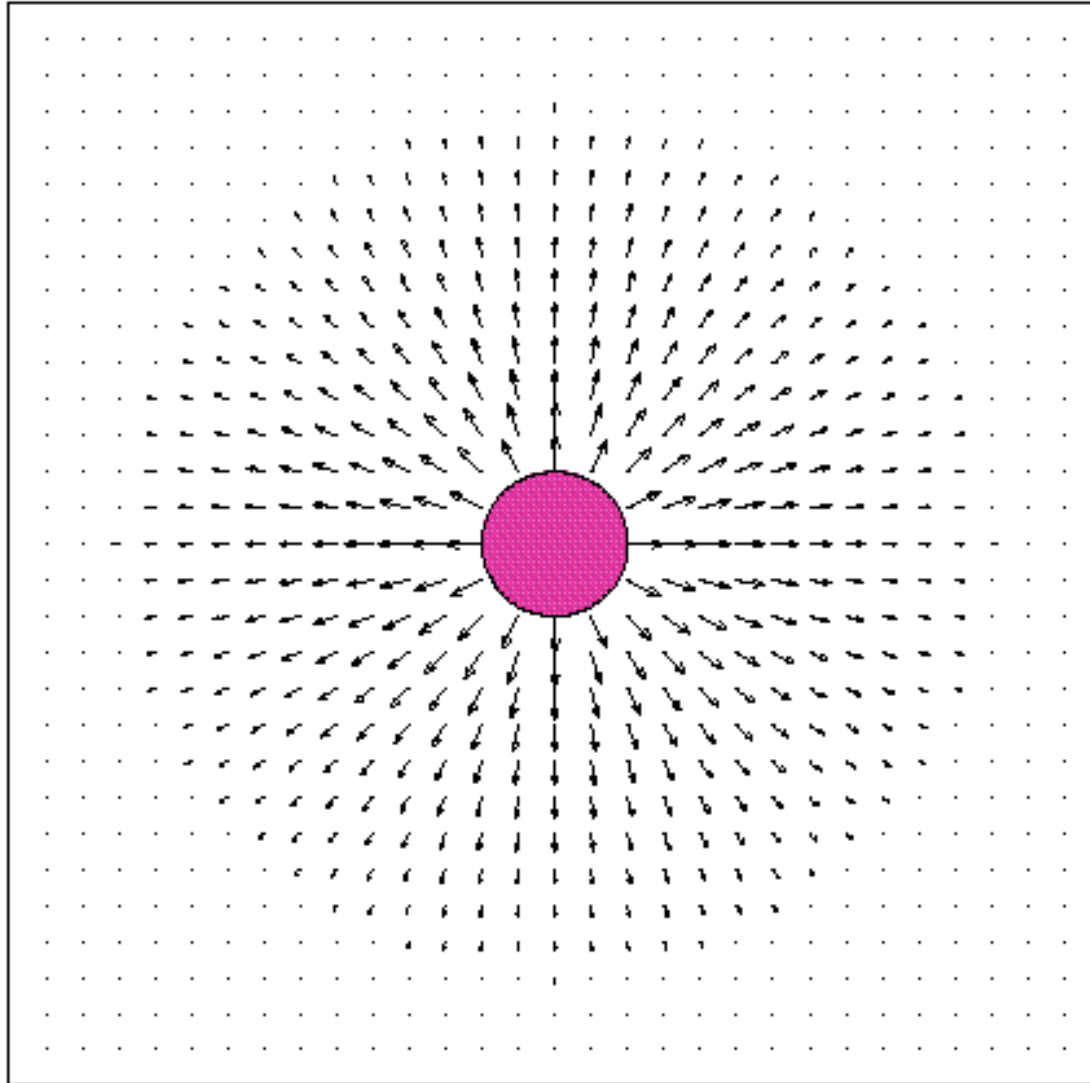
The manipulator moves in a field of forces.

The position to be reached is an attractive pole for the end effector and obstacles are repulsive surfaces for the manipulator parts.

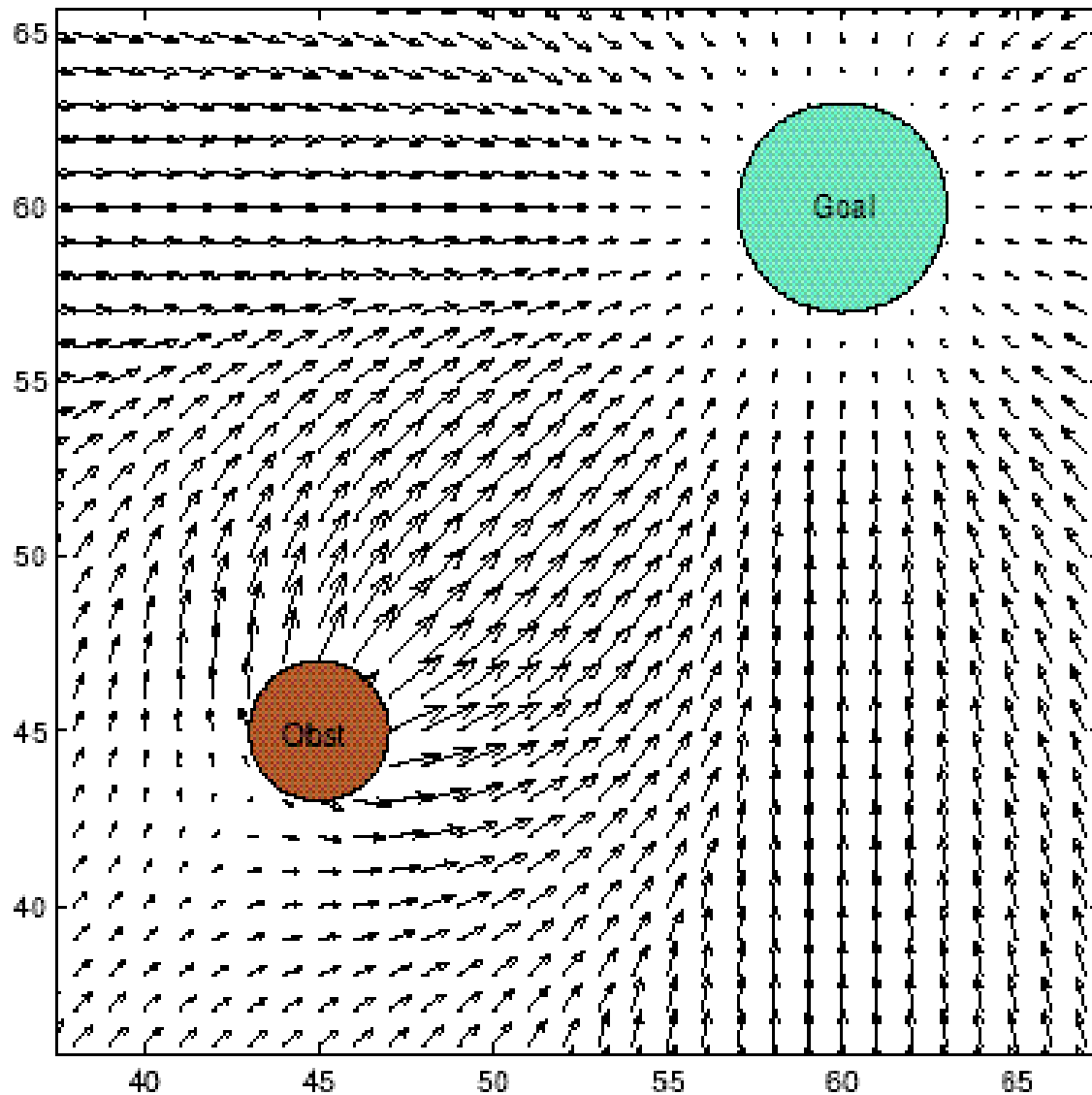
Attractive Potential Field



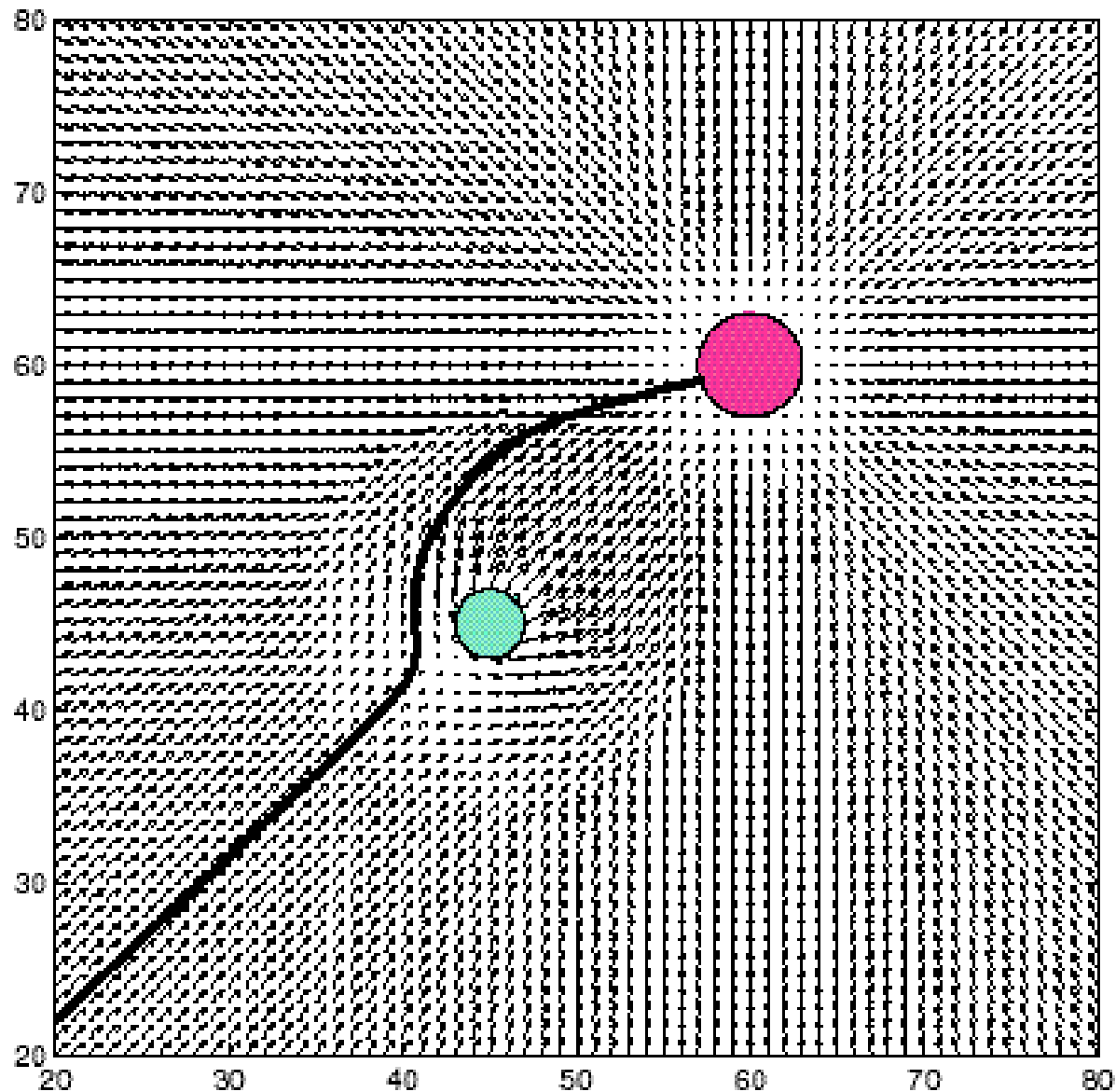
Repulsive Potential Field



Vector Sum of Two Fields



Resulting Robot Trajectory



Potential Fields

- Control laws meant to be added together are often visualized as vector fields:

$$(x, y) \rightarrow (\Delta x, \Delta y)$$

- In some cases, a vector field is the *gradient* of a potential function $P(x, y)$:

$$(\Delta x, \Delta y) = \nabla P(x, y) = \left(\frac{\partial P}{\partial x}, \frac{\partial P}{\partial y} \right)$$

Potential Fields

- The potential field $P(\mathbf{x})$ is defined over the environment.
- Sensor information \mathbf{y} is used to estimate the potential field gradient $\nabla P(\mathbf{x})$
 - No need to compute the entire field.
 - Compute individual components separately.
- The motor vector \mathbf{u} is determined to follow that gradient.

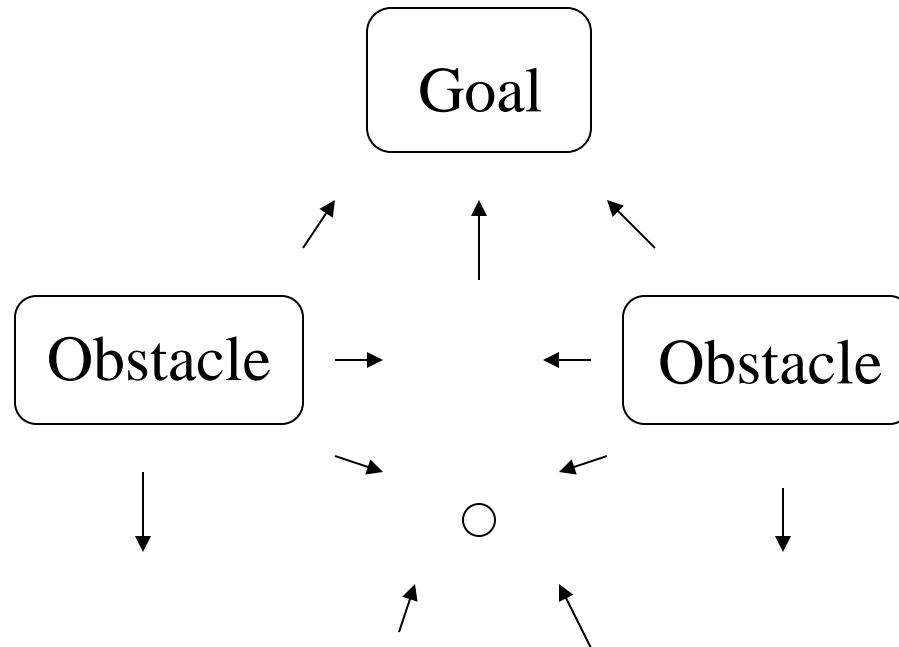
Attraction and Avoidance

- **Goal:** Surround with an attractive field.
- **Obstacles:** Surround with repulsive fields.
- *Ideal result:* move toward goal while avoiding collisions with obstacles.
 - Think of rolling down a curved surface.
- *Dynamic obstacles:* rapid update to the potential field avoids moving obstacles.

Potential Problems with Potential Fields

- *Local minima*
 - Attractive and repulsive forces can balance, so robot makes no progress.
 - Closely spaced obstacles, or dead end.
- *Unstable oscillation*
 - The dynamics of the robot/environment system can become unstable.
 - High speeds, narrow corridors, sudden changes.

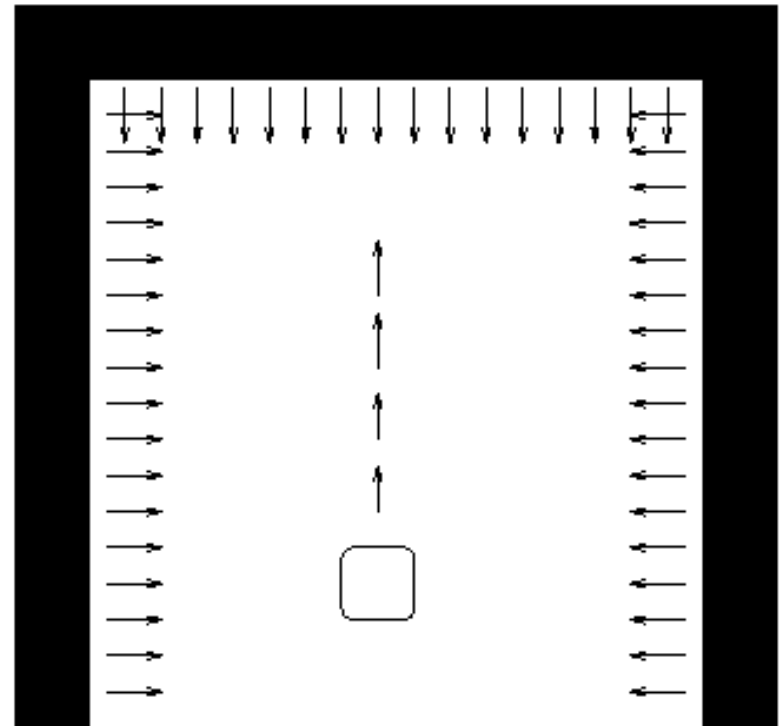
Local Minimum Problem



Box Canyon Problem

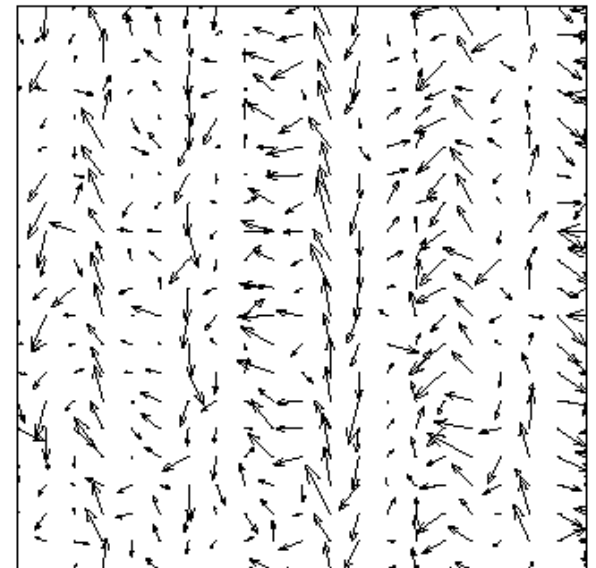
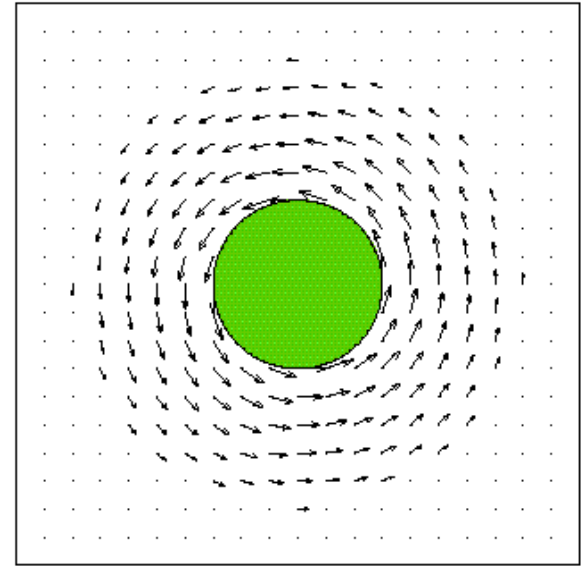


- Local minimum problem, or
- *AvoidPast* potential field.



Rotational and Random Fields

- Not gradients of potential functions
- Adding a *rotational field* around obstacles
 - Breaks symmetry
 - Avoids some local minima
 - Guides robot around groups of obstacles
- A *random field* gets the robot unstuck.
 - Avoids some local minima.



Vector Field Histogram: Fast Obstacle Avoidance

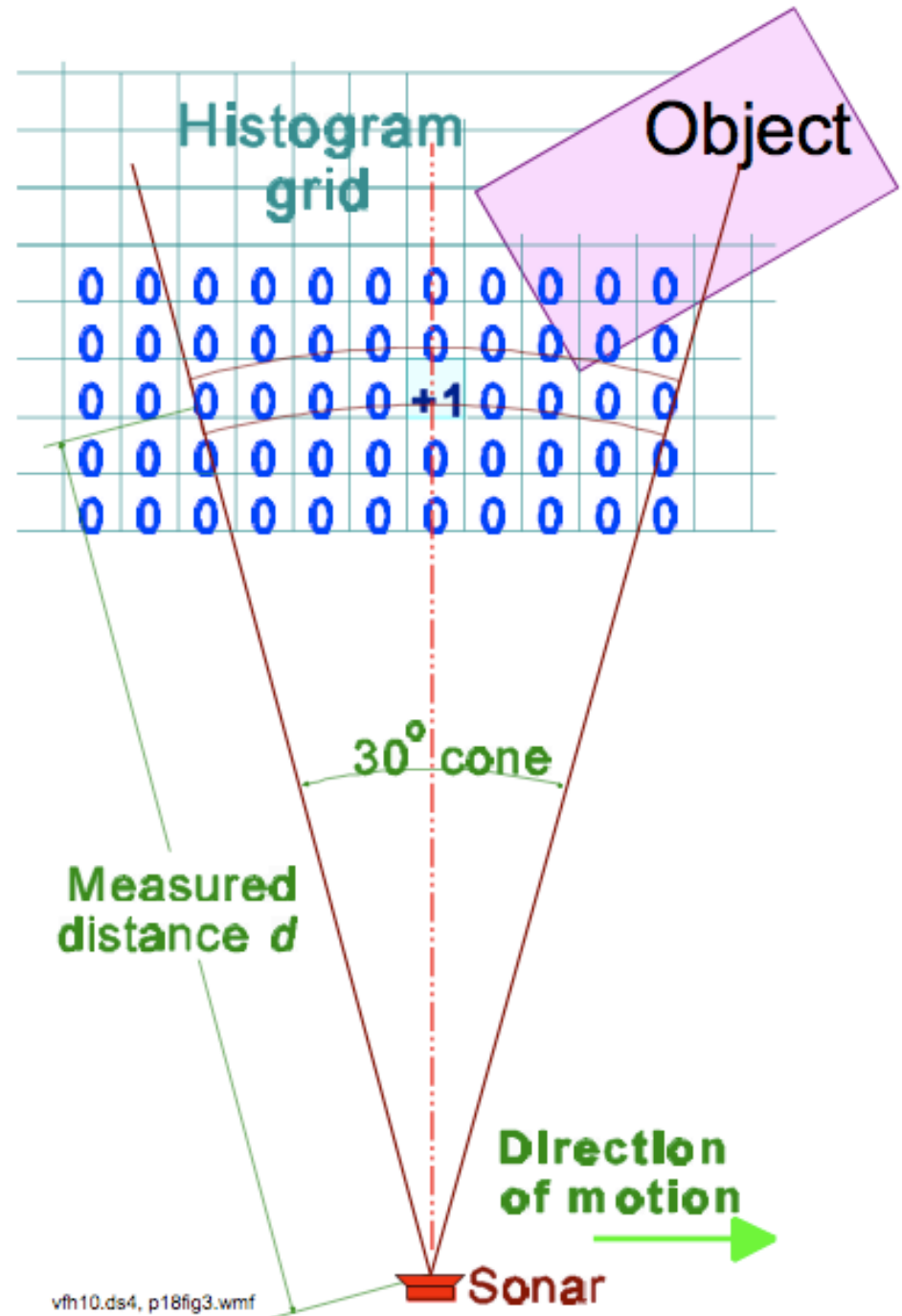
- Build a local occupancy grid map
 - Confined to a scrolling active window
 - Use only a single point on axis of sonar beam
- Build a polar histogram of obstacles
 - Define directions for safe travel
- Steering control
 - Steer midway between obstacles
 - Make progress toward the global target

CARMEL: Cybermotion K2A



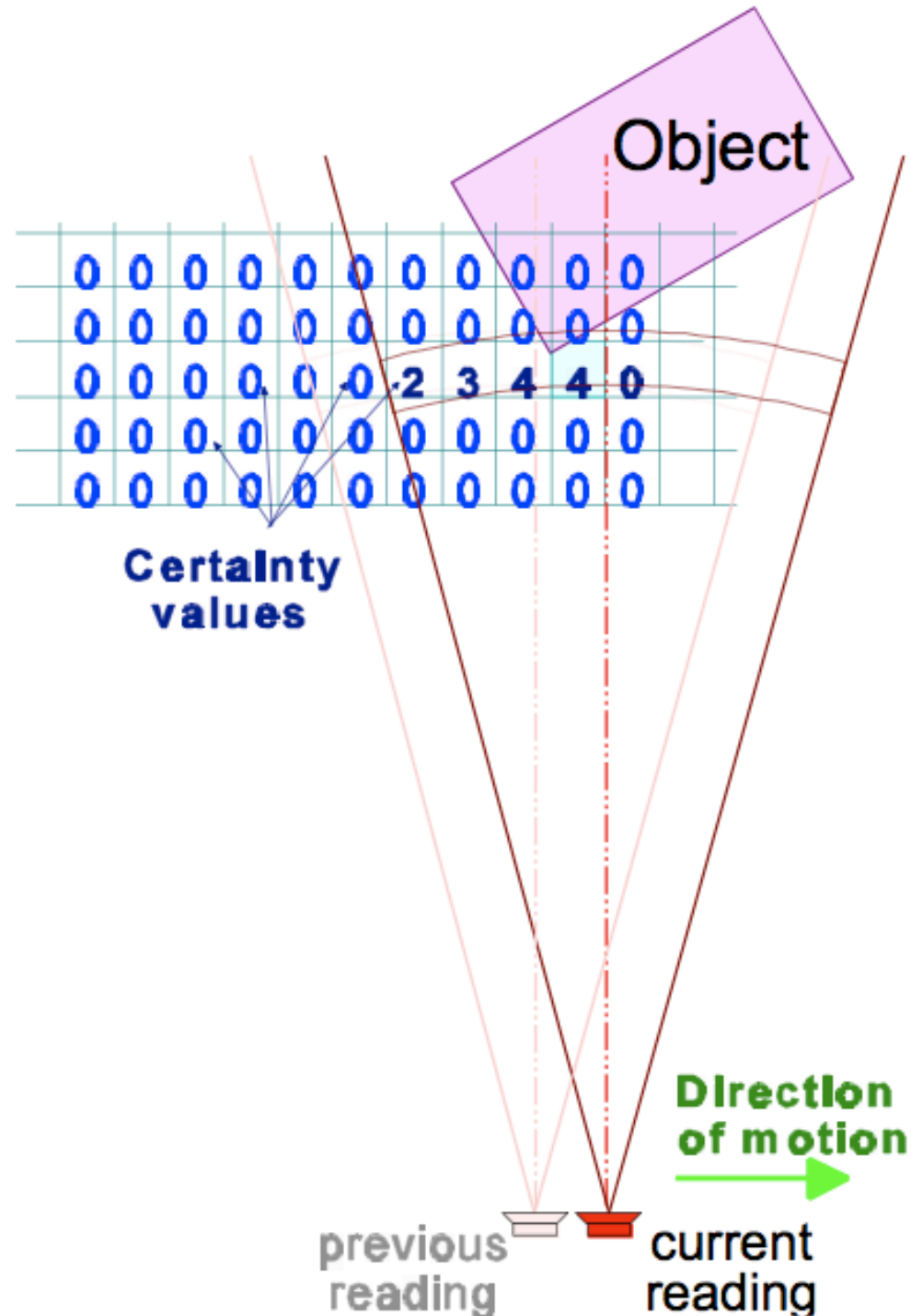
Occupancy Grid

- Given sonar distance d
- Increment single cell along axis
- (Ignores data from rest of sonar cone)



Occupancy Grid

- Collect multiple sensor readings
- Multiple readings substitutes for unsophisticated sensor model.

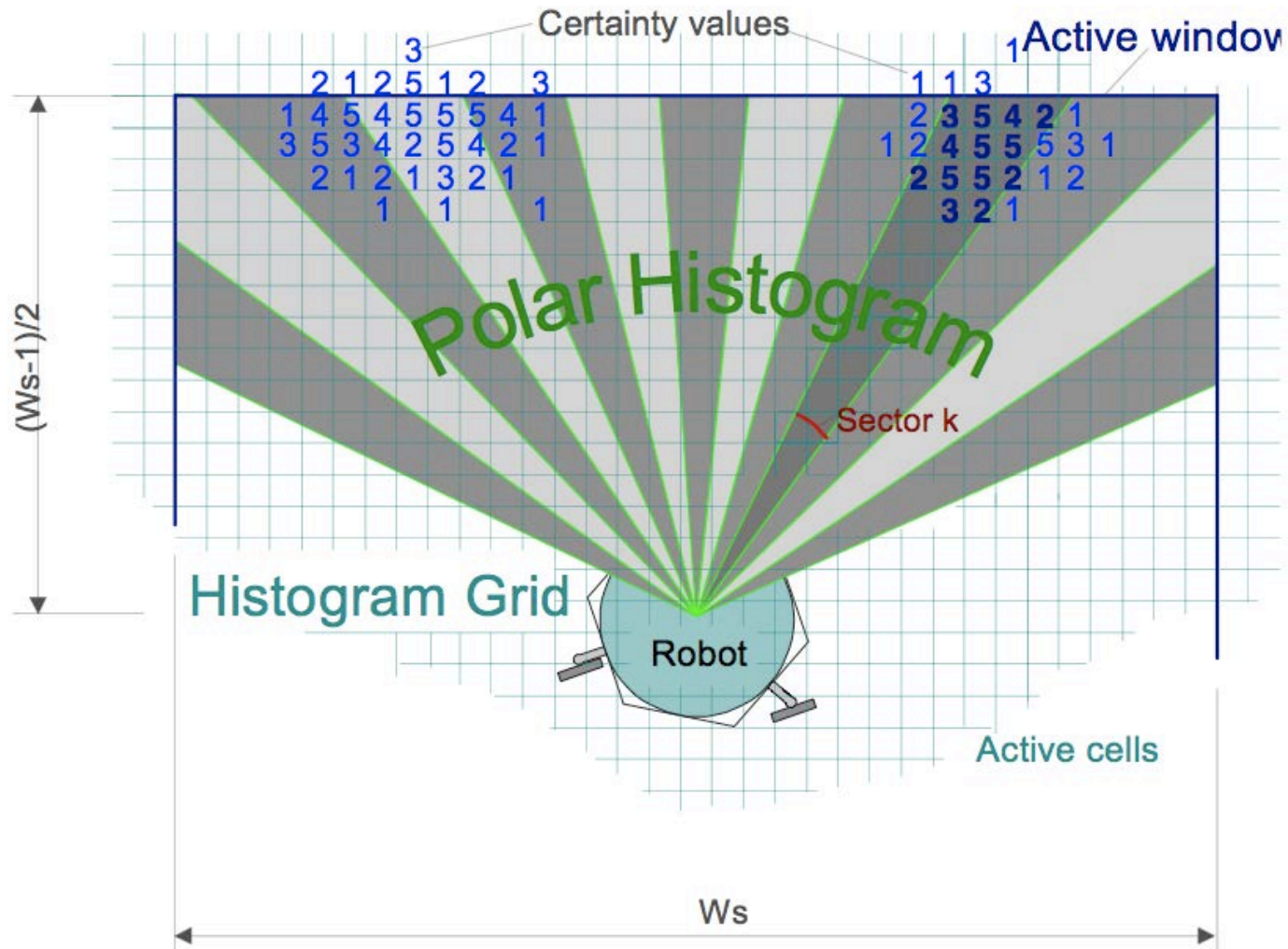


The diagram illustrates a histogram grid for robot navigation. A robot is positioned at the center of the grid. An active window is defined around the robot, and the grid is divided into cells. Certainty values are assigned to cells, with higher values indicating higher certainty. A target is shown, and the robot's path is indicated by dashed lines. Labels include: Histogram grid, W_s , Active window, Object, Certainty values, Active cells, Robot, Target, R , F_t , and F_f .

- Active window $W_s \times W_s$ around the robot
- Grid alone used to define a "virtual force field"

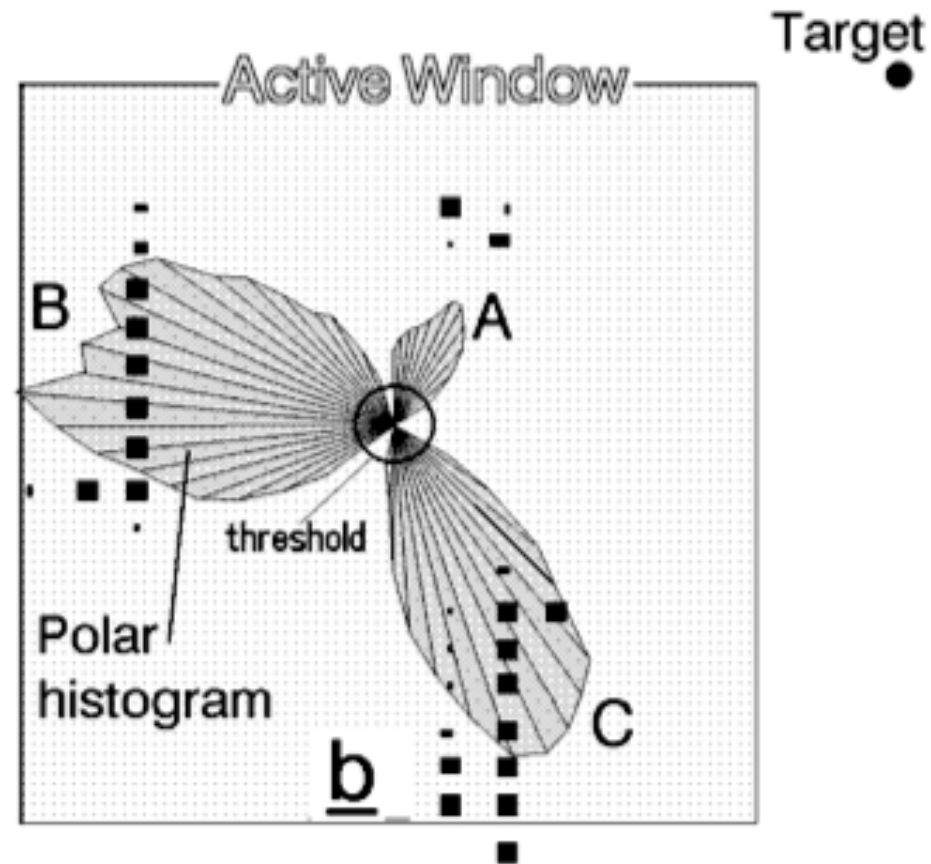
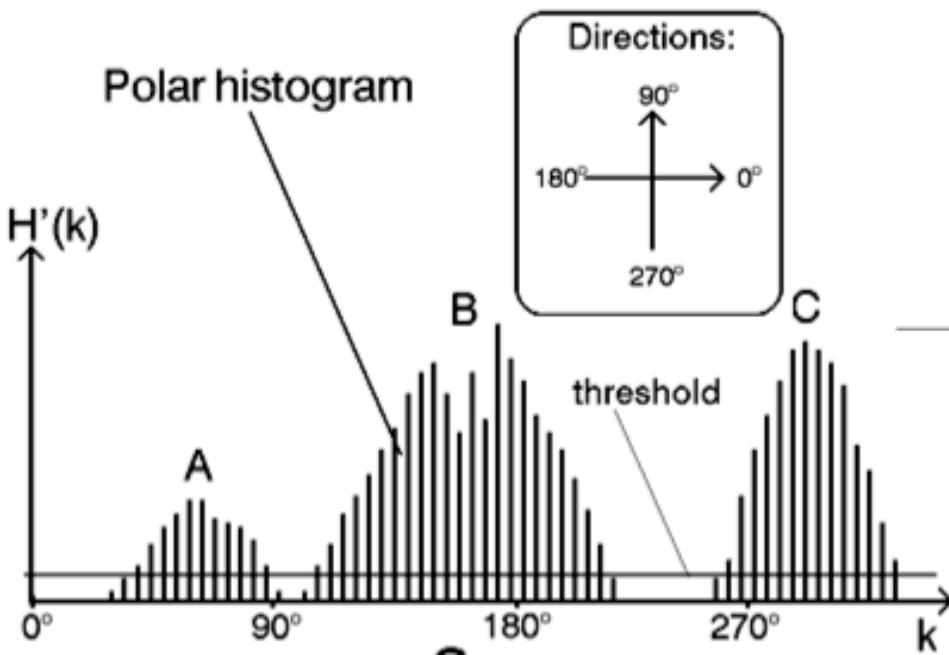
Polar Histogram

- Aggregate obstacles from occupancy grid according to direction from robot.



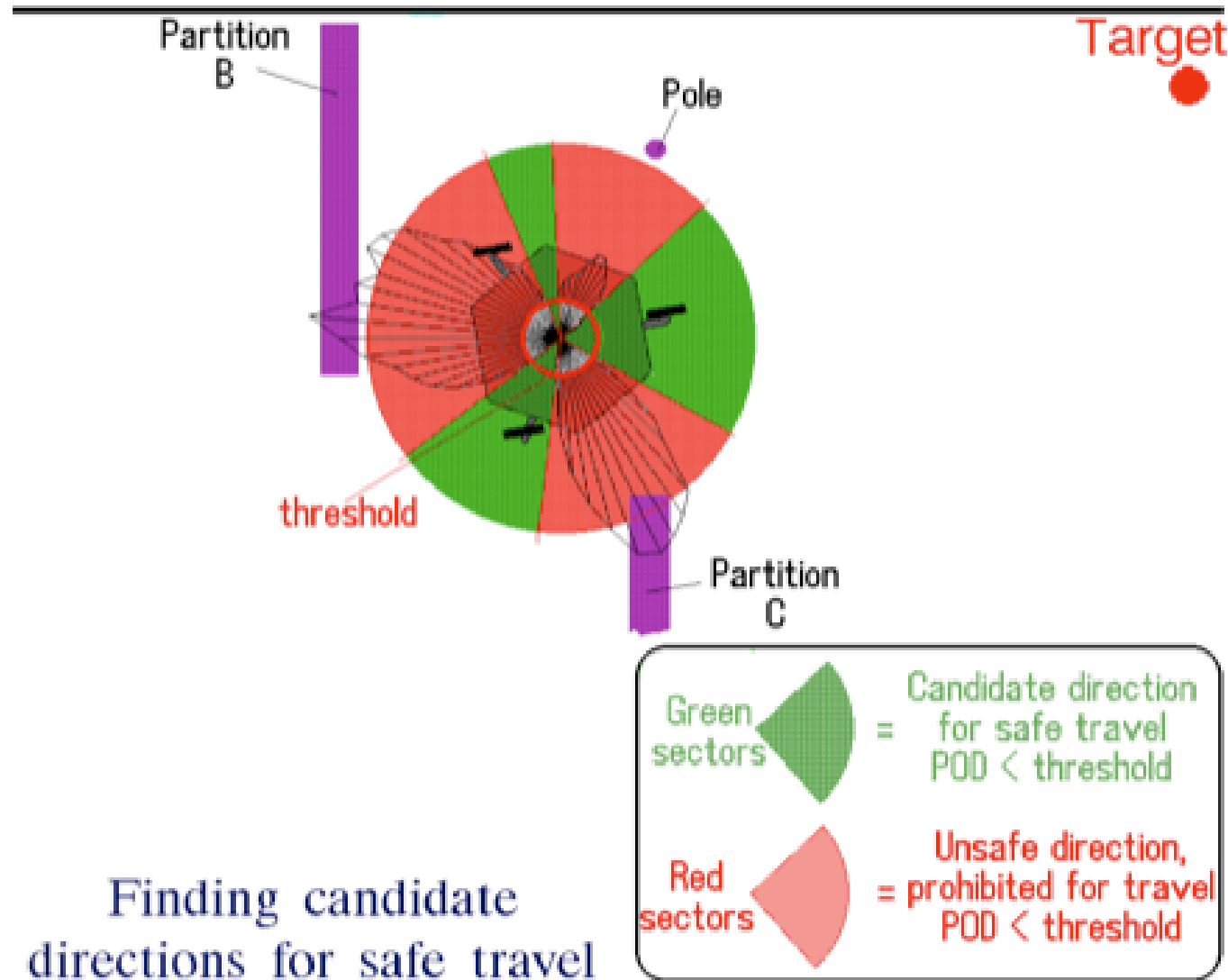
Polar Histogram

- Weight by occupancy, and inversely by distance.

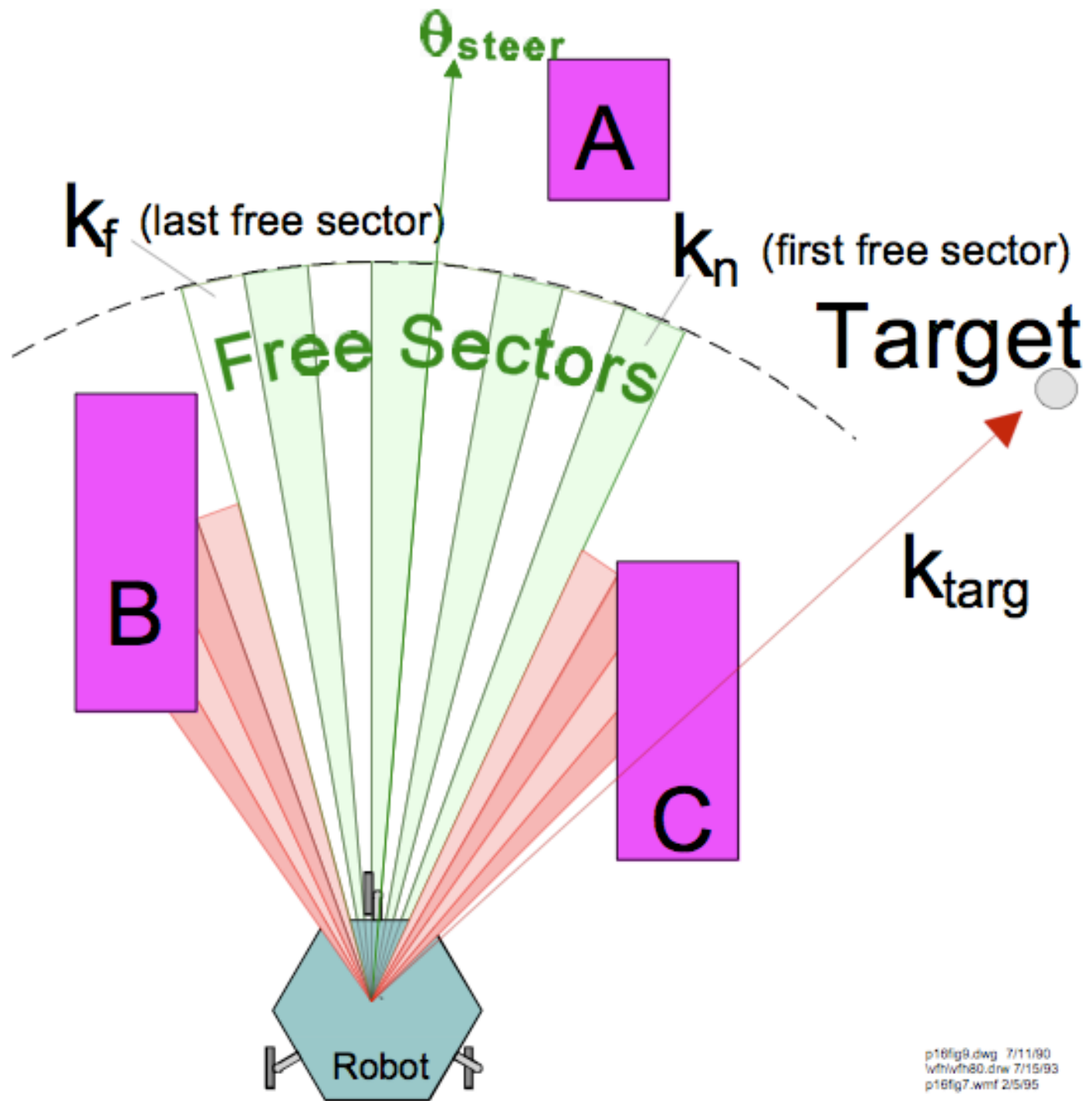


Directions for Safe Travel

- Threshold determines safety.
- Multiple levels of noise elimination.

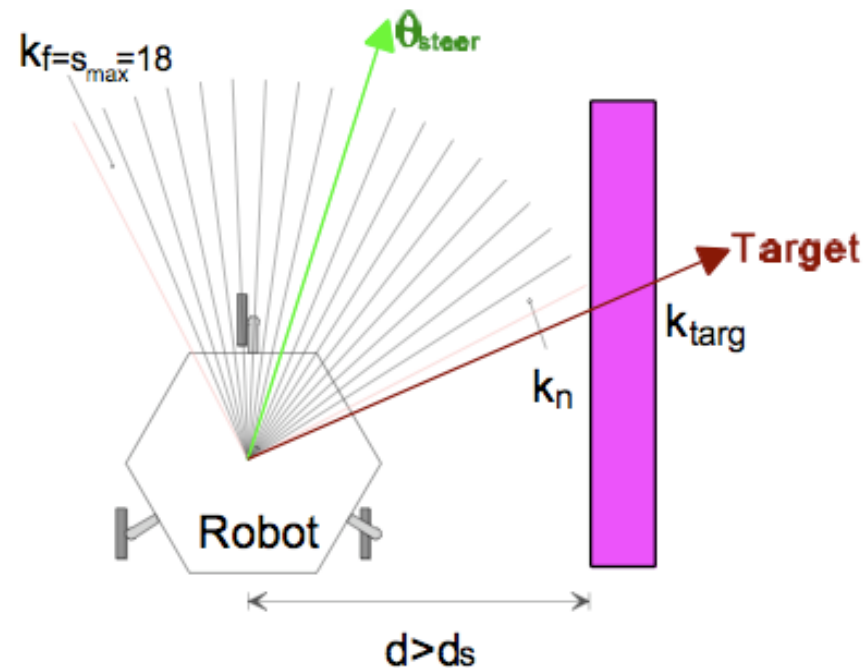
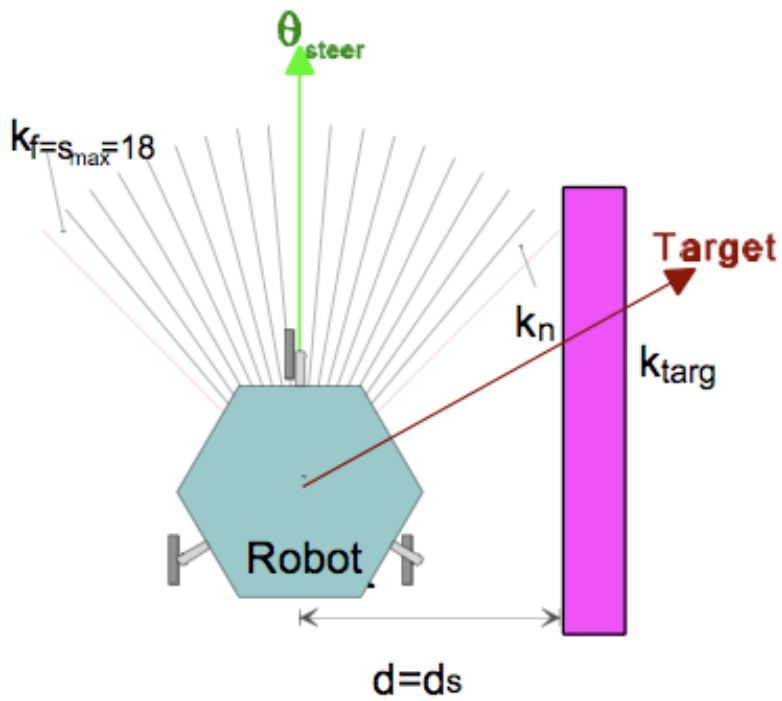
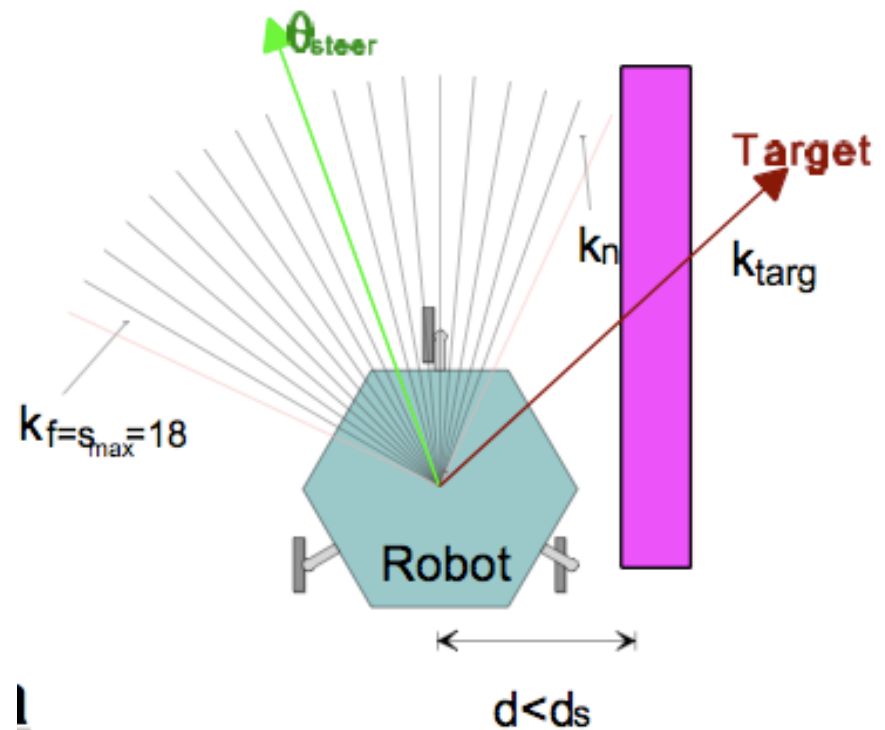


Steer to
center of
safe
sector



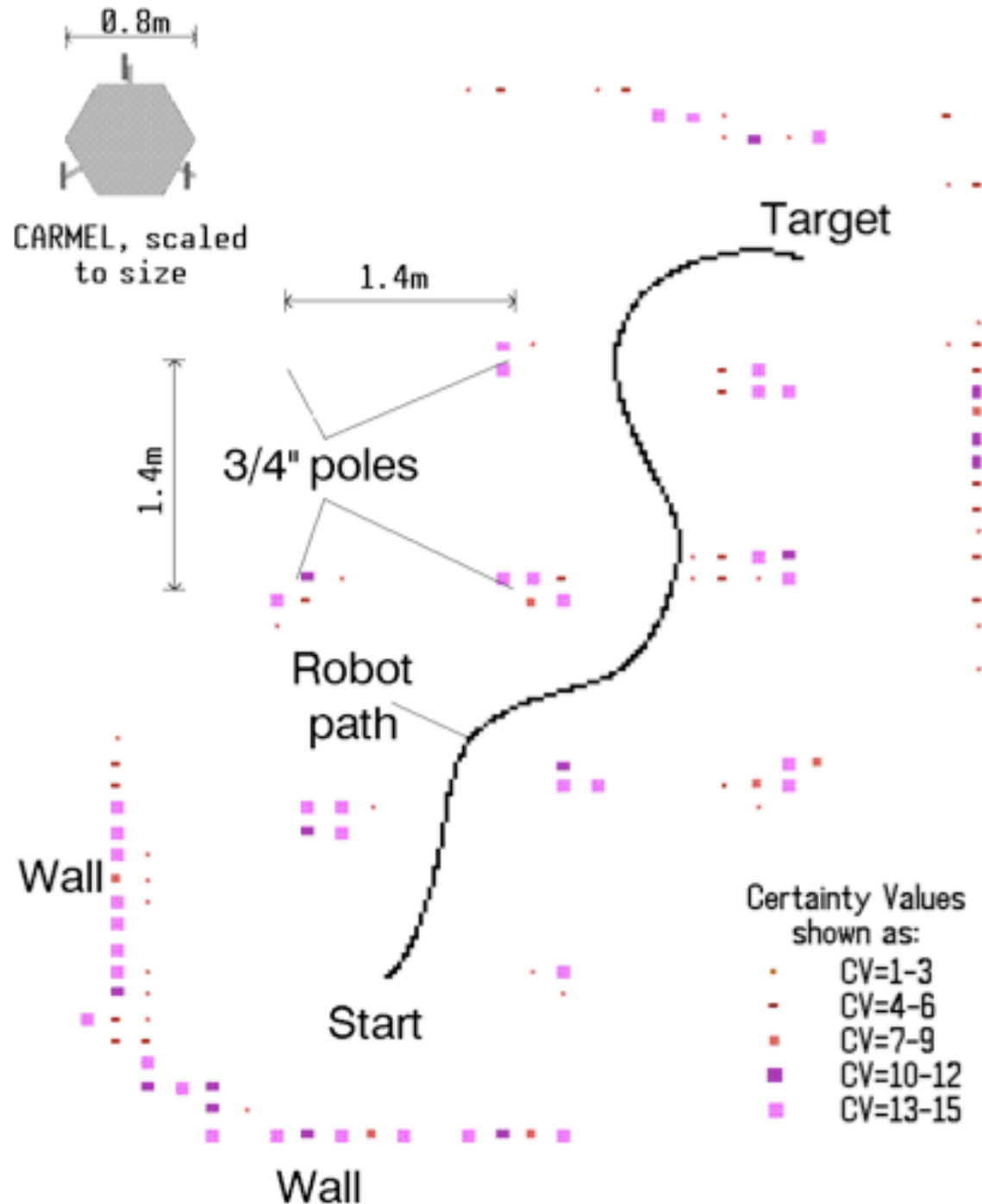
Leads to natural wall-following

- Threshold determines offset from wall.



Smooth, Natural Wandering Behavior

- Potentially quite fast!
- 1 m/s or more!



Konolige's Gradient Method

- A *path* is a sequence of points:

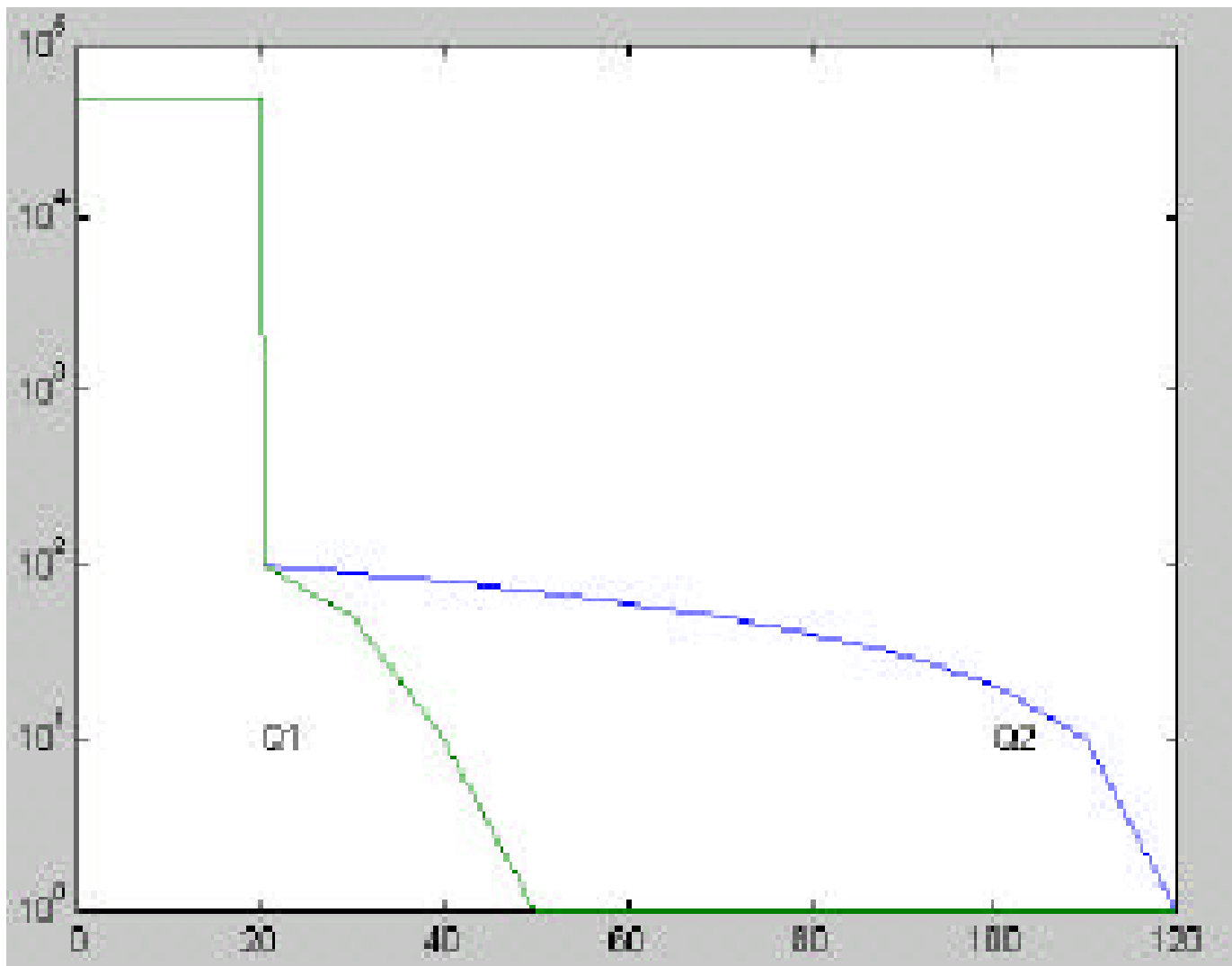
- $P = \{p_1, p_2, p_3, \dots\}$

- The *cost* of a path is

$$F(P) = \sum_i I(p_i) + \sum_i A(p_i, p_{i+1})$$

- Intrinsic cost $I(p_i)$ handles obstacles, etc.
- Adjacency cost $A(p_i, p_{i+1})$ handles path length.

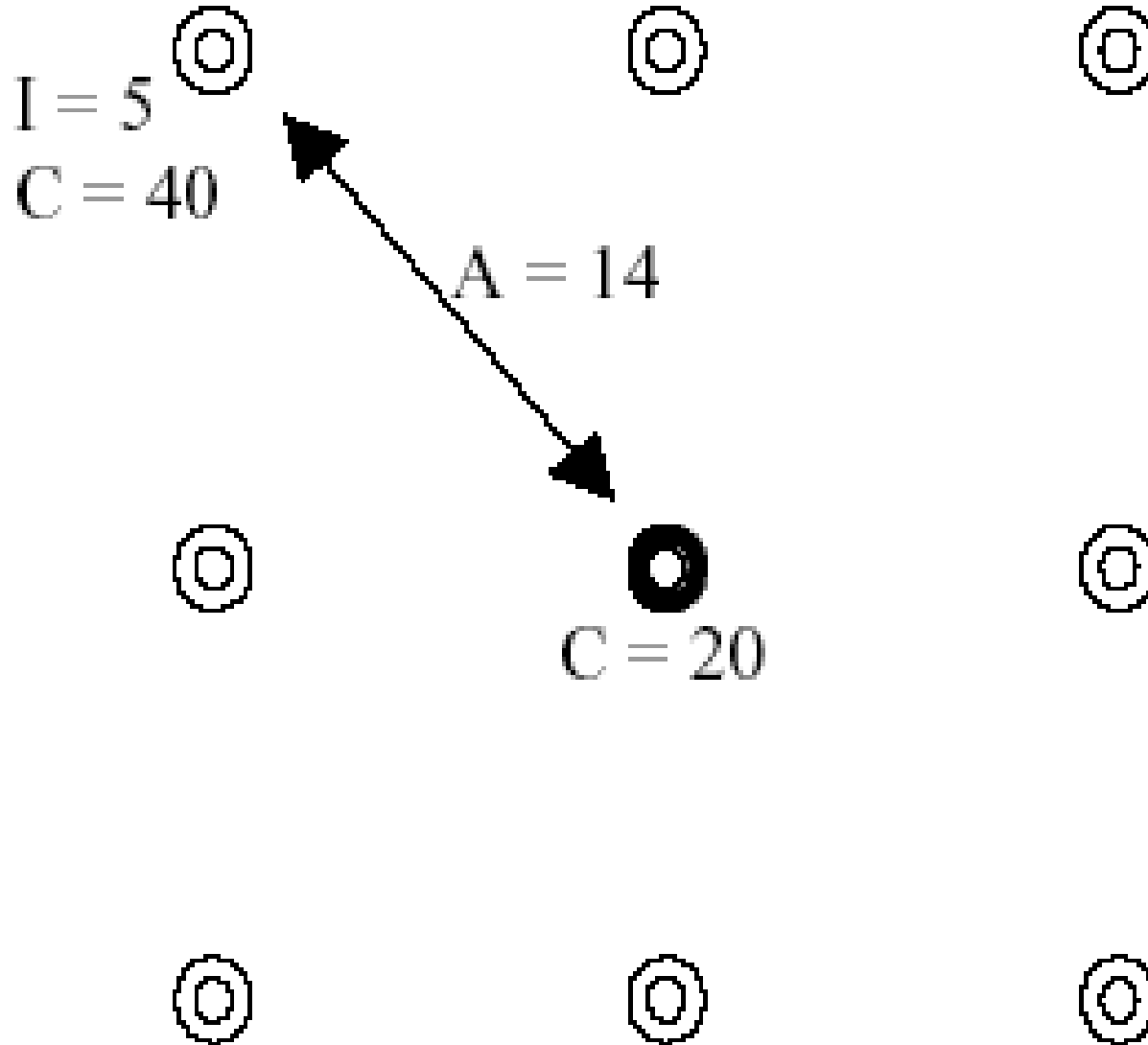
Intrinsic Cost Functions $I(p)$

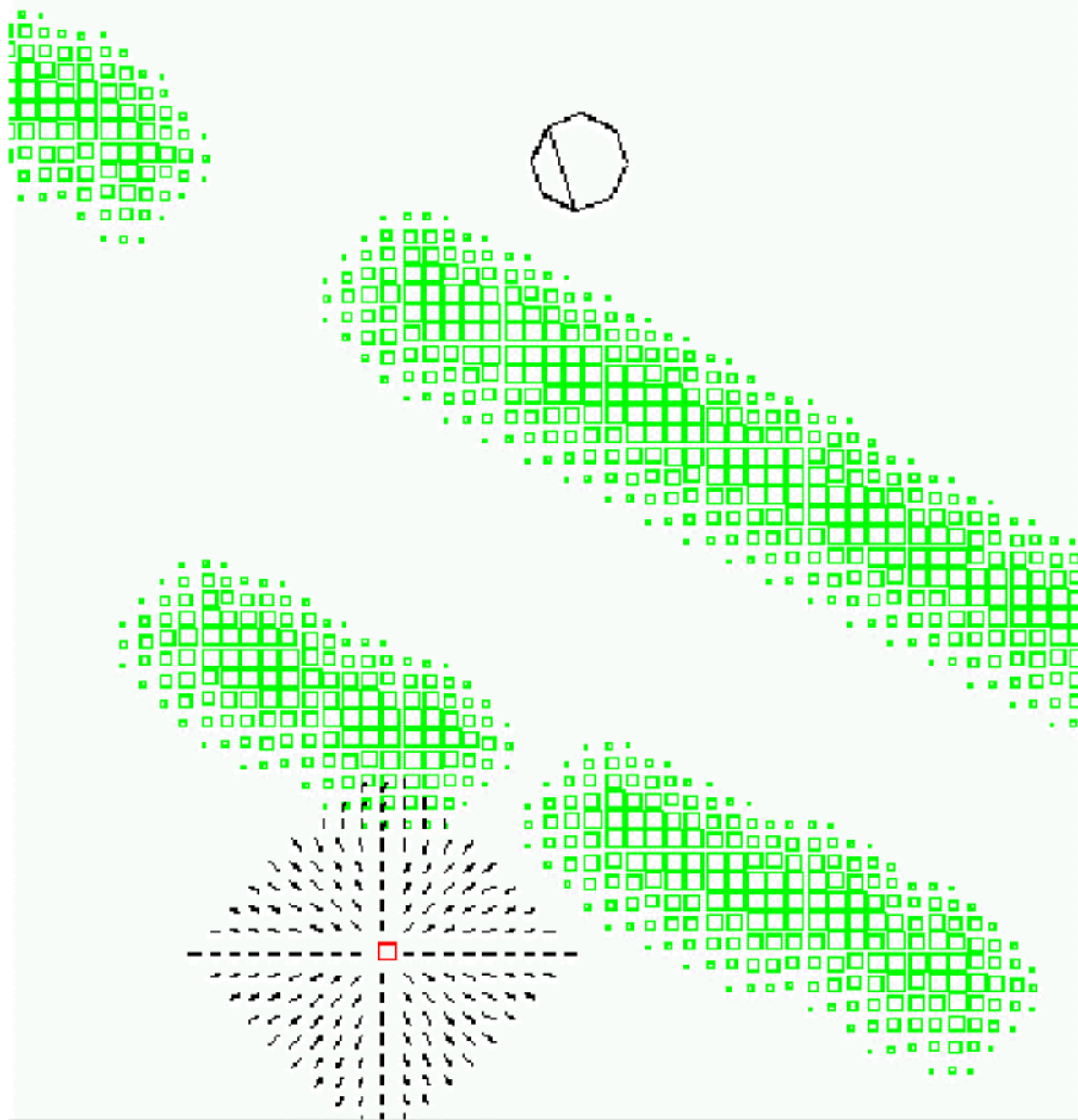


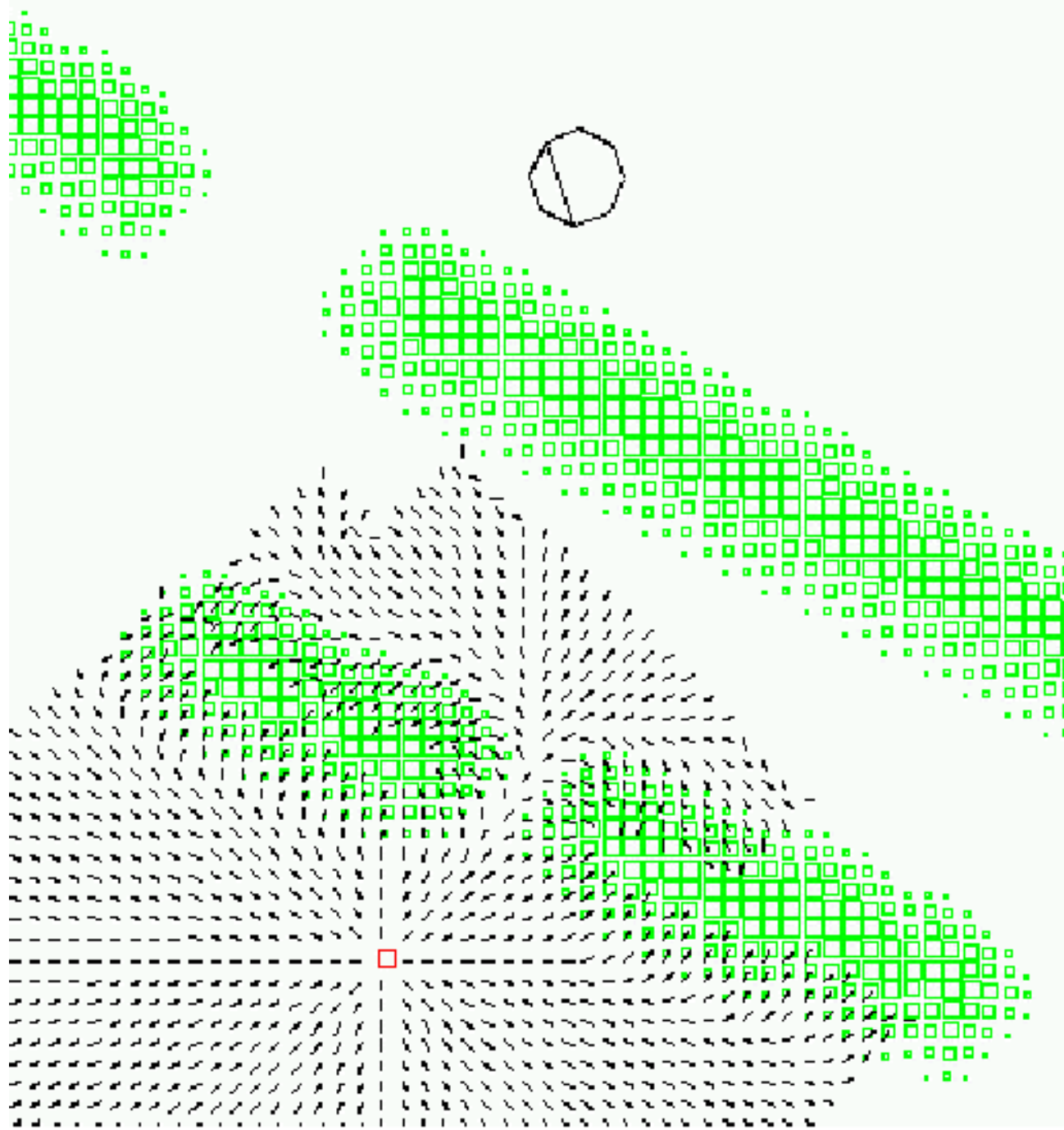
Navigation Function $N(p)$

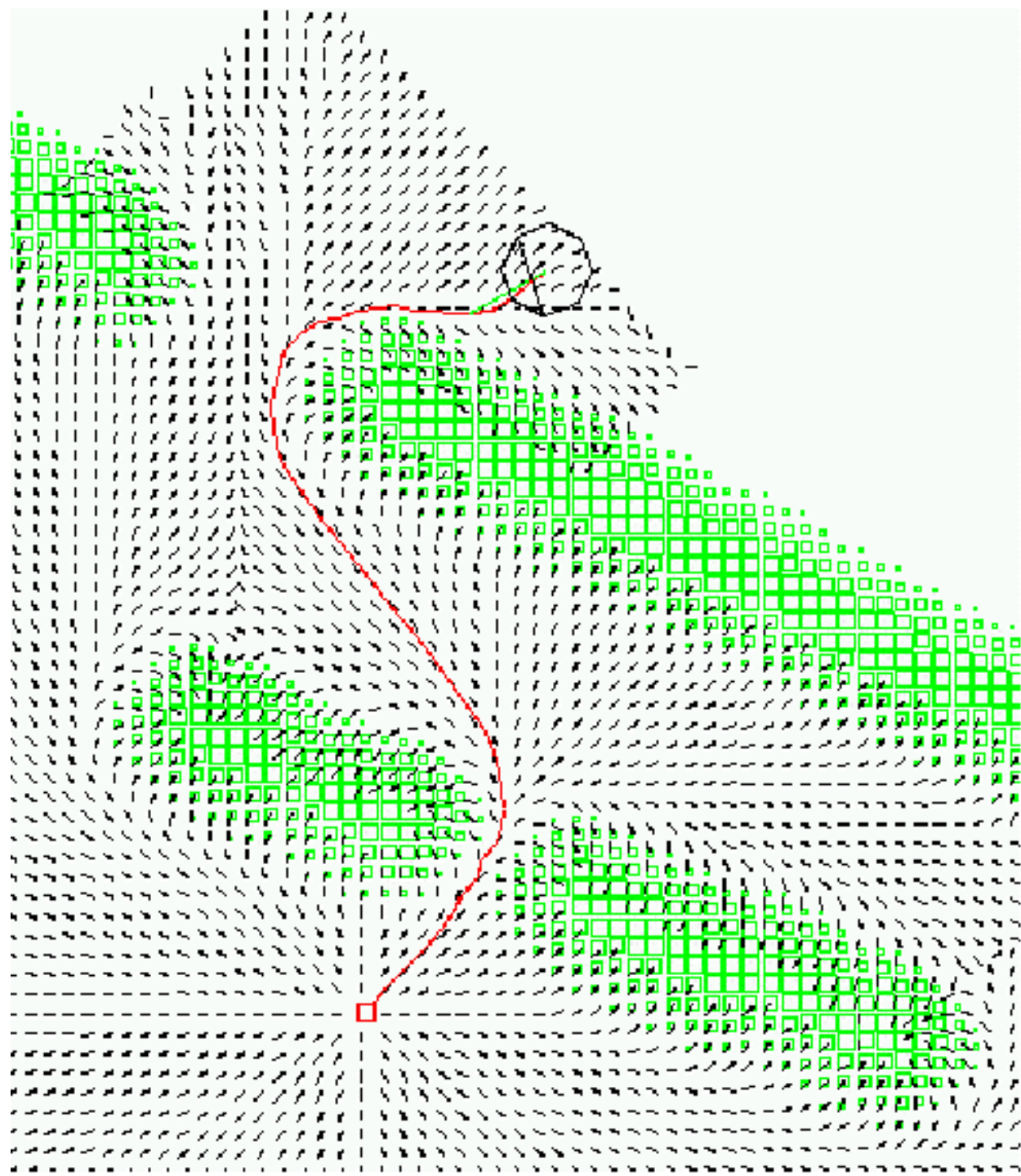
- A potential field leading to a given goal, with no local minima to get stuck in.
- For any point p , $N(p)$ is the minimum cost of any path to the goal.
- Use a *wavefront* algorithm, propagating from the goal to the current location.
 - An active point updates costs of its 8 neighbors.
 - A point becomes active if its cost decreases.
 - Continue to the robot's current position.

Wavefront Propagation









Real-Time Control

- Recalculates $N(p)$ at 10 Hz
 - (on a 266 MHz PC!)
- Handles dynamic obstacles by recalculating.
 - Cannot anticipate a collision course.
- Much faster and safer than a human operator on a comparison experiment.
- Requirements:
 - an accurate map, and
 - accurate robot localization in the map.

Model-Predictive Control

- Replan the route on each cycle (10 Hz).
 - Update the map of obstacles.
 - Recalculate $N(p)$. Plan a new route.
 - Take the first few steps.
 - Repeat the cycle.
- Obstacles are always treated as static.
 - The map is updated at 10 Hz, so the behavior looks like dynamic obstacle avoidance, even without dynamic prediction.

Plan Routes in the Local Perceptual Map

- The LPM is a *scrolling map*, so the robot is always in the center cell.
 - Shift the map only by integer numbers of cells
 - Variable heading θ .
- Sensor returns specify occupied regions of the local map.
- Select a goal near the edge of the LPM.
- Propagate the $N(p)$ wavefront from that goal.

Searching for the Best Route

- The wavefront algorithm considers all routes to the goal with the same cost $N(p)$.
- The A^* algorithm considers all routes with the same cost plus predicted completion cost $N(p) + h(p)$.
 - A^* is provably complete and optimal.

QuickTime™ and a
decompressor
are needed to see this picture.