CSCI444/944

Perception and Planning

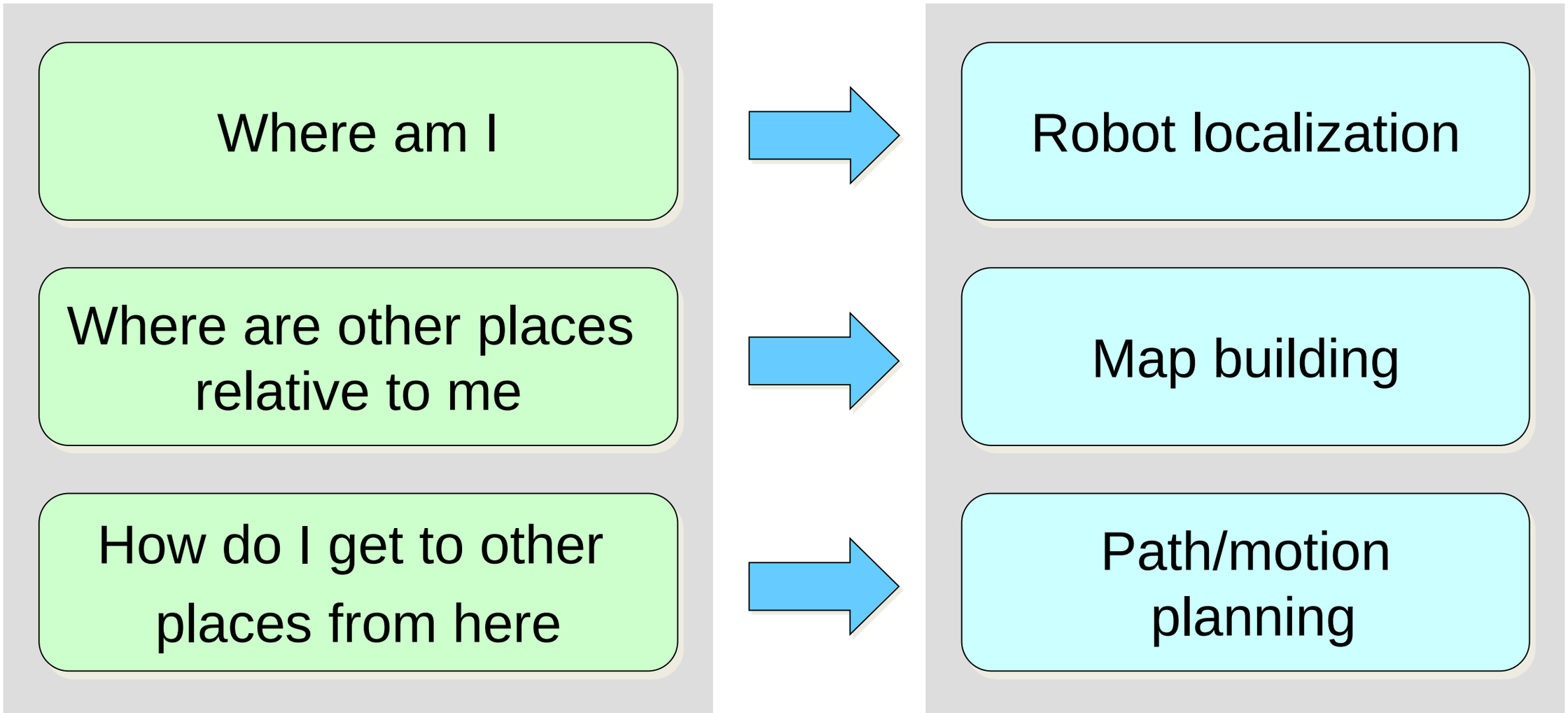
Week 5

Autonomous Navigation

Navigation Issues

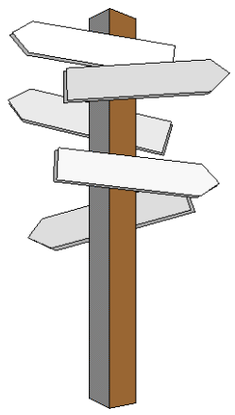
Important navigation questions

Important navigation issues

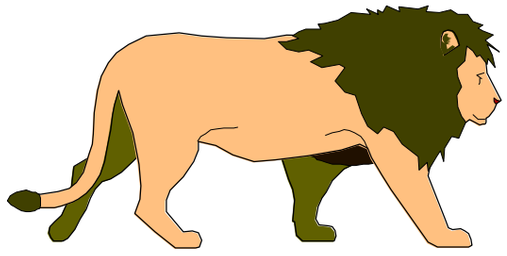


Human Navigation

- Global
  - Map-Based
  - Deliberative
  - Slow



- Local
  - Sensory-Based
  - Reactive
  - Fast



Human Navigation

- Humans use various means of navigating the environment.
- Examples:
  - Remembering landmarks
  - Use other points of reference
    - Stars
    - Maps
  - Use navigational tools
    - GPS
    - Compass
  - etc…
- Examples . . .

Blind Woman

A lady with no measurable sight spoke at a conference

- Afterwards she caught a plane back to her home city
- She needed help to find her seat on the plane but was able to achieve most other actions unaided
- She was met at the airport by a friend
- Recognised him by his voice
- Followed him through the concourse pulling a bag behind her, without touching him
- She was able to perceive gross regions of light and dark but no visual details - technically blind

Blind Woman

When asked how she navigates with such a high level of autonomy she said going somewhere is a problem solving process:

- When I have to make a journey, even a short one, I mentally walk through the process.
- When satisfied that I have a satisfactory plan to reach the destination I set out.
- When the plan fails, I can become lost and embarrassed, so I just asks someone for directions toward my destination

Native Sailor

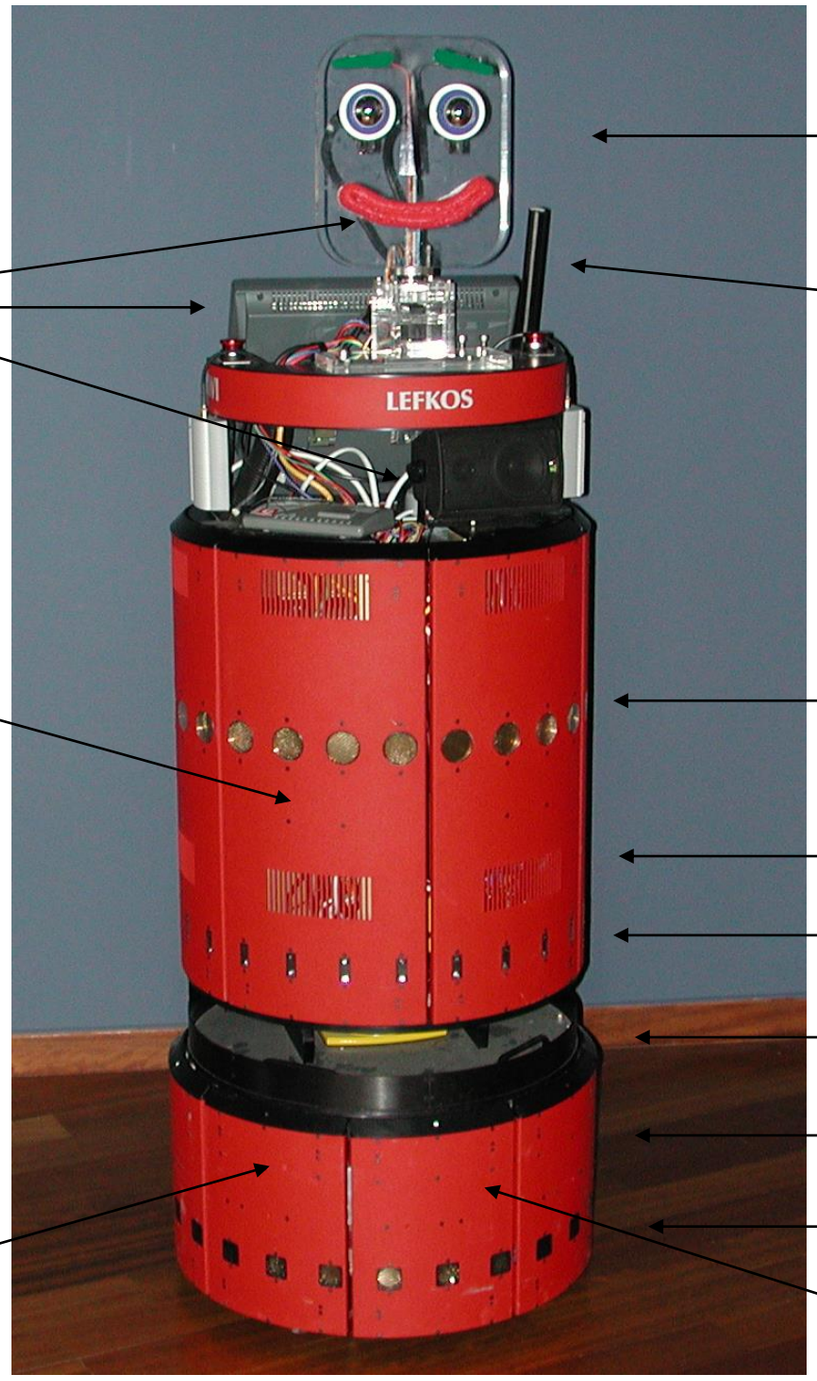
- A native from Puluwat Atoll sailing to another island:
  - The navigator imagines the journey in his mind: the destination island, the reefs along the way, the prevailing winds, the time of year, and their combined effect on the wave and cloud patterns, etc.
  - From his sense of where he is, where he is going and what is along the way he perceives a plan of action.
  - He then back sights using clumps of trees on the island as a reference to set his course.
  - As he navigates that course, he updates his plan based on his sense of his current location and the prevailing weather conditions.

European Sailor

European navigator setting out to make the same journey:

- The navigator plots his course on his nautical charts of the area
- As he sails, he measures his progress and plots his course on the same charts,
- He makes corrections to ensure that his measured course approximates his planned course.
- Given an accurate set of charts and accurate measurement of the ground speed and heading of the yacht, he reaches his destination.

Mobile Robot Navigation



Sensors

Stereo vision

Interaction

Communications

Processing Power

Sonars

Bump sensors

Infrared sensors

Laser scanner

Bump sensors

Sonars

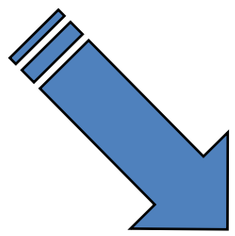
Motors

Odometry

Robot Navigation

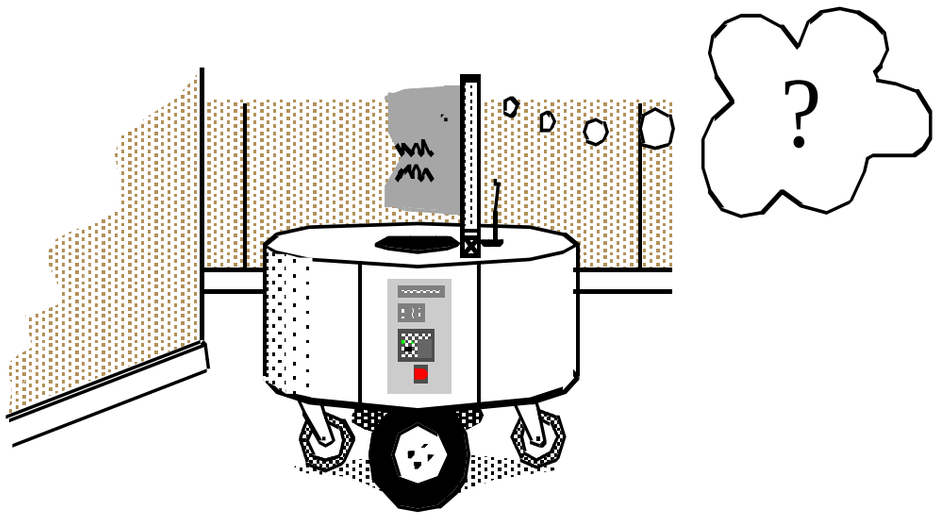
Mobile Robot navigation requires the robot to move and achieve its motion targets and goals in typical environments

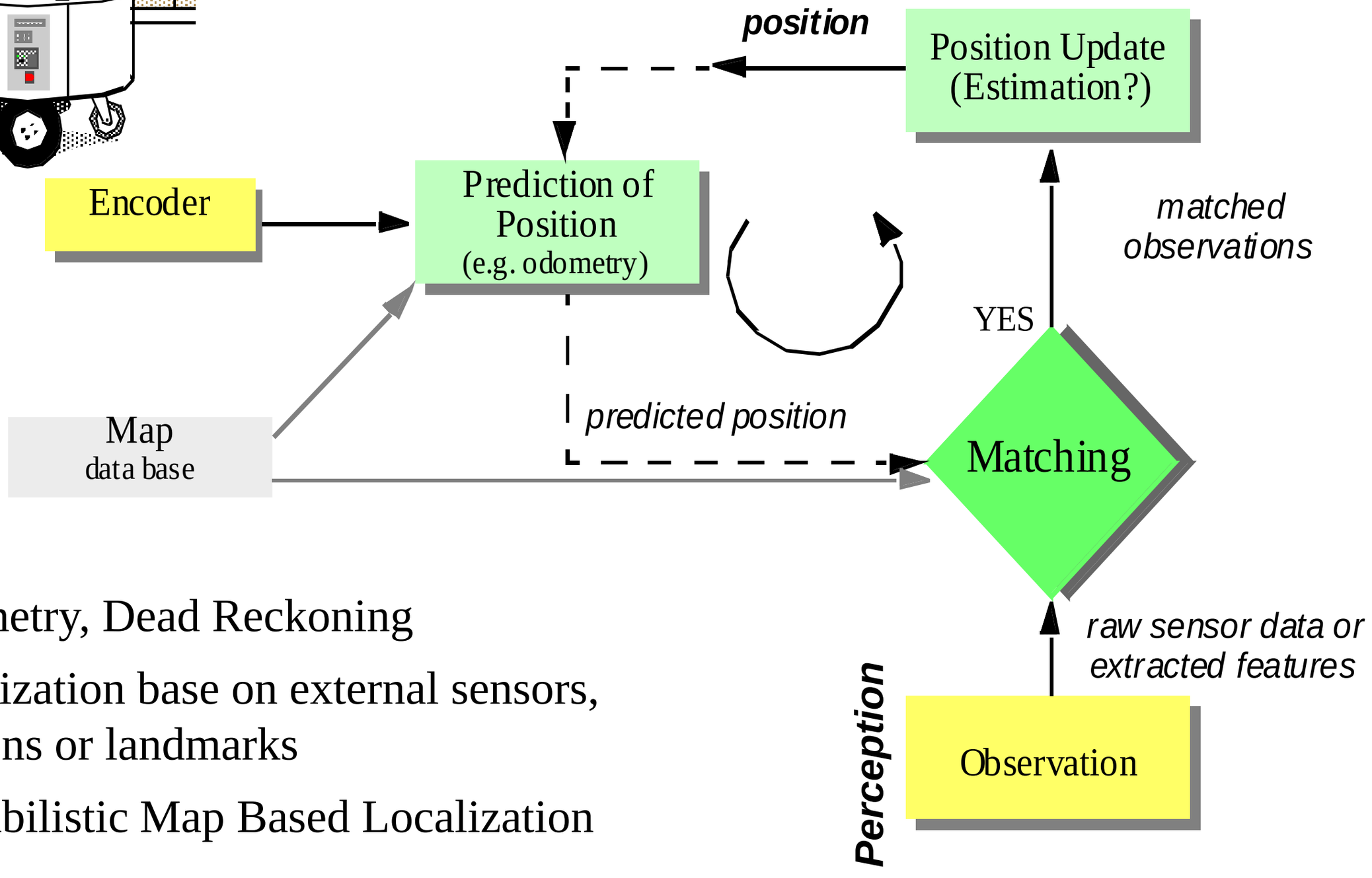
Hence



- Localization (where am I?)
- Mapping (what is my workspace?)
- Path planning (how to get there?)
- Obstacle avoidance (… get there safely…)

Localization, Where am I?





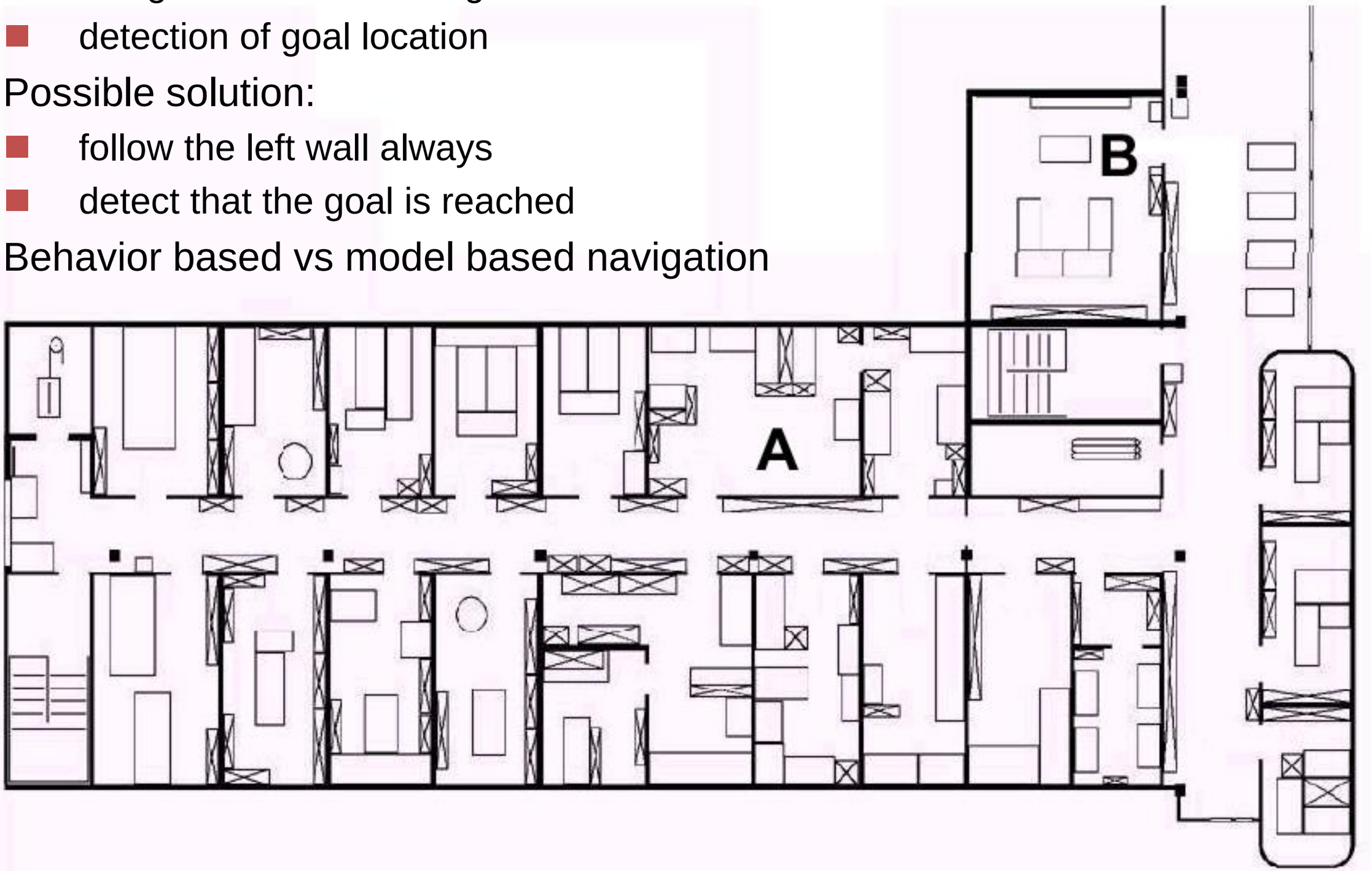
- Odometry, Dead Reckoning
- Localization base on external sensors, beacons or landmarks
- Probabilistic Map Based Localization

Challenges of Localization

- Knowing the absolute position. GPS is not always sufficient:
  - Unavailable in indoor environments
  - Reflection off buildings cause errors
  - Unavailable underwater, in tunnels, caves, etc.
- Planning needs more than just location
- Factors influencing the quality of location estimates
  - Sensor noise
  - Sensor errors
  - Wheel slippage
  - Odometry position estimation (drift)

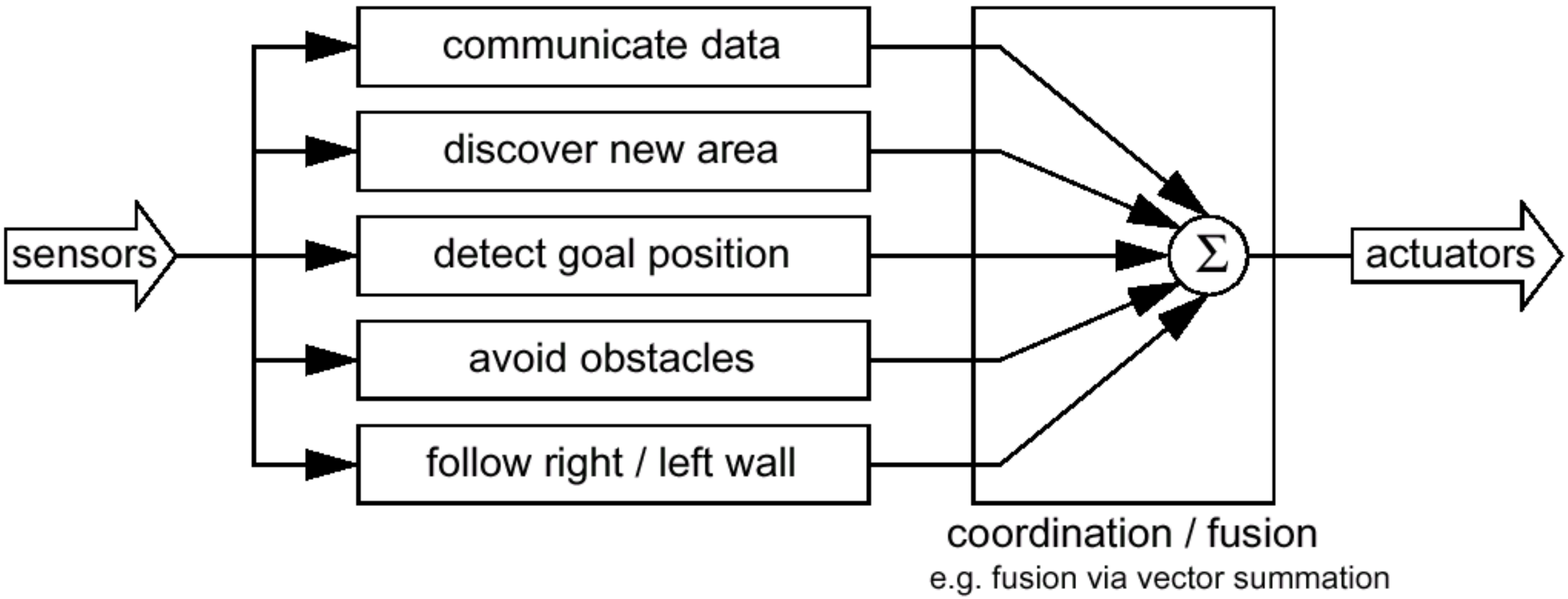
To Localize or Not?

- Constraints on navigation:
  - navigation without hitting obstacles
  - detection of goal location
- Possible solution:
  - follow the left wall always
  - detect that the goal is reached
- Behavior based vs model based navigation



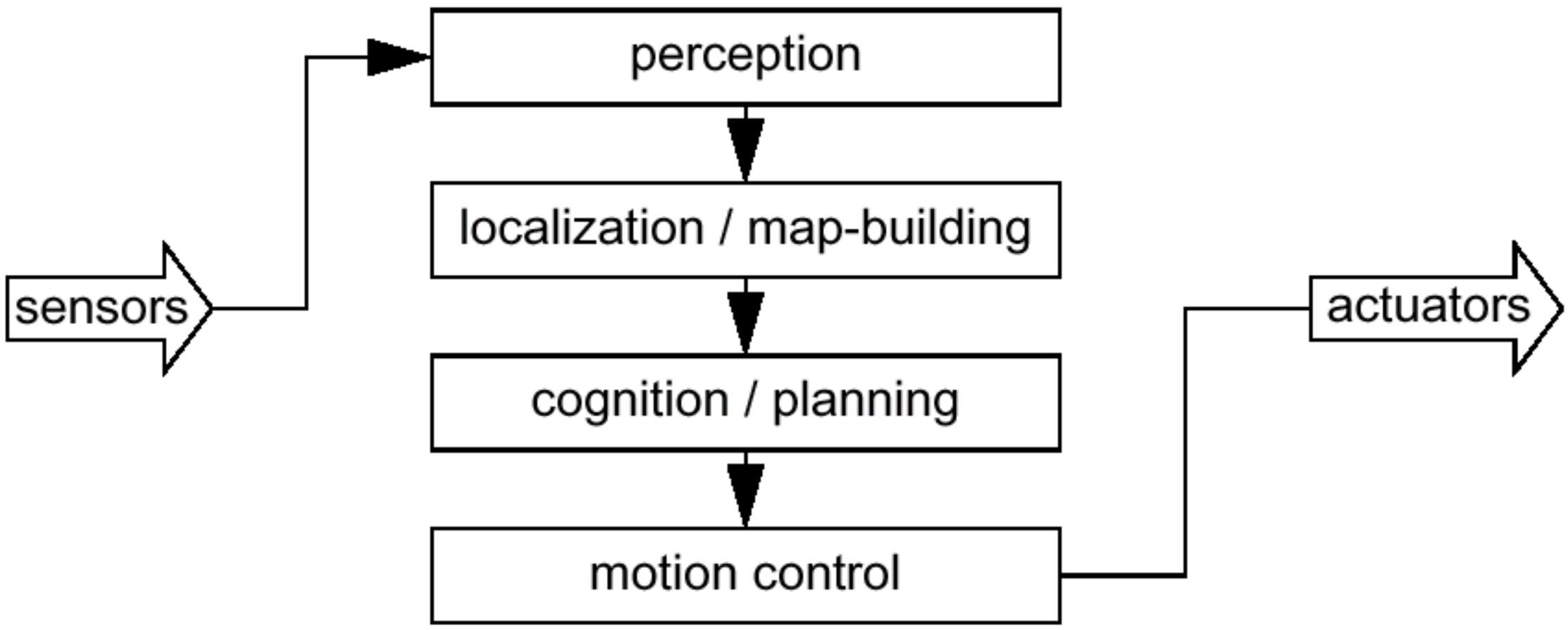
Behavior Based Navigation

- Uses no map or localisation
- Just react to what is perceived



Model Based Navigation

- Or, localize and use a map and landmarks



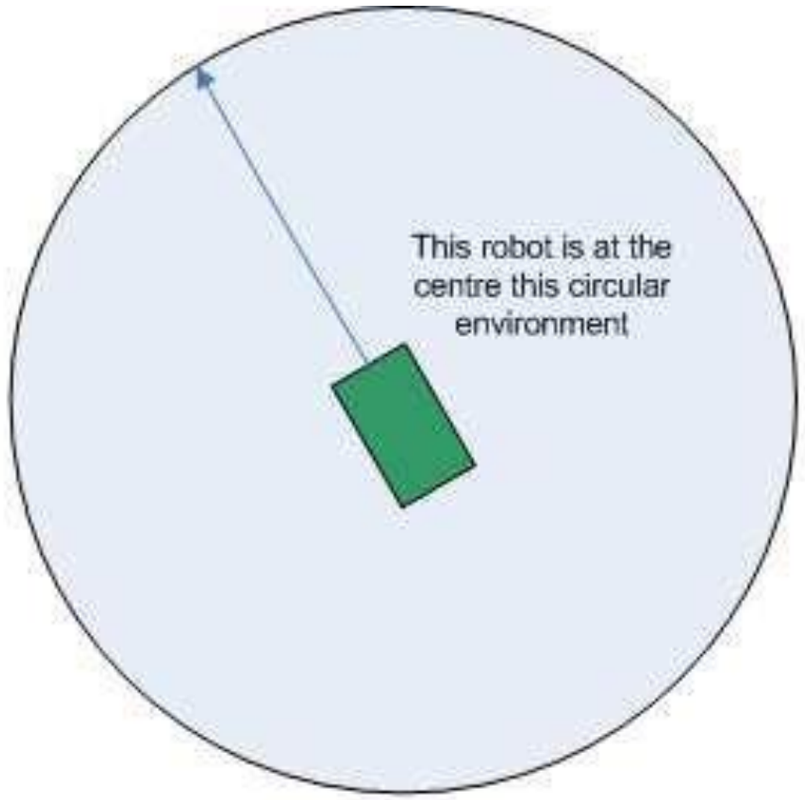
Sensor Noise

- Where does it come from?
  - Environment:
    e.g. surface, illumination …
  - Measurement principle:
    e.g. crosstalk between ultrasonic sensors
- What does it cause?
  - Poor information
- What to do?
  - Sensor fusion and Integration
    - Over multiple sensors
    - Over multiple time steps
    - Incorporate sensor model

Sensor Aliasing

- What is it?
  - Same reading, multiple possible poses (states)
- How generic?
  - Very..
- What is the result?
  - Insufficient information
- What to do?
  - Integrate
    - Over multiple sensors
    - Over time

a) the distance measured to the wall is the same in the two locations shown and they cannot be distinguished



b) The distance measured to the circular boundary is the same in all directions and makes it impossible to determine the orientation of the robot

Odometry (Dead Reckoning)

- What is “odometry and dead reckoning”?
  - Position update is based on **proprioceptive** sensors
    - Odometry: wheel sensors
    - Dead reckoning: heading sensors (e.g. gyros, accelerometers)
- How do we do this?
  - The movement of the robot is sensed with wheel encoders and used to track the robot’s position.
    - Pros: Straight forward, easy
    - Cons: Errors are integrated -> unbound
  - Improvement: add additional heading sensors (e.g. gyroscope)

Errors in Odometry

- Error types
  - Deterministic (systematic. e.g. drift)
  - Non-deterministic (non-systematic, e.g. surface irregularities)
- What to do with them?
  - Calibration can eliminate deterministic errors
  - Non-deterministic errors: leave them!
- Major Error Sources:
  - Limited resolution during integration (time increments, measurement resolution …)
  - Misalignment of the wheels (deterministic)
  - Unequal wheel diameter (deterministic)
  - Variation in the contact point of the wheel
  - Unequal floor contact (slipping, not planar …)
  - …

Classification of Integration Errors

- **Range error**: integrated path length (distance) of the robots movement
  - sum of the wheel movements
- **Turn error**: similar to range error, but for turns
  - difference of the wheel motions
- **Drift error**: difference in the error of the wheels leads to an error in the robots angular orientation

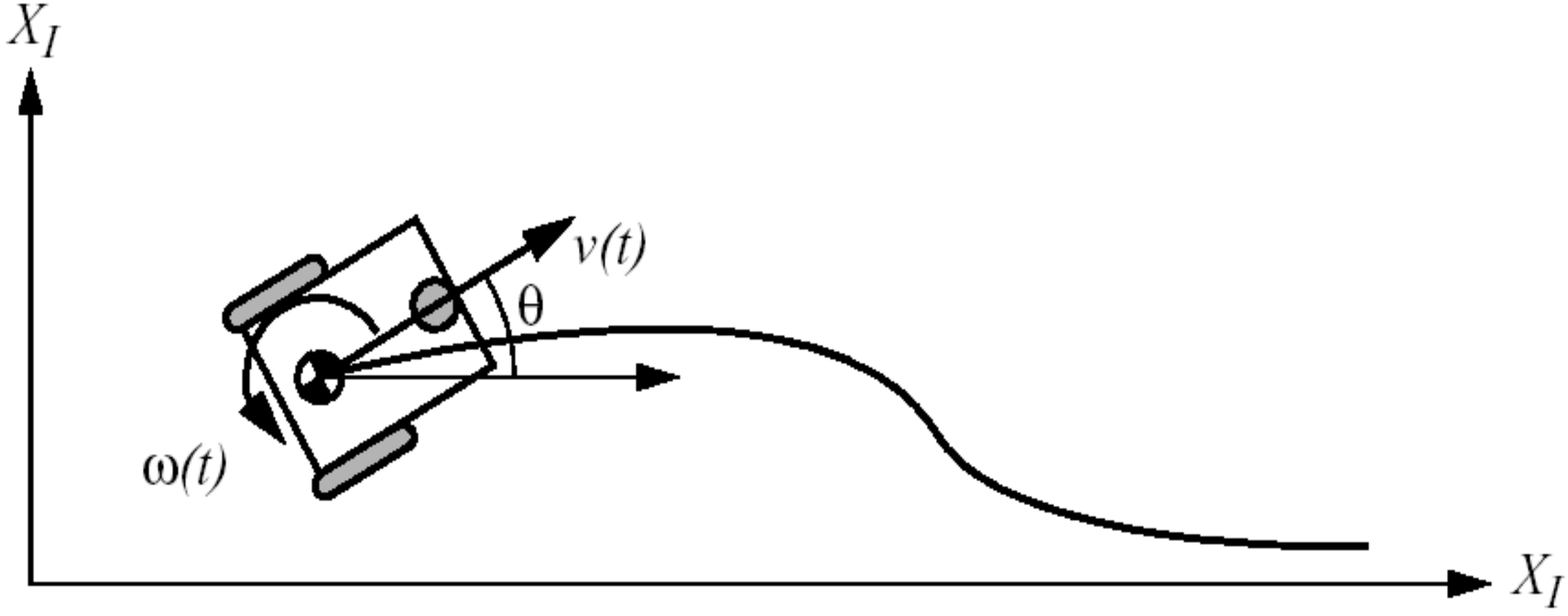
Over long periods of time, turn and drift errors

far outweigh range errors!

  - Consider moving forward on a straight line along the **x** axis. The error in the **y**-position introduced by a move of d meters will have a drift of $d * \sin \Delta\Theta$, which can be quite large as the angular error $\Delta\Theta$ grows.

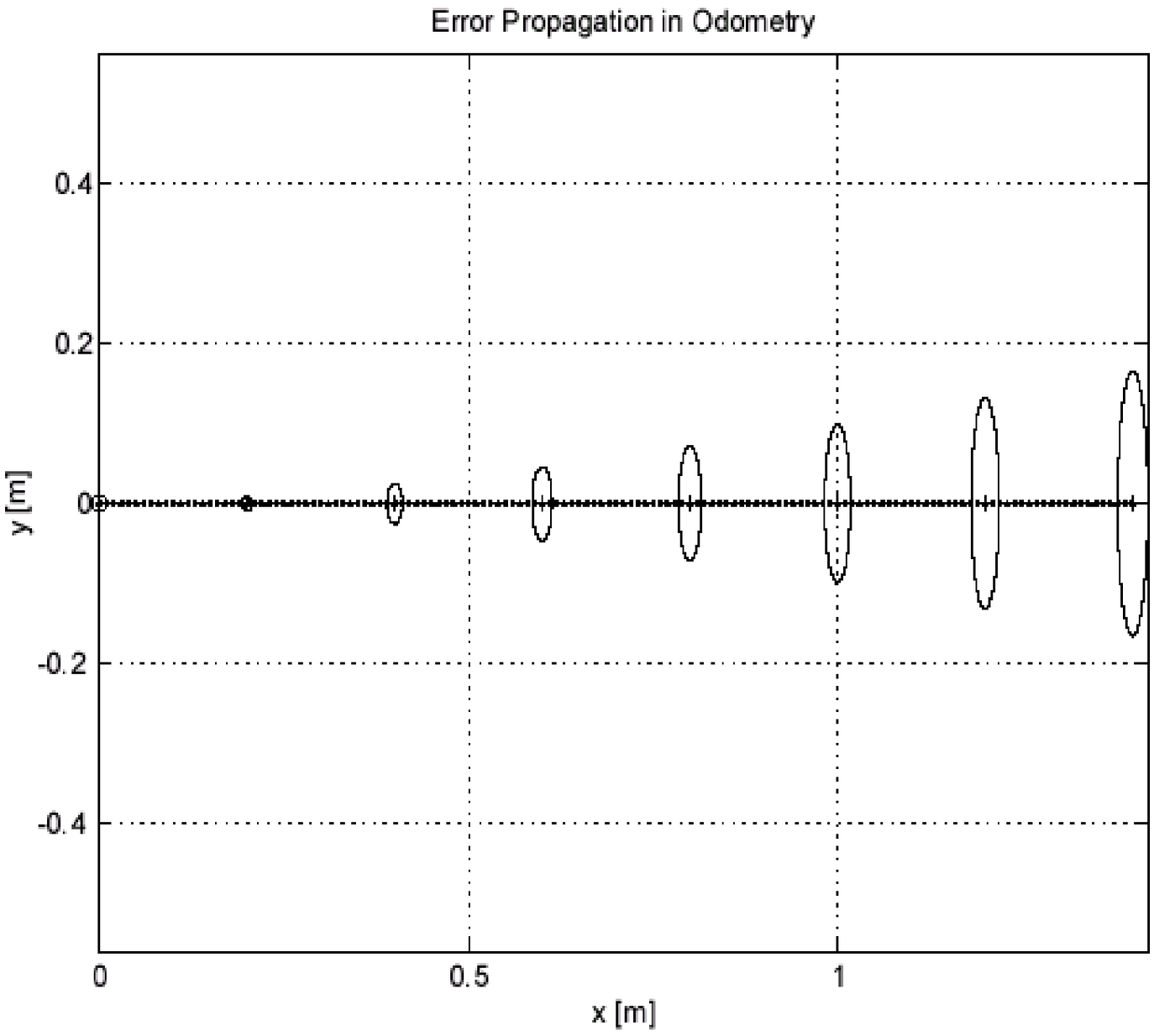
The Differential Drive Robot

$$p = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \qquad p' = p + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix}$$



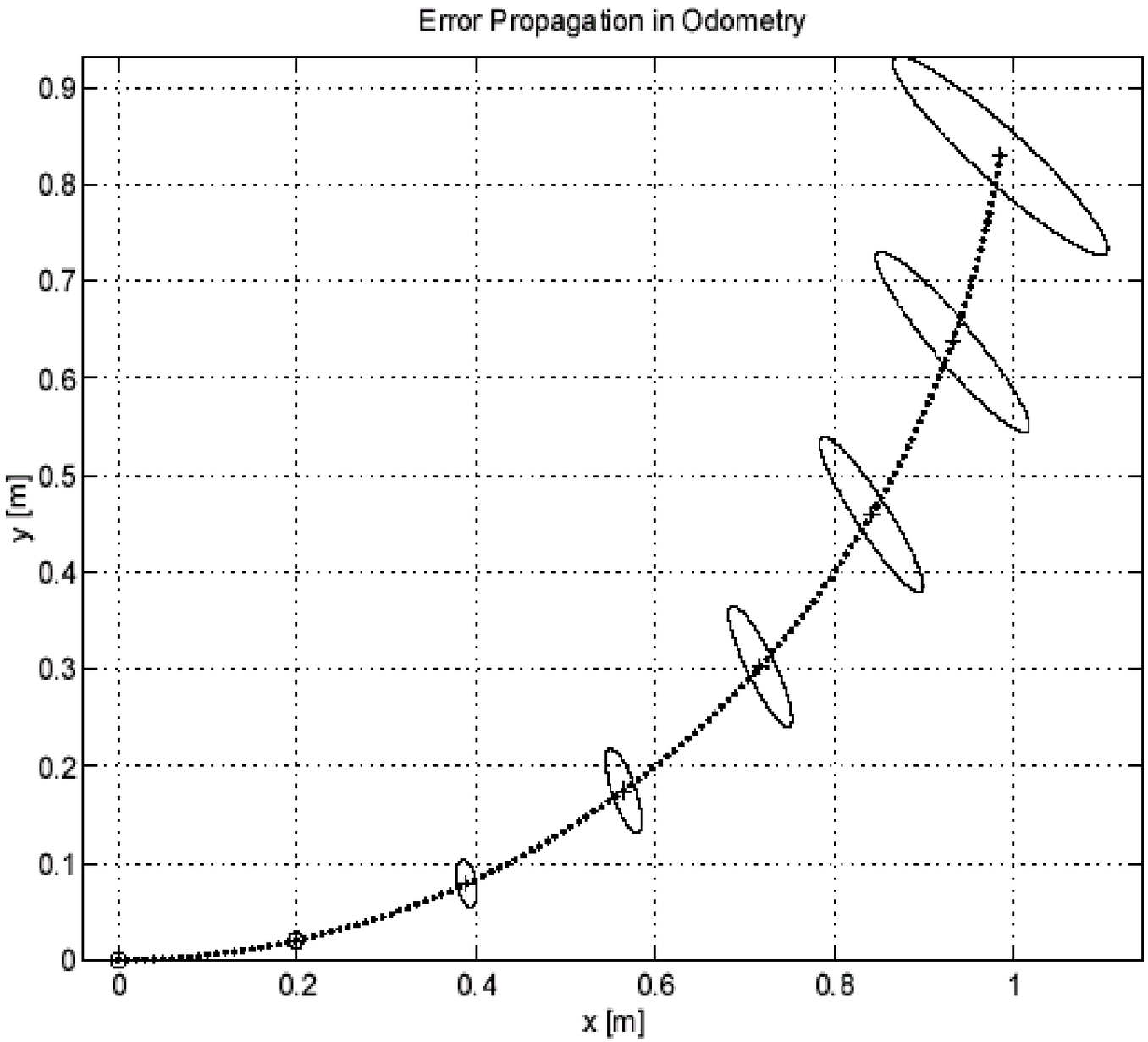
Growth of Pose Uncertainty for Straight Line Movement

- Note: Errors perpendicular to the direction of movement are growing much faster!



Growth of Pose Uncertainty for Movement on a Circle

- Note: Error ellipse does not remain perpendicular to the direction of movement!



Map Representation

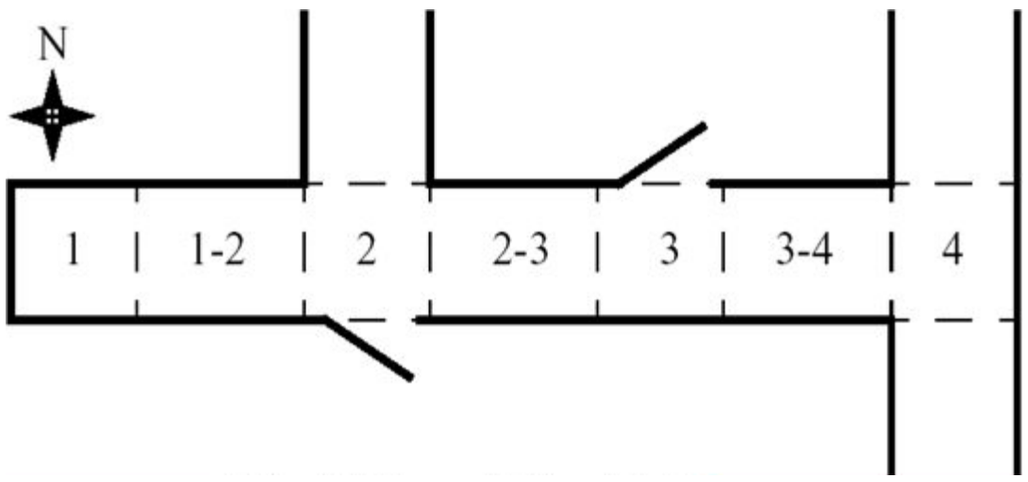
- Issues:
  1. Map precision vs. application
  2. Feature precision vs. map precision
  3. Precision vs. computational complexity
- Continuous Representation
- Decomposition (Discretization)

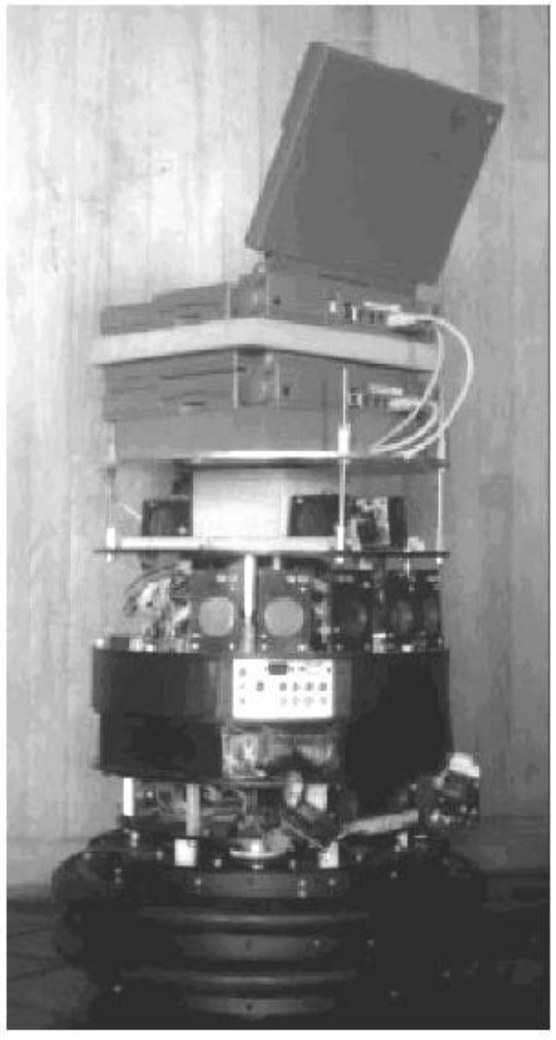
Representation of the Environment

- Environment Representation
  - Continuous Metric $\rightarrow$ x,y,θ
  - Discrete Metric $\rightarrow$ metric grid
  - Discrete Topological $\rightarrow$ topological grid
- Environment Modeling of Features
  - Raw sensor data, e.g. laser range data, grayscale images
    - large volume of data, low distinctiveness on the level of individual values
    - makes use of all acquired information
  - Low level features, e.g. outline of geometric features
    - medium volume of data, average distinctiveness
    - filters out the useful information, still ambiguities
  - High level features, e.g. doors, a car, the Eiffel tower
    - low volume of data, high distinctiveness
    - filters out the useful information, few/no ambiguities, not enough information

Map Representation Examples

- The Dervish Robot
- Designed for
  - Office navigation
  - Office deliveries
- Topological Localization with Sonar
  - Assumes that the environment is topologically structured.
  - Quantizates environment into discrete states:





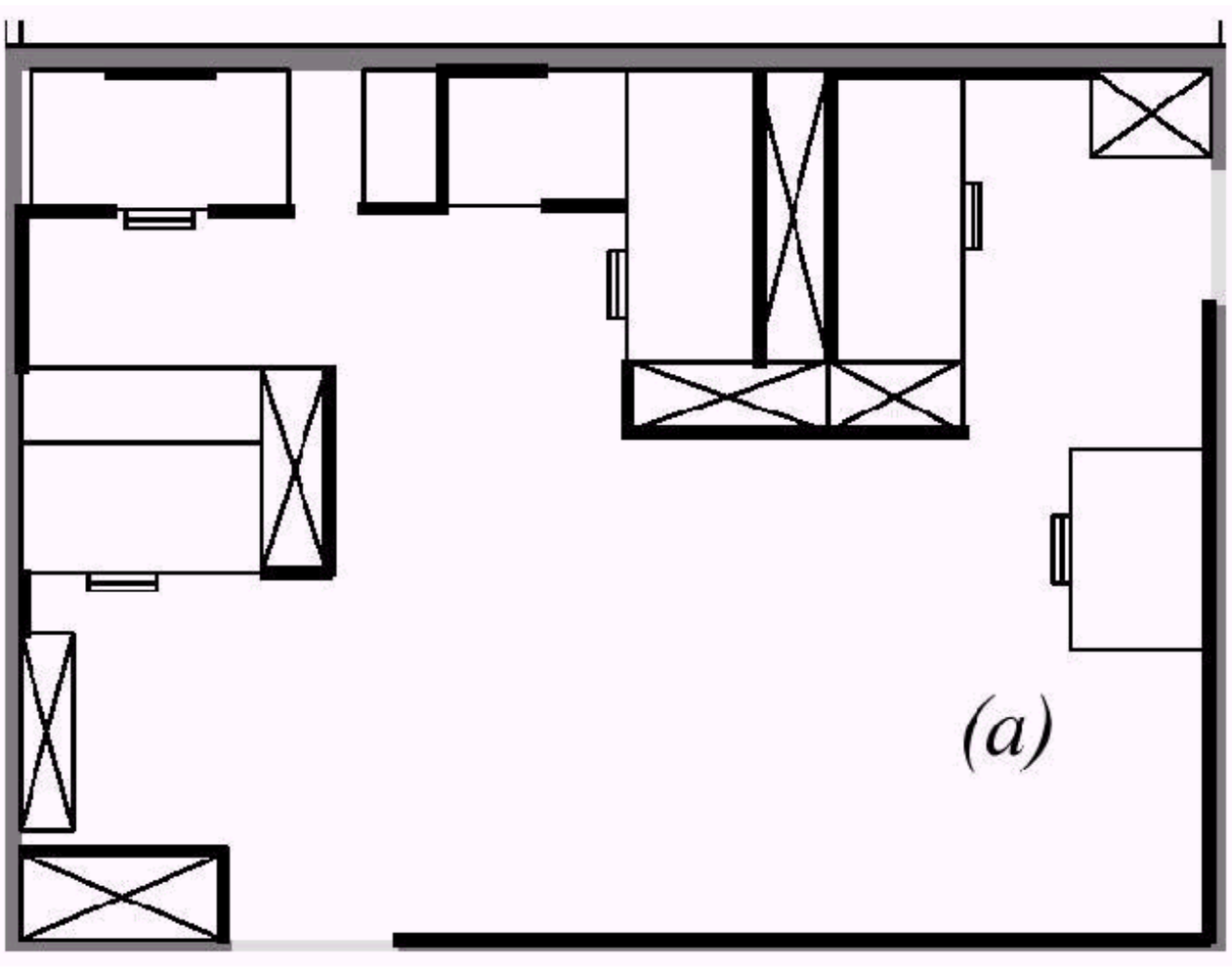
Source: https://pdfs.semanticscholar.org/d6b4/fcf9dc5f365ef2dda56f69424058bb287754.pdf

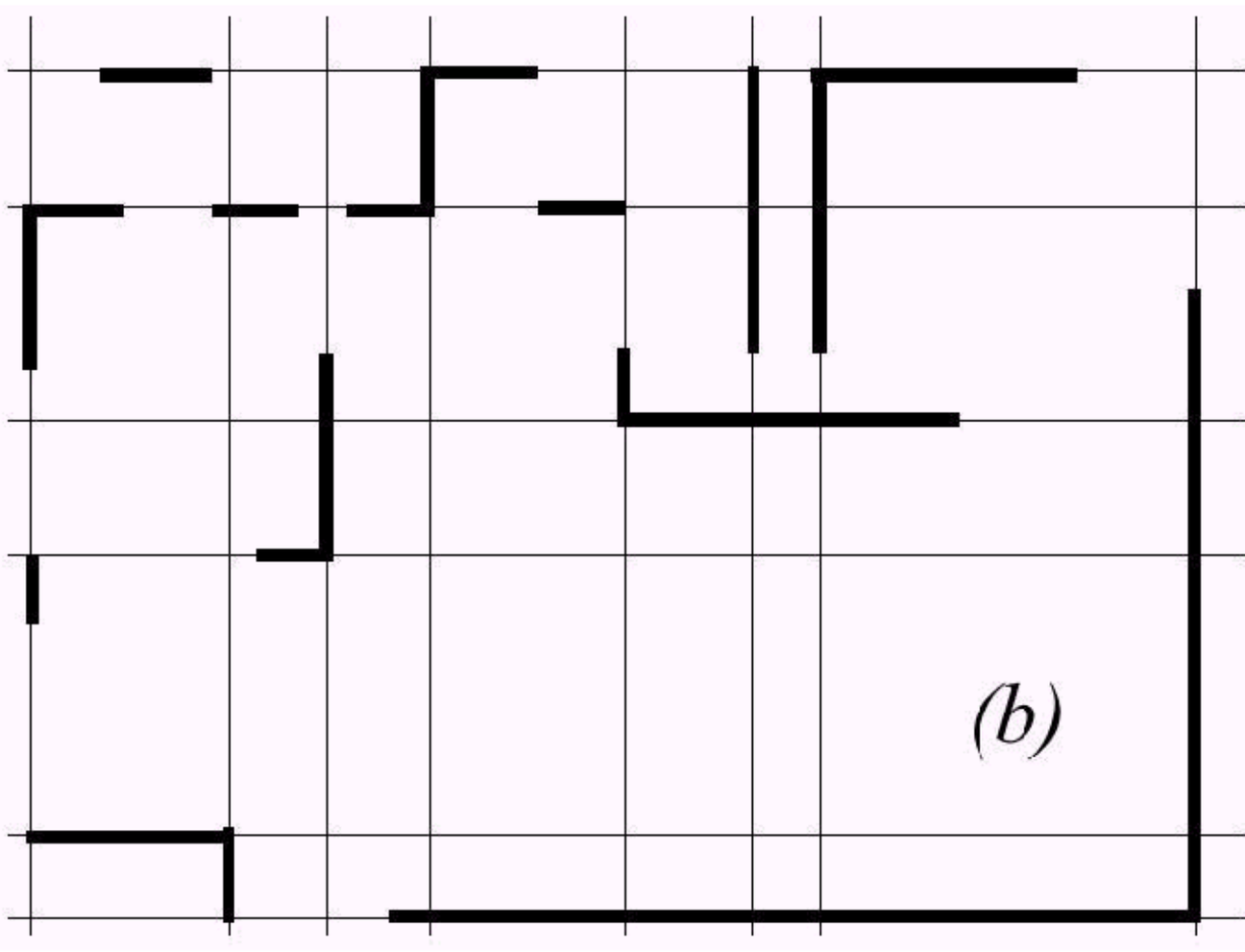
Continuous Line-Based Representation

a) Architecture map

b) Representation with set of lines of vectors

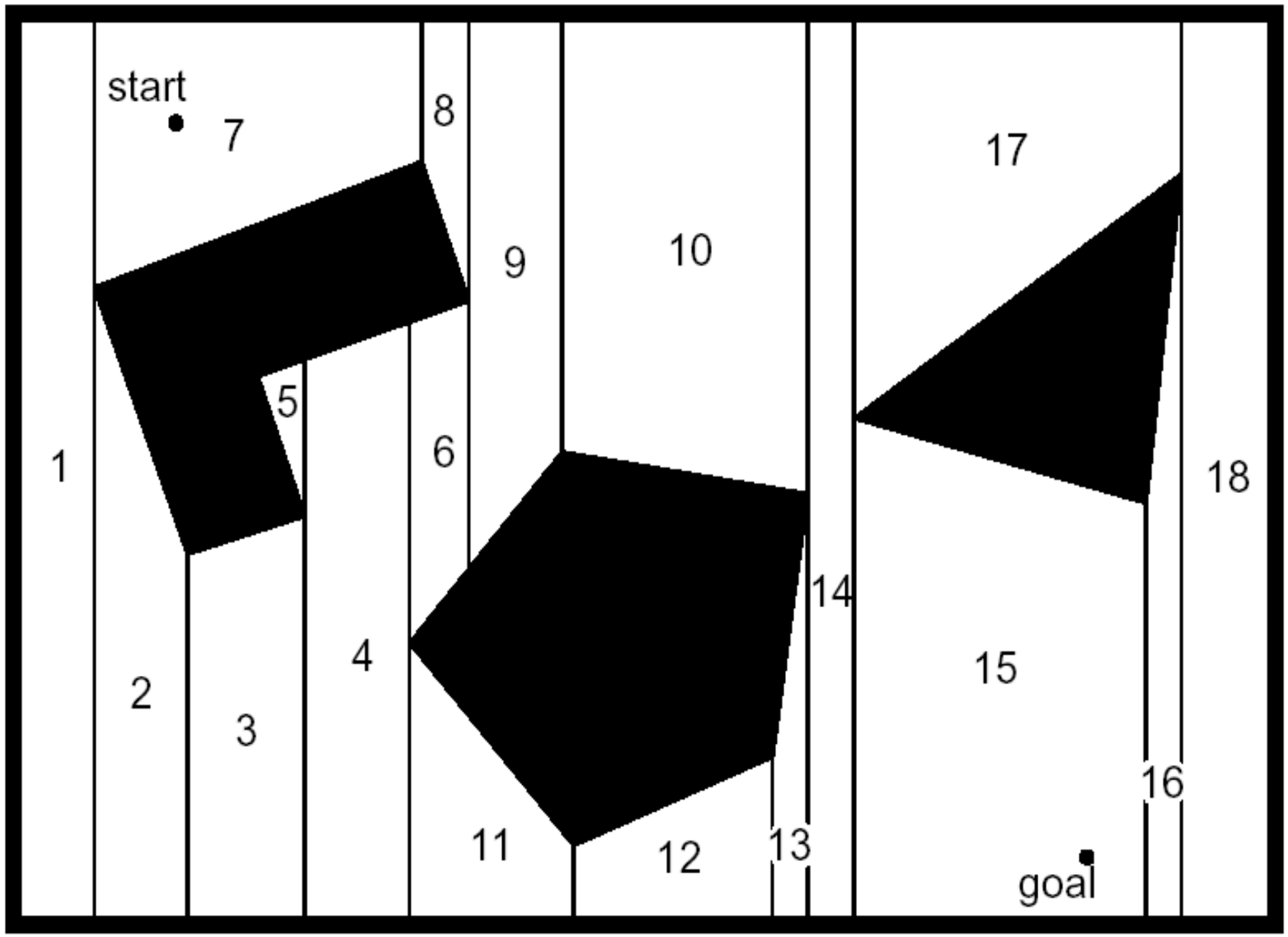
– Assumes environment is geometric





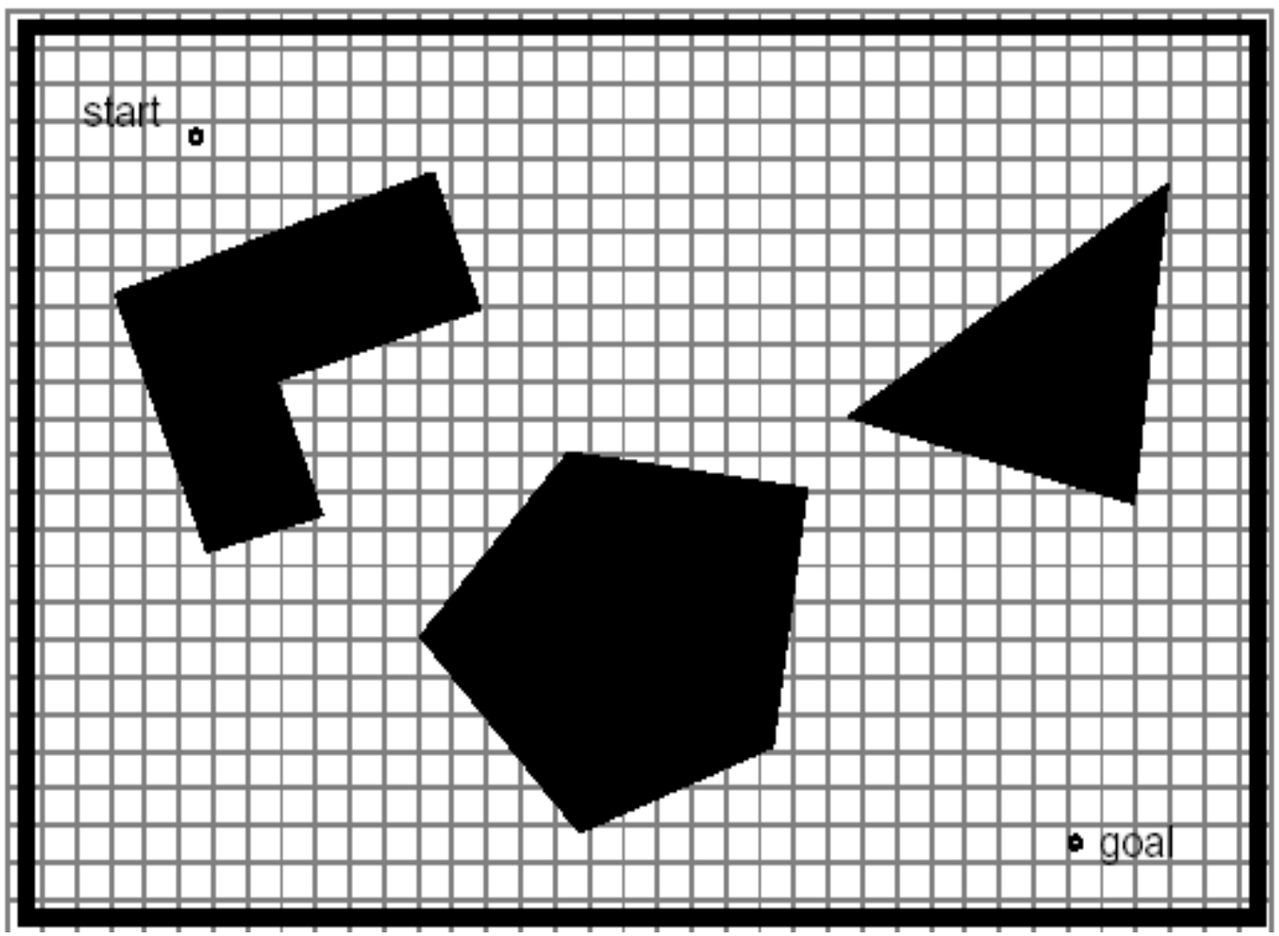
Exact Cell Representation

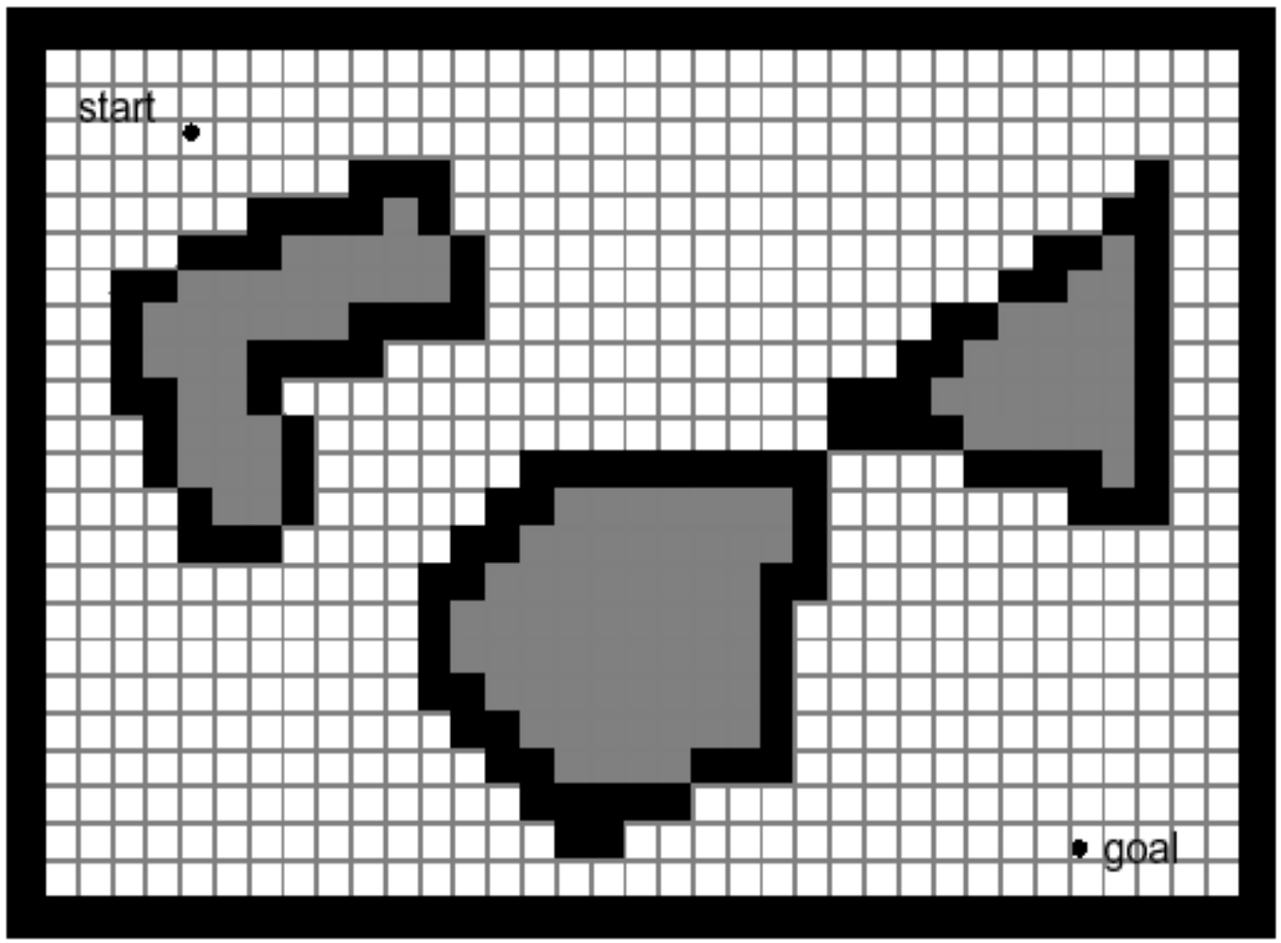
- Graphical representation of objects
- Segments map into cells that are based on objects



Fixed Cell Representation

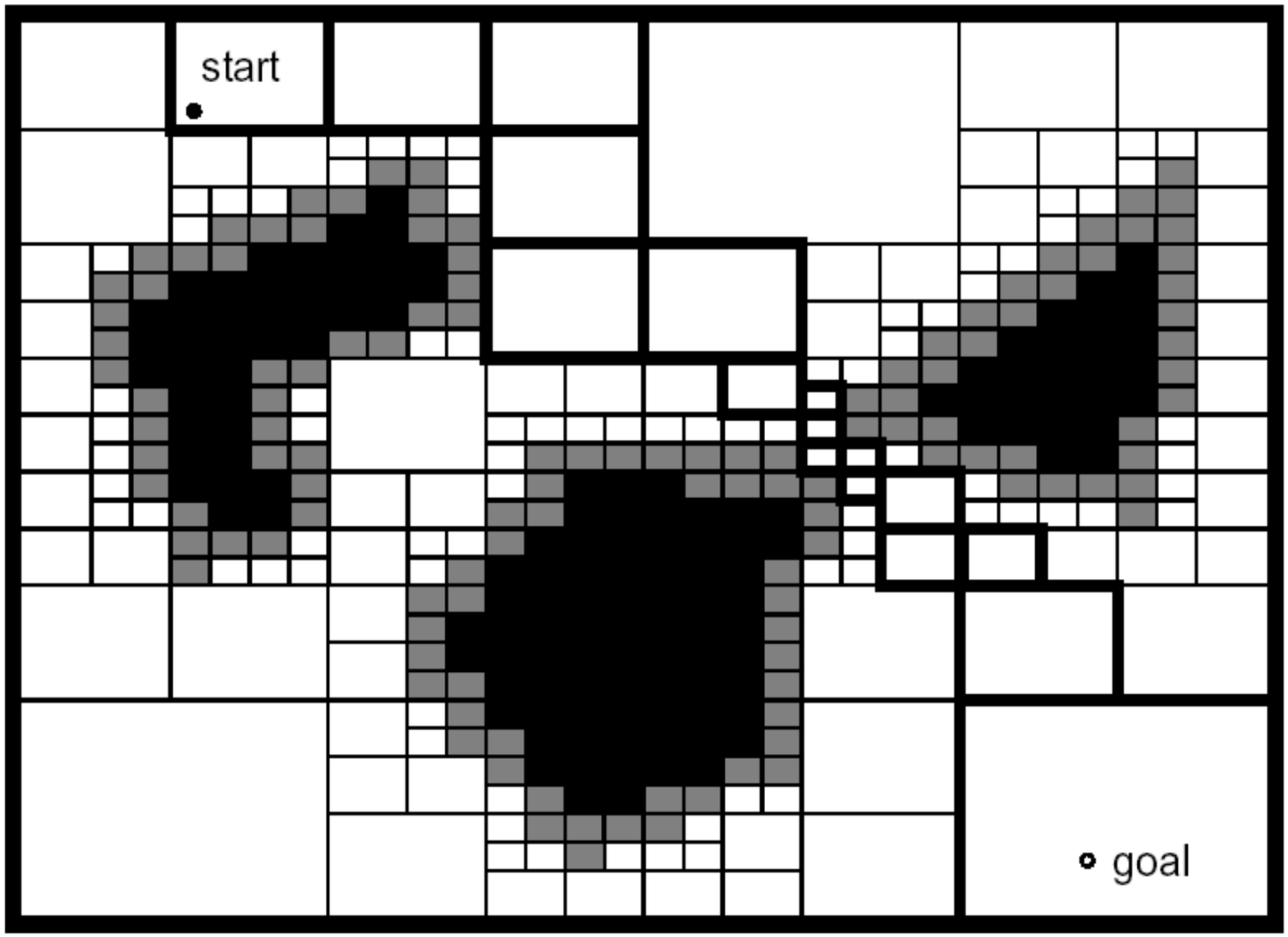
- Objects are mapped to cells in a regular grid
  - i.e. Occupancy grid





Adaptive Cell Representation

- Objects are represented with variable sized cells and values.

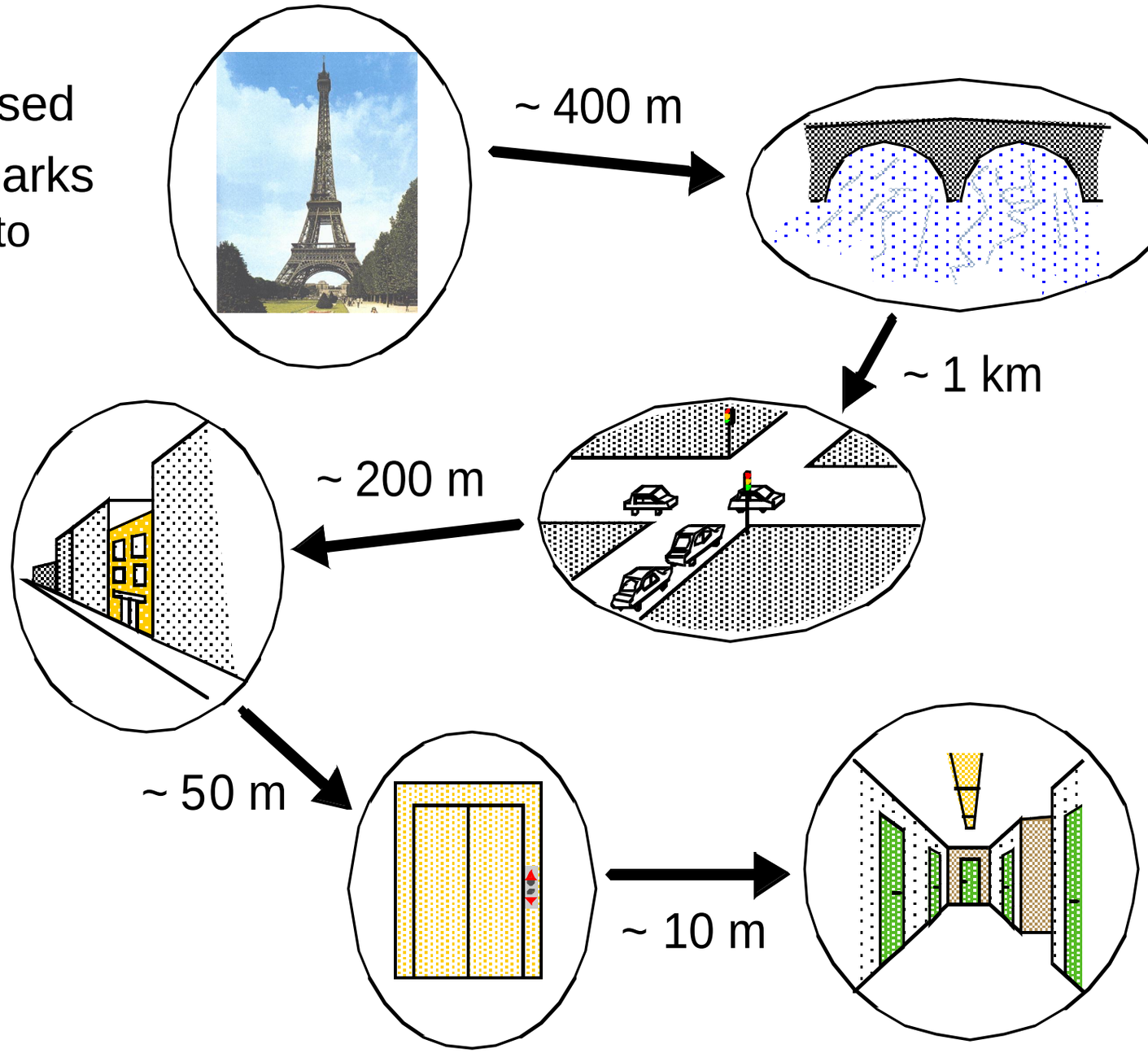


Map Representation Challenges

- Finely granulated *fixed decomposition* grids result in a huge table
  - Huge processing power needed.
  - Large memory requirements.
- Reducing complexity:
  Randomized Sampling / Particle Filter
  - Approximated belief state by representing only a ‘representative’ subset of all states (possible locations).
  - E.g update only 10% of all possible locations.
  - The sampling process is typically weighted, e.g. put more samples around the local peaks in the probability density function.
  - However, you have to ensure some less likely locations are still tracked, otherwise the robot might get lost.

Topological Decomposition

- Landmark perception and connectivity.
- Landmarks can be decomposed
- Allows organization of landmarks
  - (i.e. from global landmarks to local landmarks).
  - Organized by context of appearance.

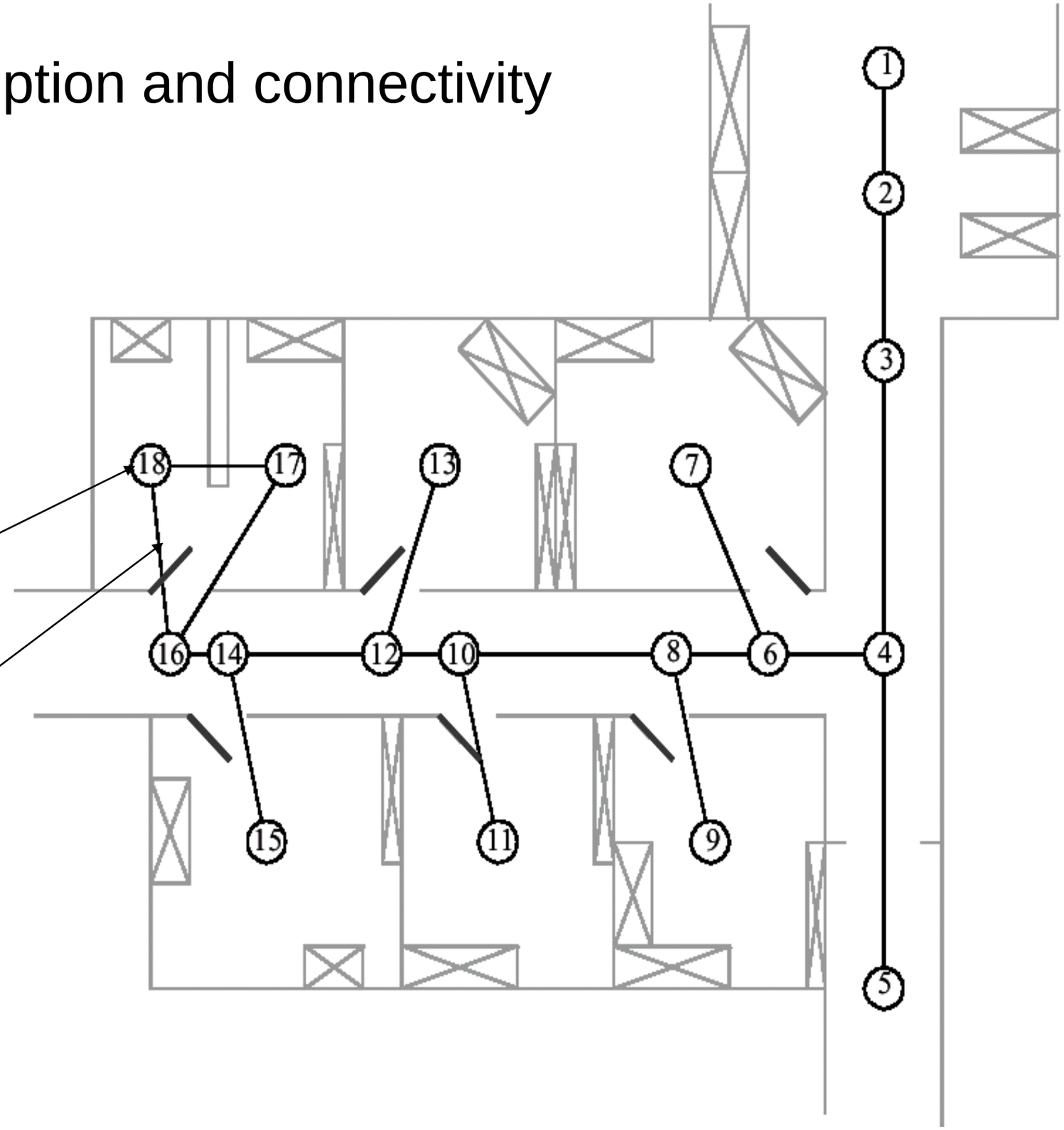


Topological Decomposition

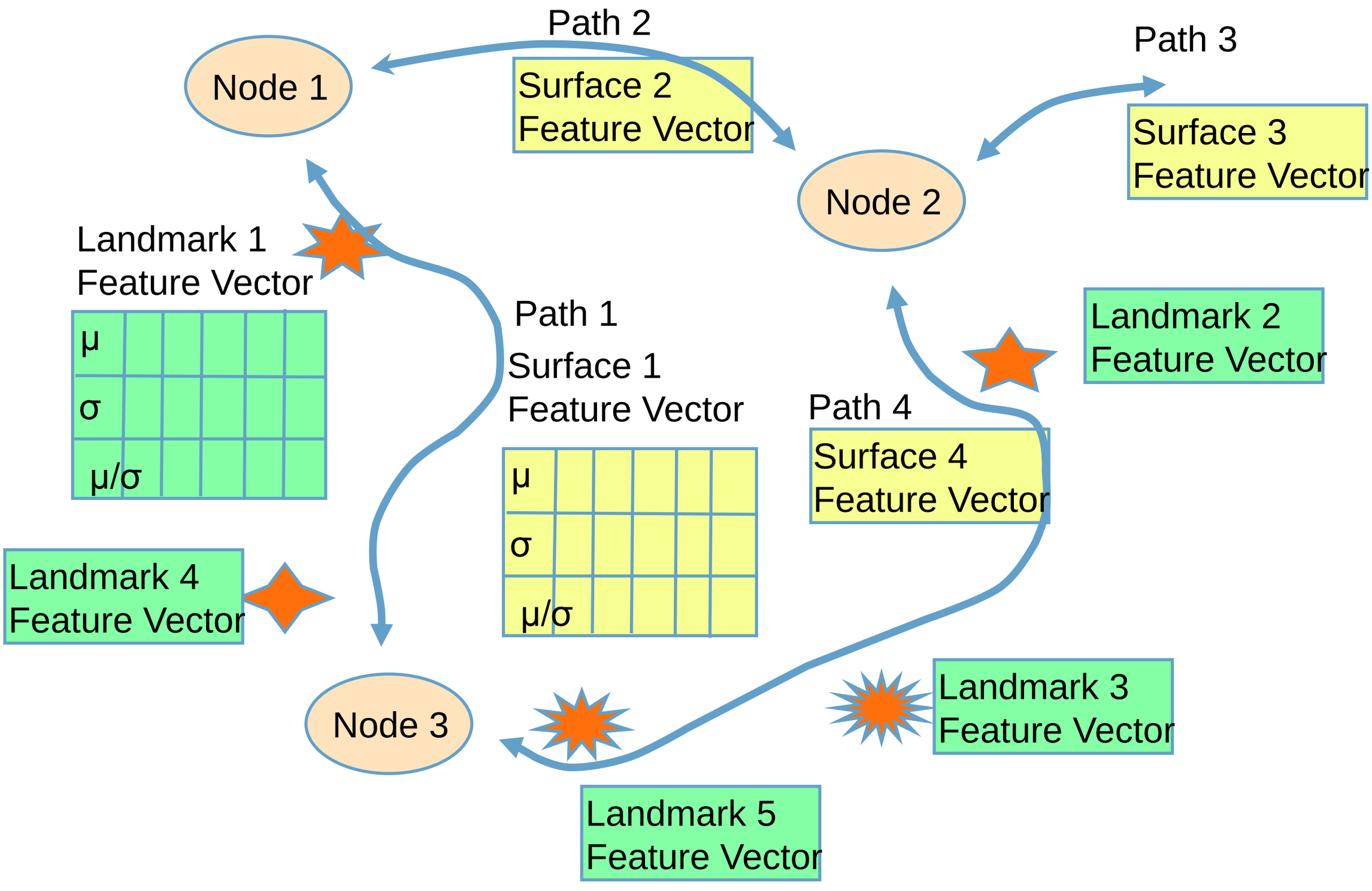
- Landmark perception and connectivity

node

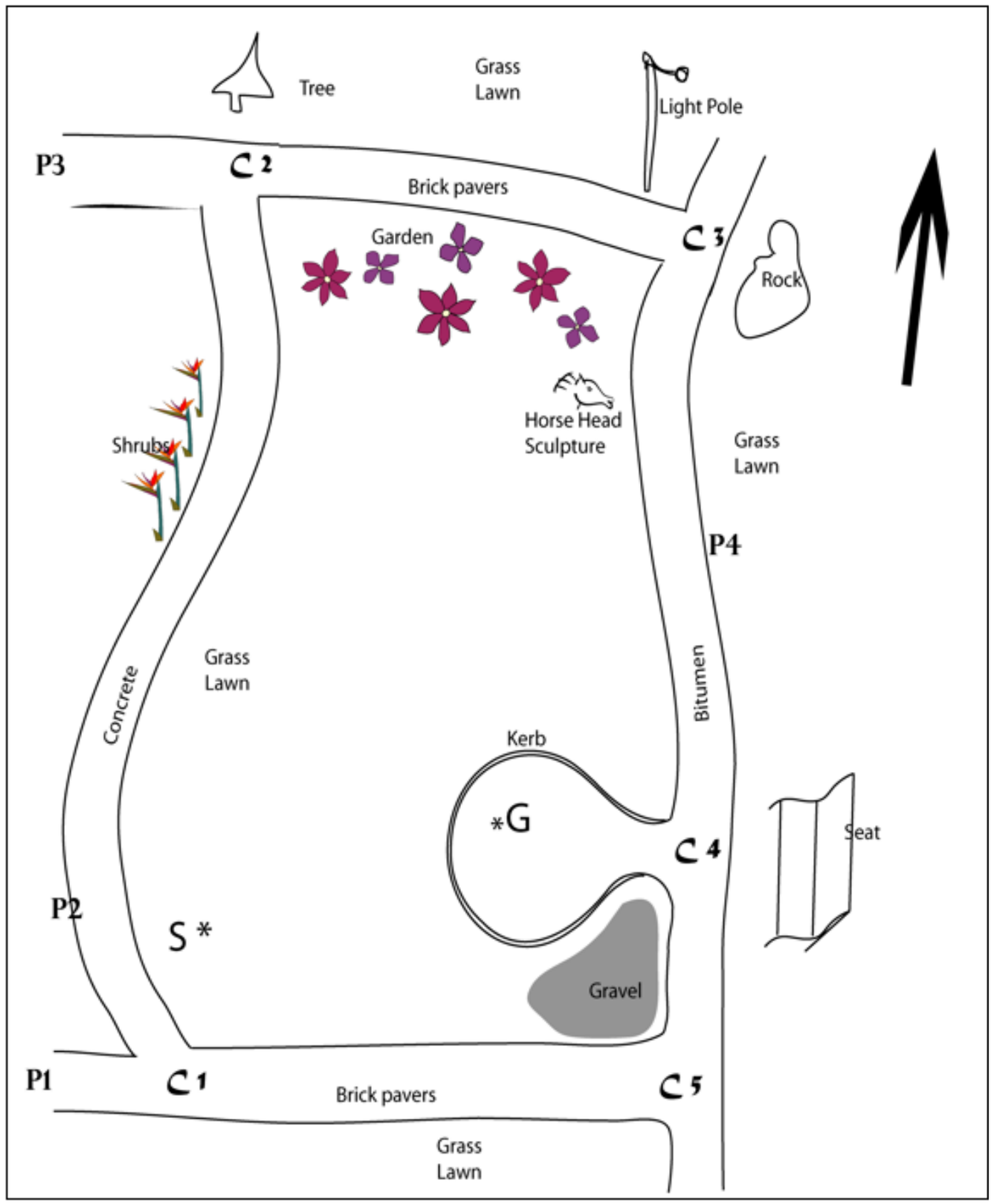
Connectivity (arc)



Landmark map



Fuzzy map



- Based on descriptive maps.
- Describes features by their properties.

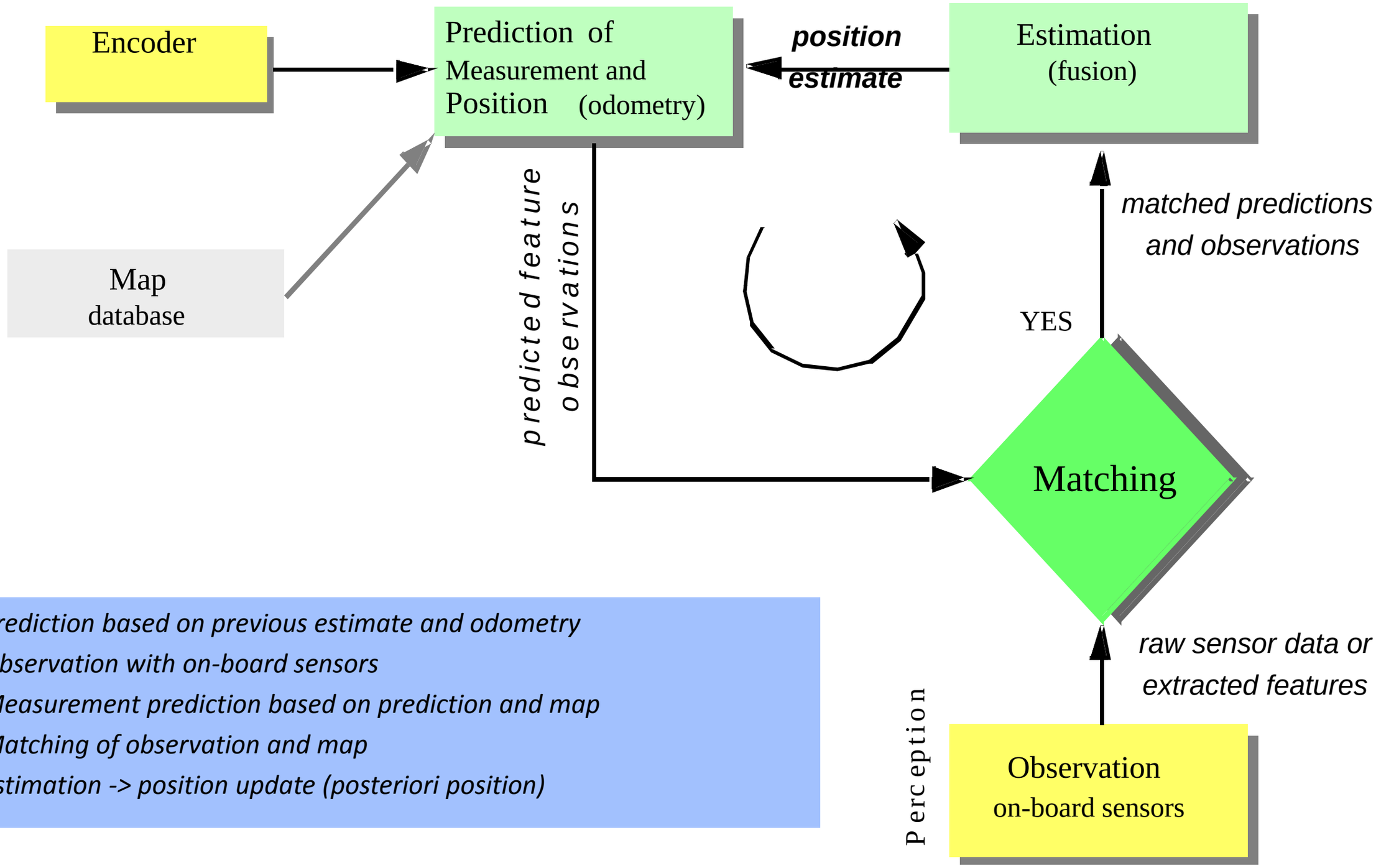
Planning

Goal ⇒ G - plan path to G from current location
Current Location ⇒ I think I am on P3
History ⇒ I turned right onto P3 at Landmark = Tree
Scan ⇒ Expect to find
Surface = brick pavers
Landmark to right = garden
Landmark on left = Green Grass
Path ⇒ Choose shortest
Travel East along P3 to C1
Turn right onto p4
Travel South to C4
Turn right into Bay
Head west to G

Localisation

IF surface = paving bricks THEN
IF heading ≈ west THEN
IF LastLandmark = Light Pole THEN
IF distance from Last Landmark ≈ 50 THEN
IF Garden to south THEN
On path P3 , Heading toward C2

The Five Steps for Map-Based Localization



1. *Prediction based on previous estimate and odometry*
2. *Observation with on-board sensors*
3. *Measurement prediction based on prediction and map*
4. *Matching of observation and map*
5. *Estimation -> position update (posteriori position)*

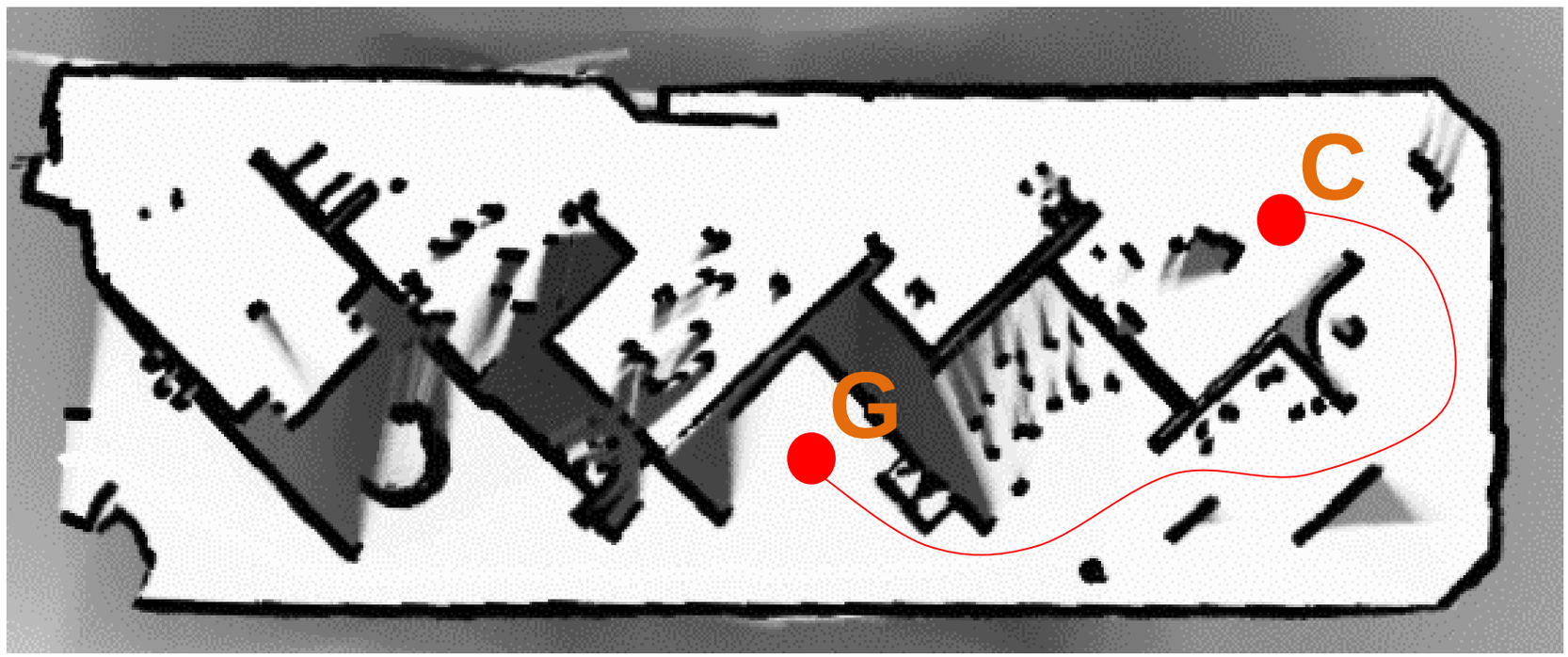
Challenges of Planning

- Real world is dynamic
- Perception is still a major challenge
  - Error prone
  - Extraction of useful information difficult
- Traversal of open space
- How to build up topology (boundaries of nodes)
- Sensor fusion
- …

Robot Path Planning

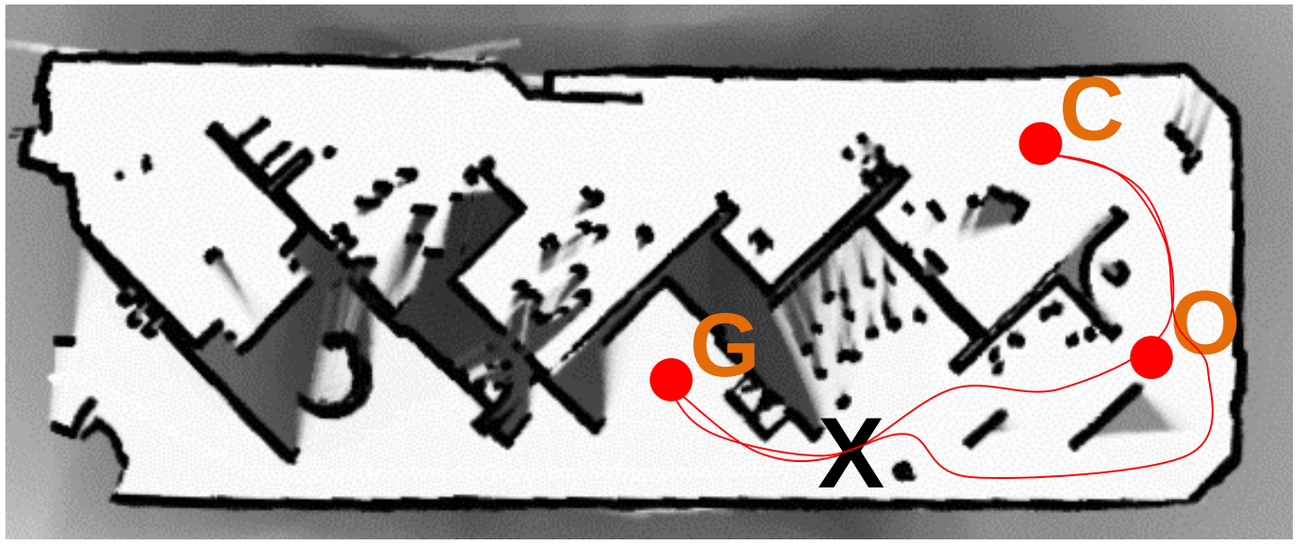
- What is path planning?
  - Given an environment representation - Map

- Knowledge of current position C
- Target position G
- A path has to be planned and tracked that will take the robot from C to G



Robot Path Planning

- Example of challenges with path planning:
  - During execution (run-time)
  - Objects O blocks the robot



- The planned path is no longer valid
- The obstacle needs to be avoided and the path needs to be re-planned

Robot Path Planning

- Robot path planning involves giving a robot the capability of planning its own motions. Deciding automatically what motions to execute in order to achieve a task specified by initial and goal state, and spatial arrangements of physical objects.

- Agre and Chapman (1990) claim that the way we plan depends on how we are going to use those plans.
- They contrast two views of plans:
    - **Plan as a program view**
        - plan use is the execution of an effective procedure
    - **Plan as a communication view**
        - plan use is like following natural language instructions

Plan-as-program view

- The plan-as-program view:
  - Understands navigation as a matter of problem solving and control.
  - The world presents the robot with a series of formally defined problems that require a solution.
  - The planner produces a solution to each problem which is executed by the robot.
  - This approach is similar to how a European navigator estimates his position on a map and plots a path to his destination. However, when one points to a position on the map, the navigator is often unable to relate this to the physical environment through which he is sailing.

- Sensor inputs are only used to monitor progress relative to the plan, not to modify the plan.
- The execution of the plan is intended to be domain independent, making the construction of the map very domain dependent if it is to be effective.
- The computational complexity of the plan and map construction can grow exponentially with all the details that have to be included to deal with all possible situations.

Plan-as-communication view

- In the plan-as-communication view:
  - The plan plays only an indirect role in planning.
  - Plan use is like following natural language instructions.
  - The plan does not directly determine the user's course of action, but rather advises him on how to achieve his goal.
  - The plan is one of a number of sources of information for deciding what to do.

- Using the plan, requires figuring out how to relate it the task and, in the case of navigation, the ability to perceive the environment. Domain specific knowledge is required.
- The world of everyday life is not a series of problems to be solved, but rather a series of contingencies to handle and opportunities to take.
- An example of this is shown by how the natives of Puluwat Atoll can sail 1,000s of kms to other islands based on passed down stories.

Robot Path Planning

- Many planners implemented use a planning language with which to express the plan and an interpreter to execute the plan.

- The planning language is like a programming language with a set of parametrized primitives and composition operators.

- For example, indoor service robots are often programmed with a sequence of corridors and doorways derived from a map.

Robot Path Planning

- Alternatively, if the plan is to be executed by a mobile robot, the plan may be expressed geometrically.

- The plan is described in precise geometric coordinates, headings and velocities.

- Such a plan can be viewed on a graphics display, executed by a simulated robot or used to control a real robot.

Path Planning Algorithms

- In a less structured, complex, or partly unknown environment the number of possible paths can be extremely large.

- In these circumstances, it may be more efficient to store a map of the world and plan paths through the world using a path planning algorithm rather than to search a large number of known paths stored in memory.

- If the environment is only partially known or ever changing, there may be very few paths that exist over a long period of time, and an algorithmic path planner has to be used.

Path Planning Algorithms

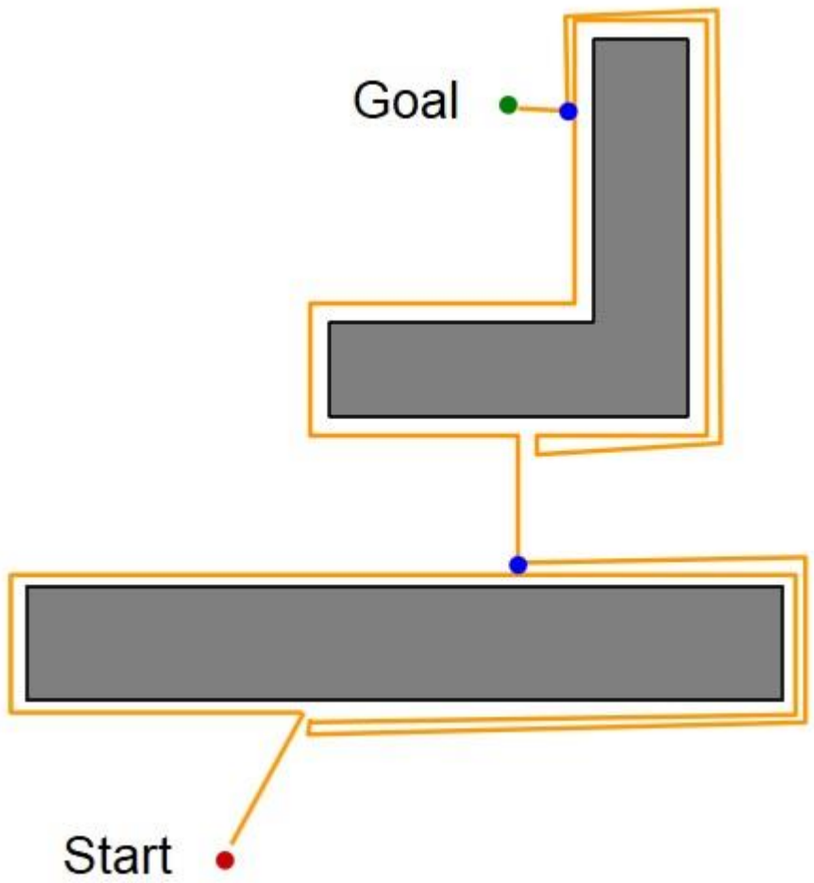
- Most path planners abstract the search space to a graph of possible paths.
- This graph is then searched to find the shortest path.
- This approach arises naturally in a learning environment, where a robot may have traversed several paths to map the world.

Bug Algorithms

Bug algorithms assume only local knowledge of the environment and a global goal

- Bug behaviors are simple:
  - Bug 1: Follow a wall (right or left)
  - Bug 2: Move in a straight line toward goal
- Such Bugs assume essentially tactile sensing
- Tangent Bug deals with finite distance sensing

Bug 1 Algorithm



1) head toward goal

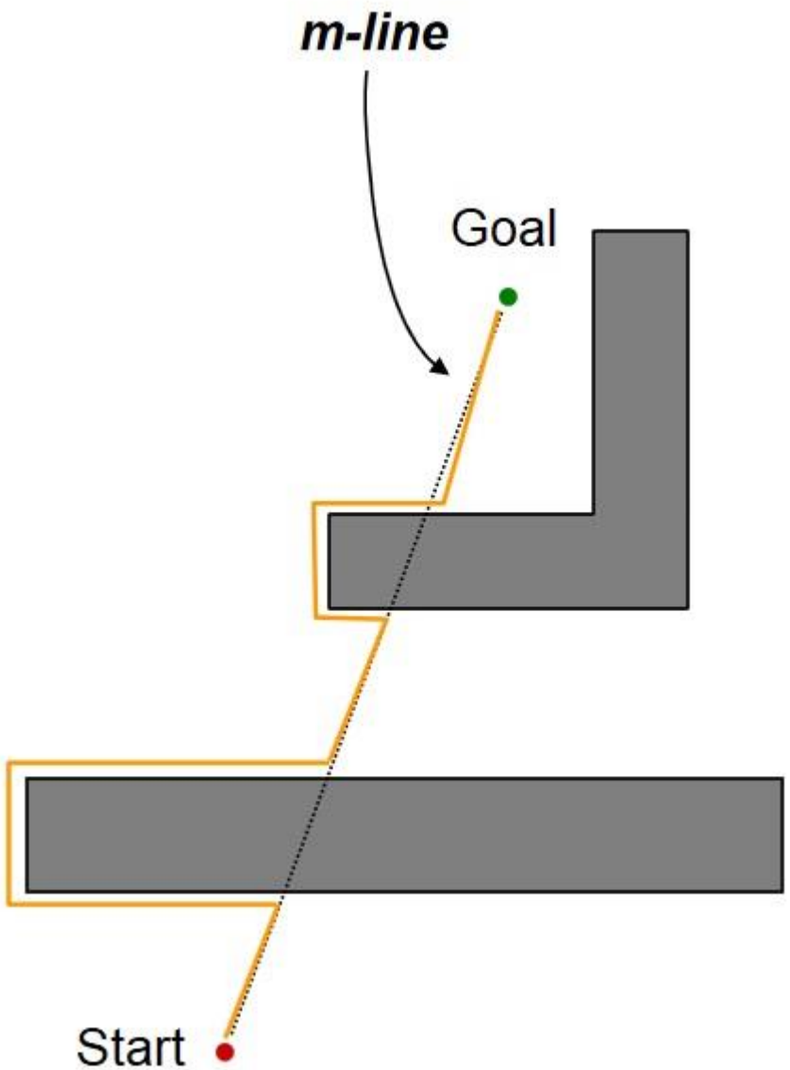
2) if an obstacle is encountered, circumnavigate it and remember how close you get to the goal.

3) return to that closest point (by wall-following) and continue.

Bug 1 Algorithm

- Let $q^L_0 = q_{start}$; i = 1
- repeat
  - repeat
    - from q^L_{i-1} move toward q_{goal}
  - until goal is reached or obstacle encountered at q^H_i
  - if goal is reached, exit
  - repeat
    - follow boundary recording pt q^L_i with shortest distance to goal
  - until q_{goal} is reached or q^H_i is re-encountered
  - if goal is reached, exit
  - Go to q^L_i
  - if move toward q_{goal} moves into obstacle
    - exit with failure
  - else
    - i=i+1
    - continue

Bug 2 Algorithm



1) head toward the goal on the *m-line.*

2) if an obstacle is in the way, follow it until you get to the *m-line* again.

Note: the m-line is the line between the start and the goal.

Bug 2 Algorithm

- Let $q^L_0 = q_{start;}$ i = 1
- repeat
  - repeat
    - from q^L_{i-1} move toward q_{goal} along the m-line
  - until goal is reached or obstacle encountered at q^H_i
  - if goal is reached, exit
  - repeat
    - follow boundary
  - until q_{goal} is reached or q^H_i is re-encountered or m-line is re-encountered, x is not q^H_i, $d(x,q_{goal}) < d(q^H_i,q_{goal})$ and way to goal is unimpeded
  - if goal is reached, exit
  - if q^H_i is reached, return failure
  - else
    - $q^L_i = m$
    - i=i+1
    - continue

Bug2 Algorithm

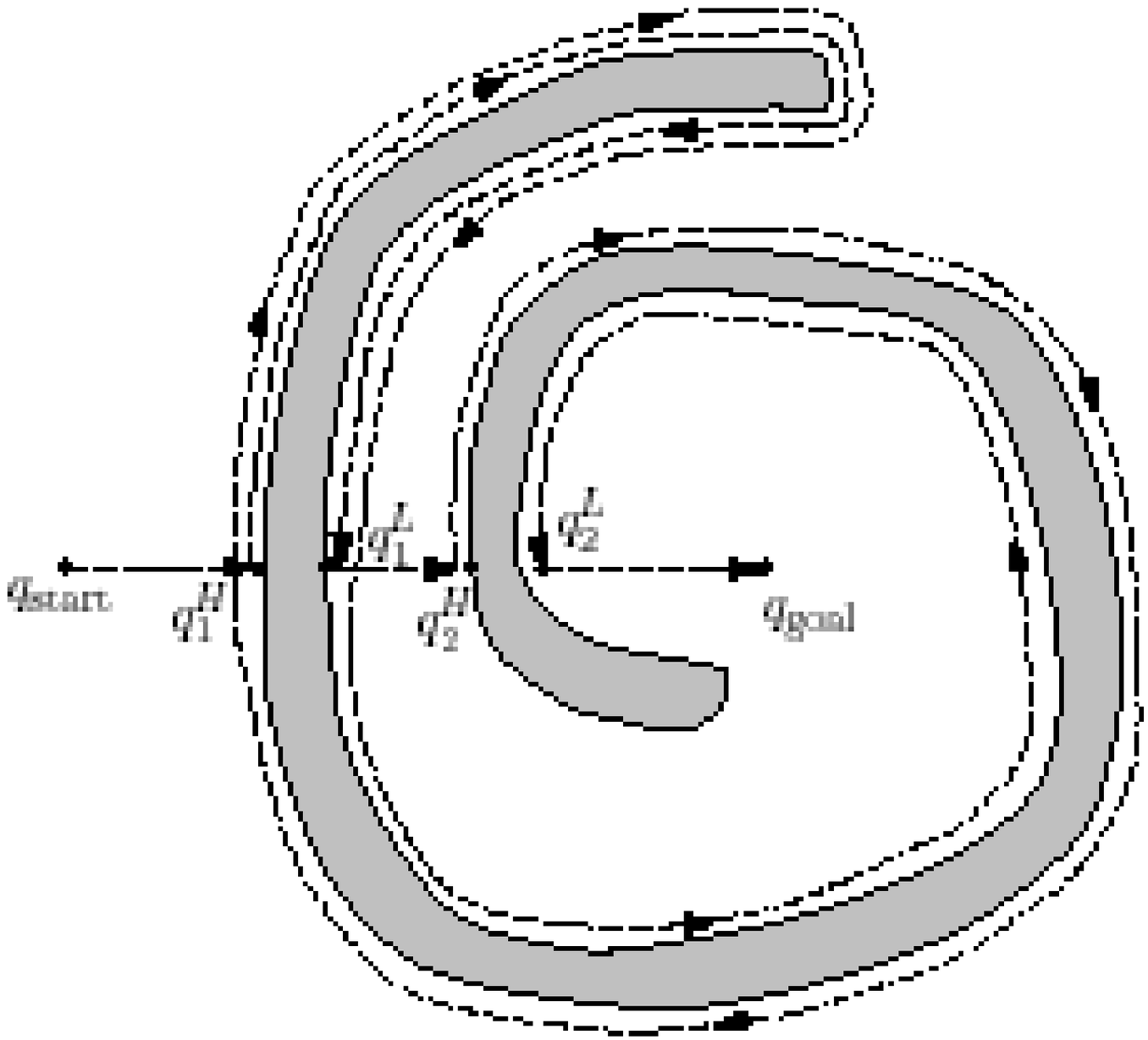


FIGURE 2.4. Bug2 Algorithm

Improvement:

- If is q_i^H re-encountered continue to follow boundary.

Shortest Path Algorithms

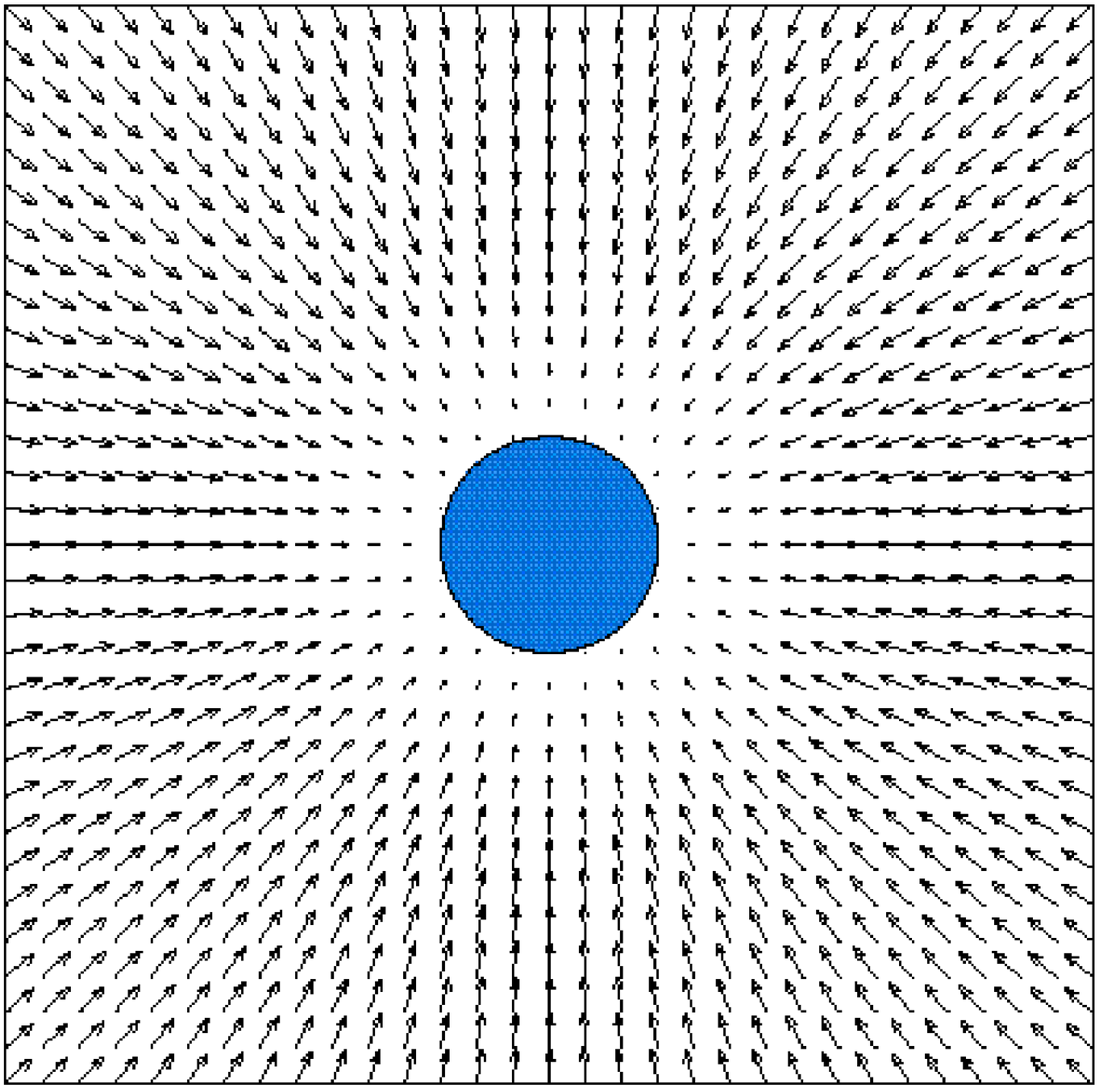
- 1) A* search
- 2) Breath-First Search
- 3) Depth-First search
- 4) D* algorithm
- etc.

- *Ref: https://en.wikipedia.org/wiki/Shortest_path_problem*

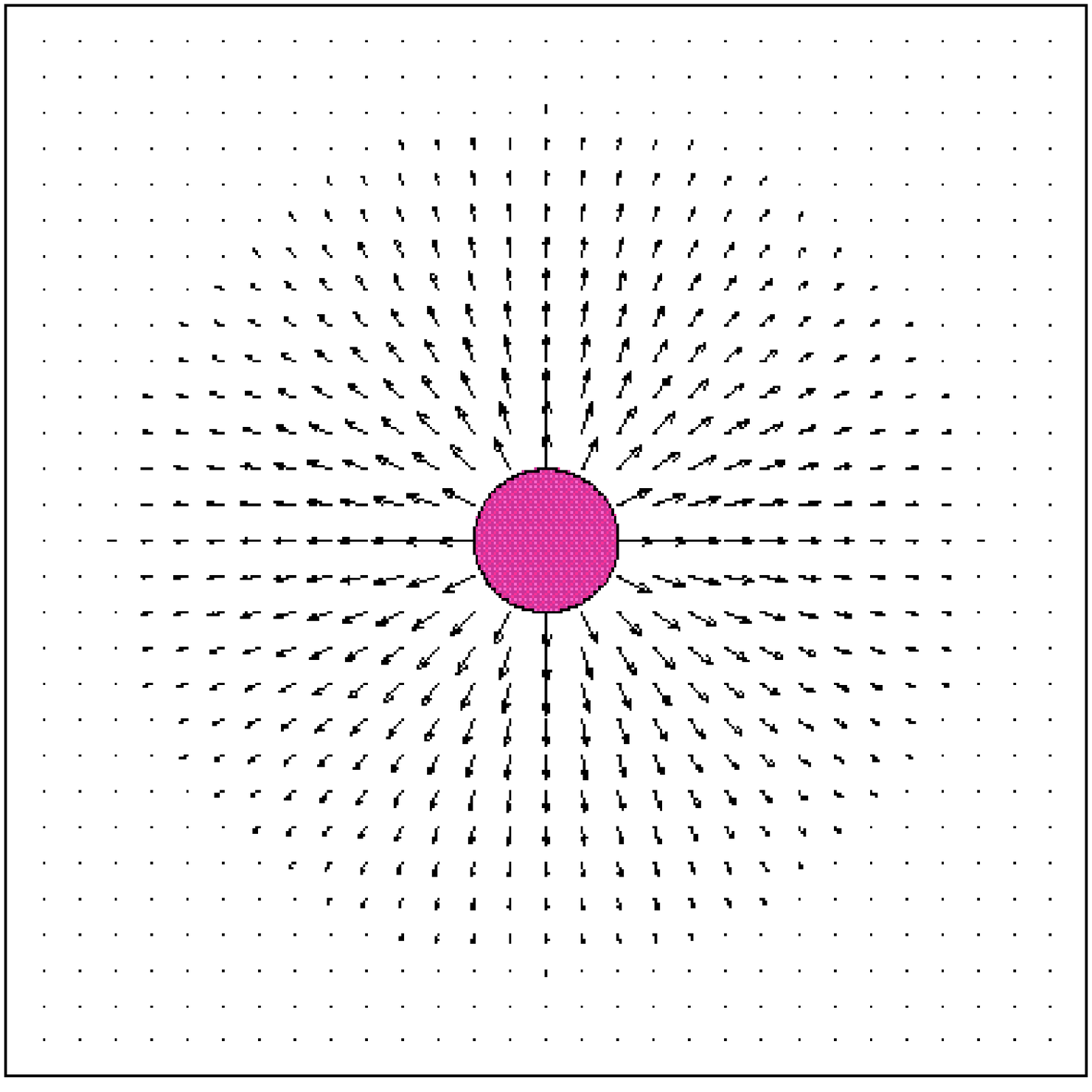
Potential Field Algorithms

- The goal emits a constant field that attracts the robot
- Obstacles emit a repulsive field that repels the robot.
- By adding and subtracting fields the robot's direction is determined

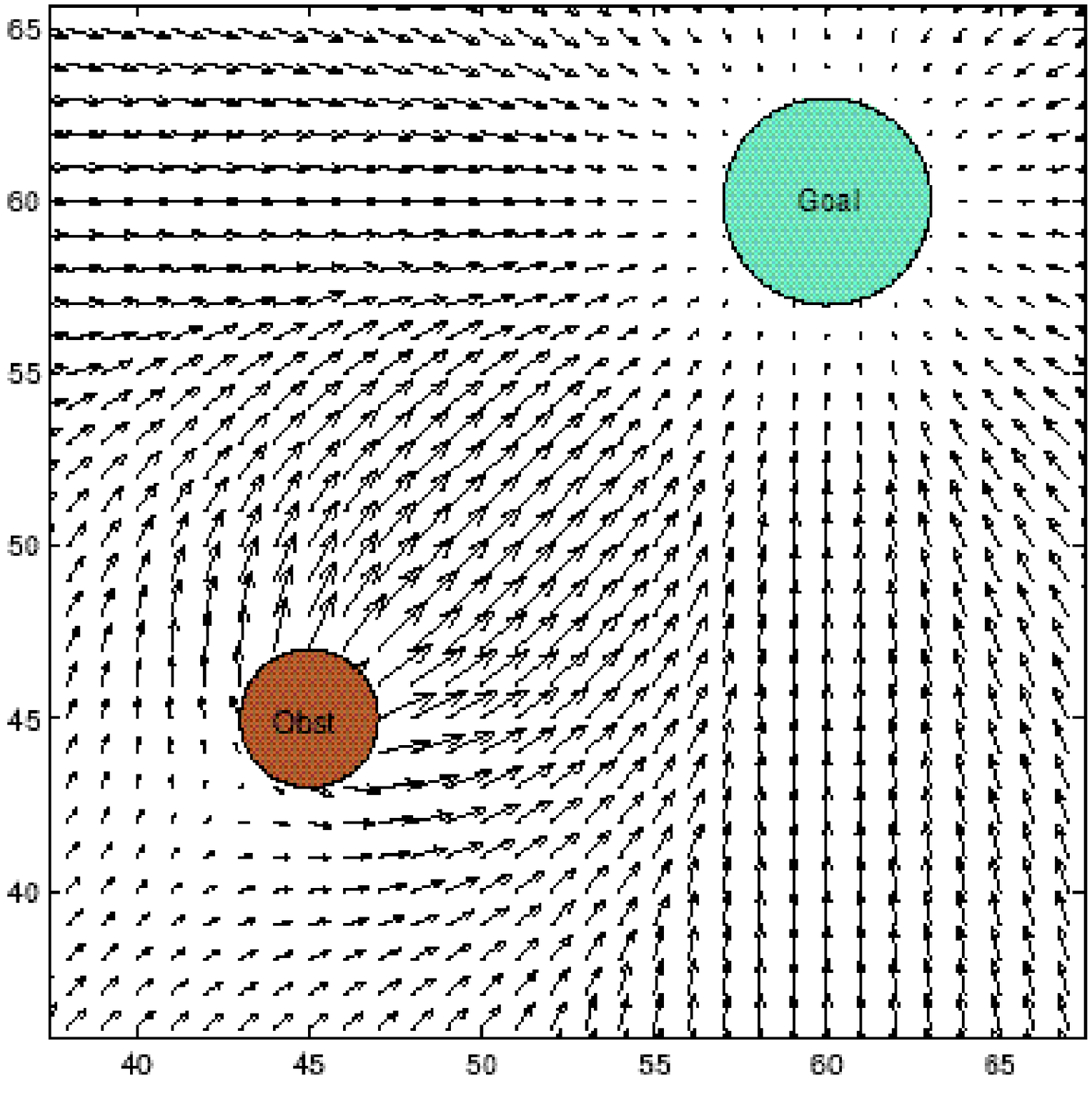
Attractive Potential Field



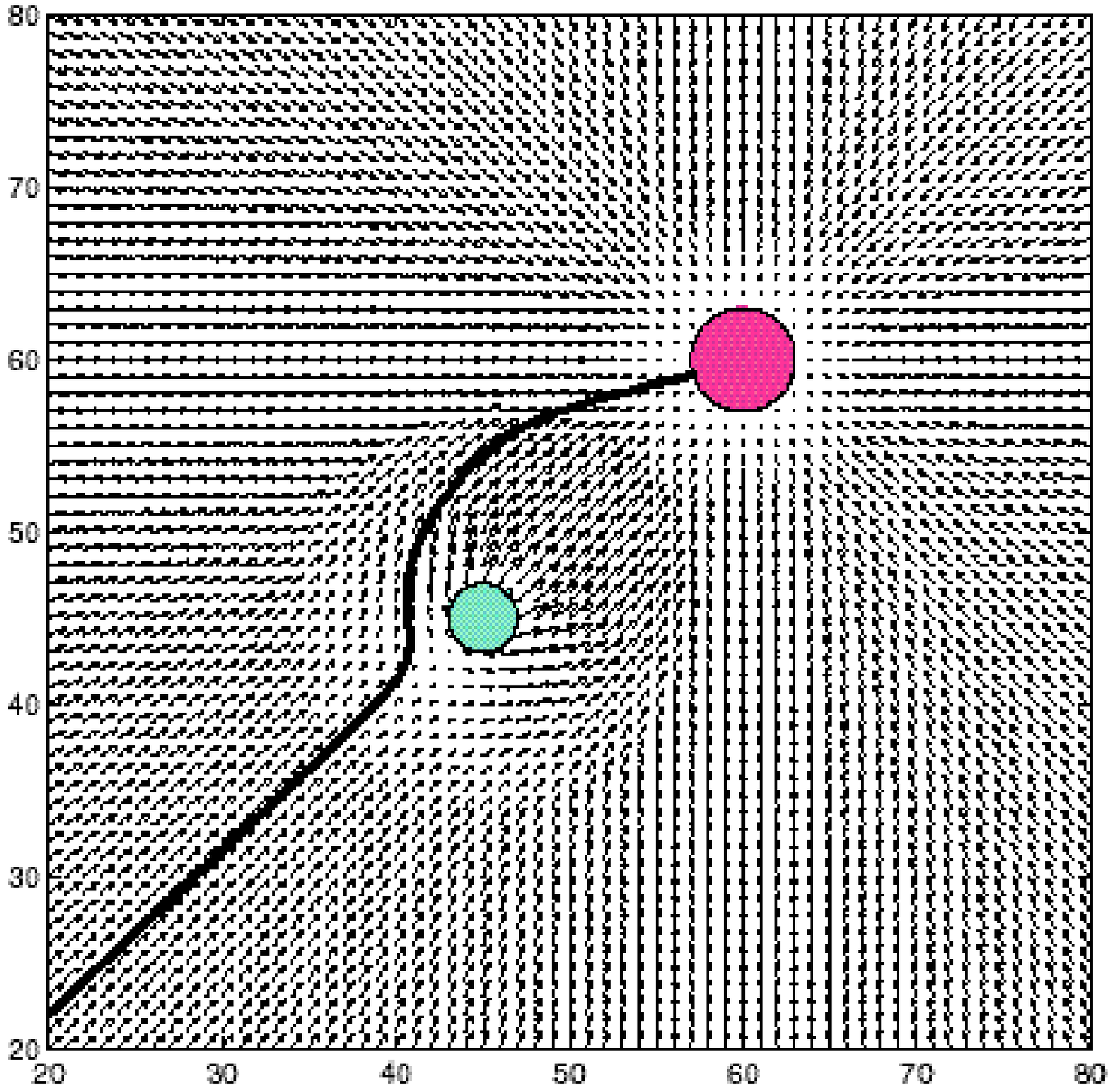
Repulsive Potential Field



Vector Sum of Two Fields



Resulting Robot Trajectory



Potential Fields

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

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

- A vector field can be the *gradient* of a potential field function $P(x,y)$:

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

Potential Fields

- The potential field $P(\mathbf{x})$ is defined over the environment.
- Sensor information **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 **u** is determined to follow that gradient.

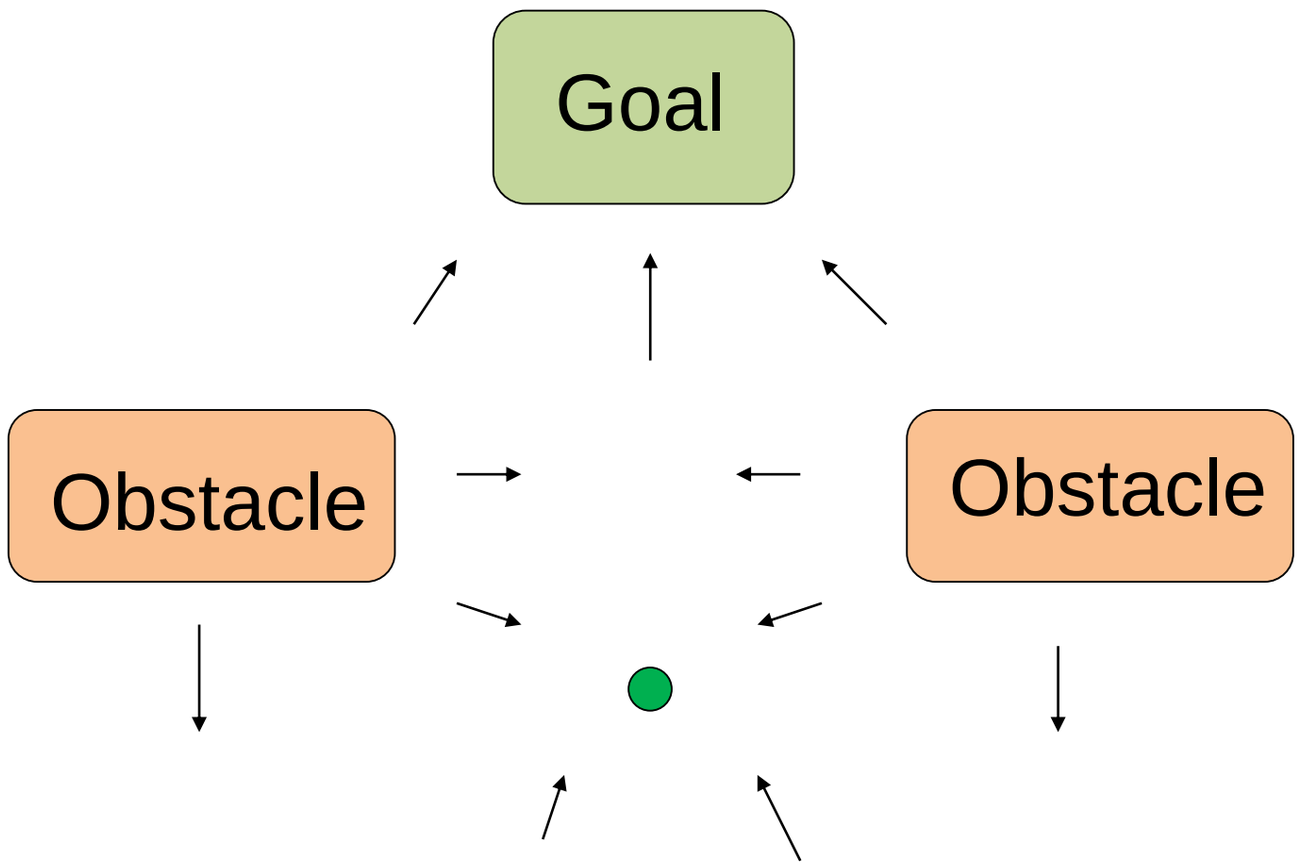
Attraction and Avoidance

- **Goal**: Surrounded with an attractive field.
- **Obstacles**: Surrounded 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

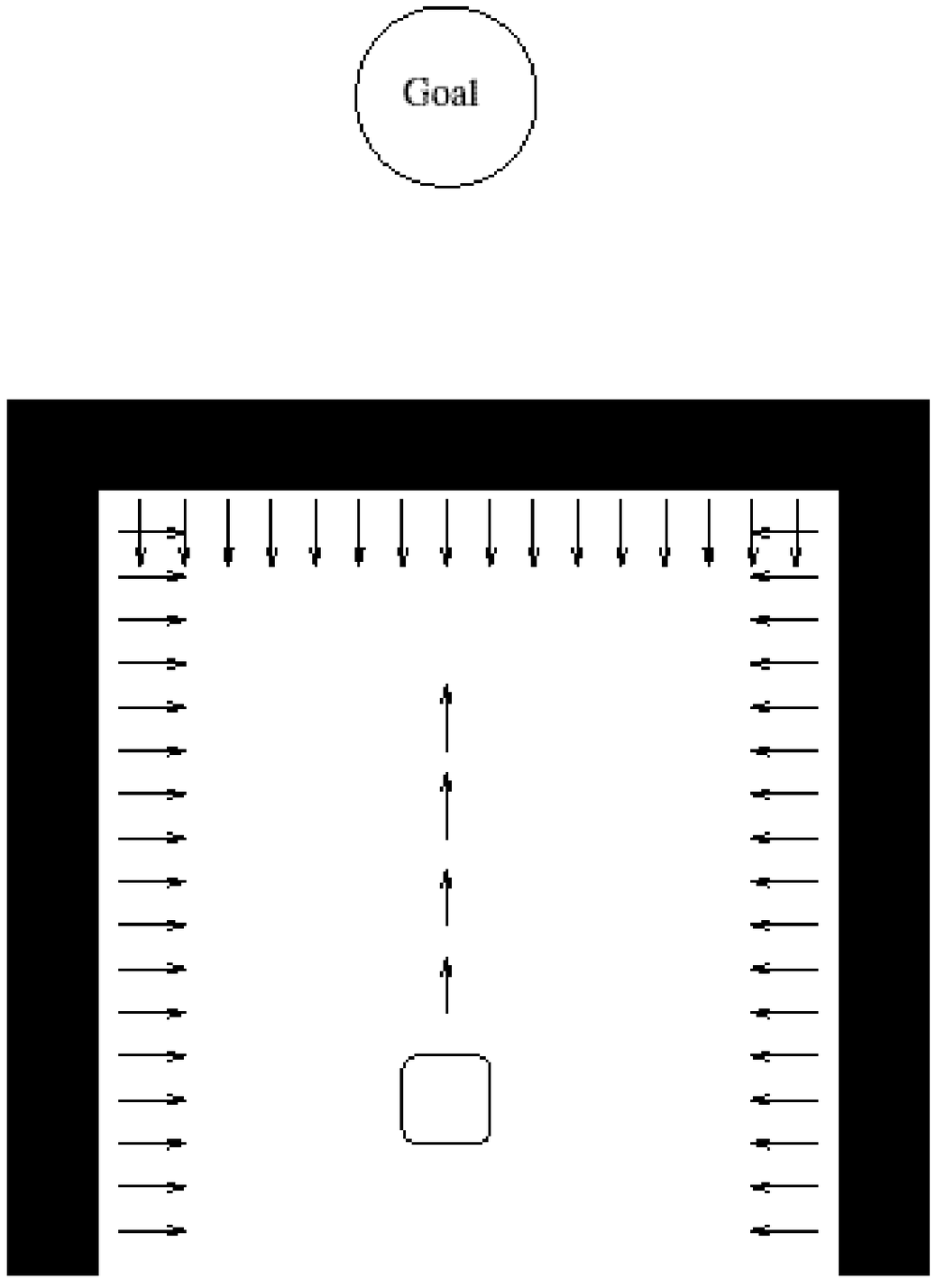
- *Local minima*
  - Attractive and repulsive forces can balance, so robot makes no progress.
  - Closely spaced obstacles, or dead end.
- *Unstable oscillation*
  - The dynamics of the robot/environment system can become unstable.
  - High speeds, narrow corridors, sudden changes.

Local Minimum Problem



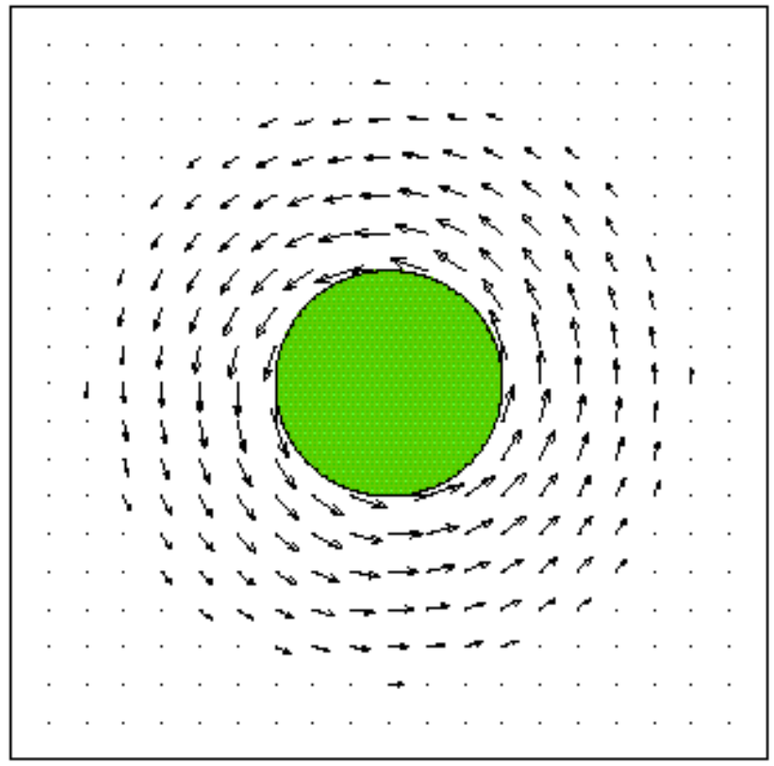
Box Canyon Problem

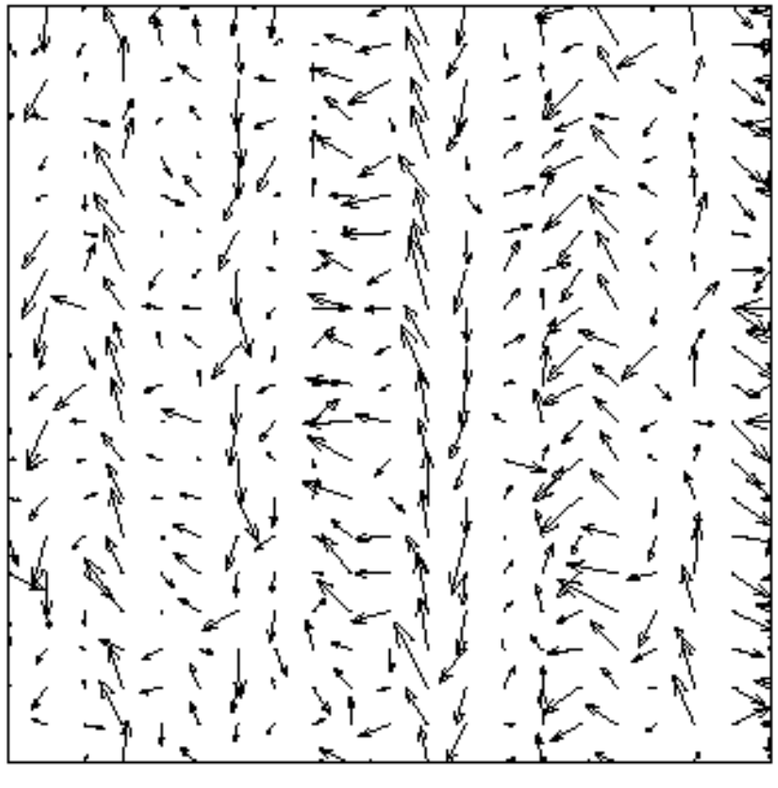
- Local minimum problem
  - *Robot avoids past* potential field thus getting stuck in dead ends or canyons.



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
- Adding a *random field* gets the robot unstuck.
  - Avoids some local minima.





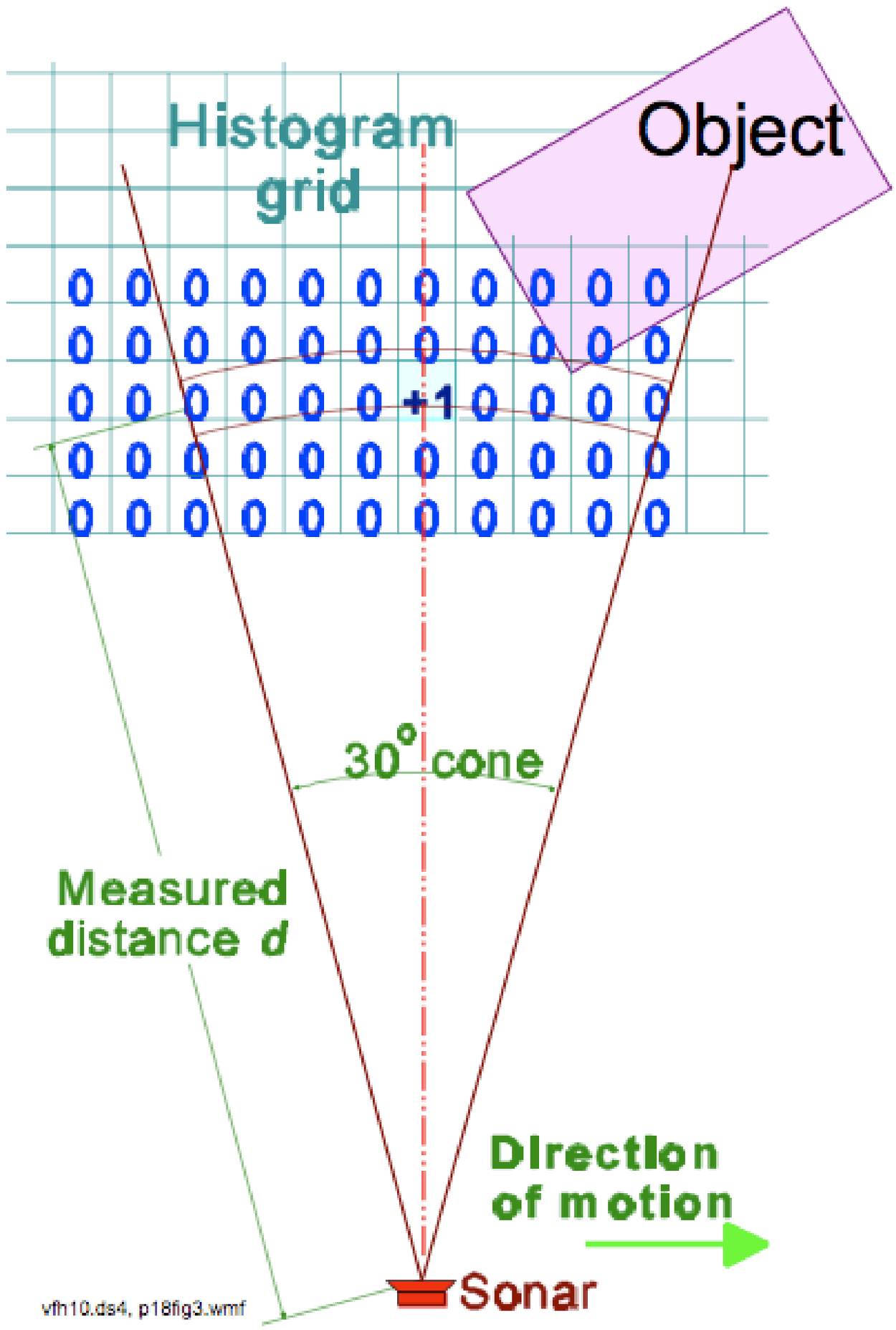
Vector Field Histogram: Fast Obstacle Avoidance

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

CARMEL: Cybermotion K2A

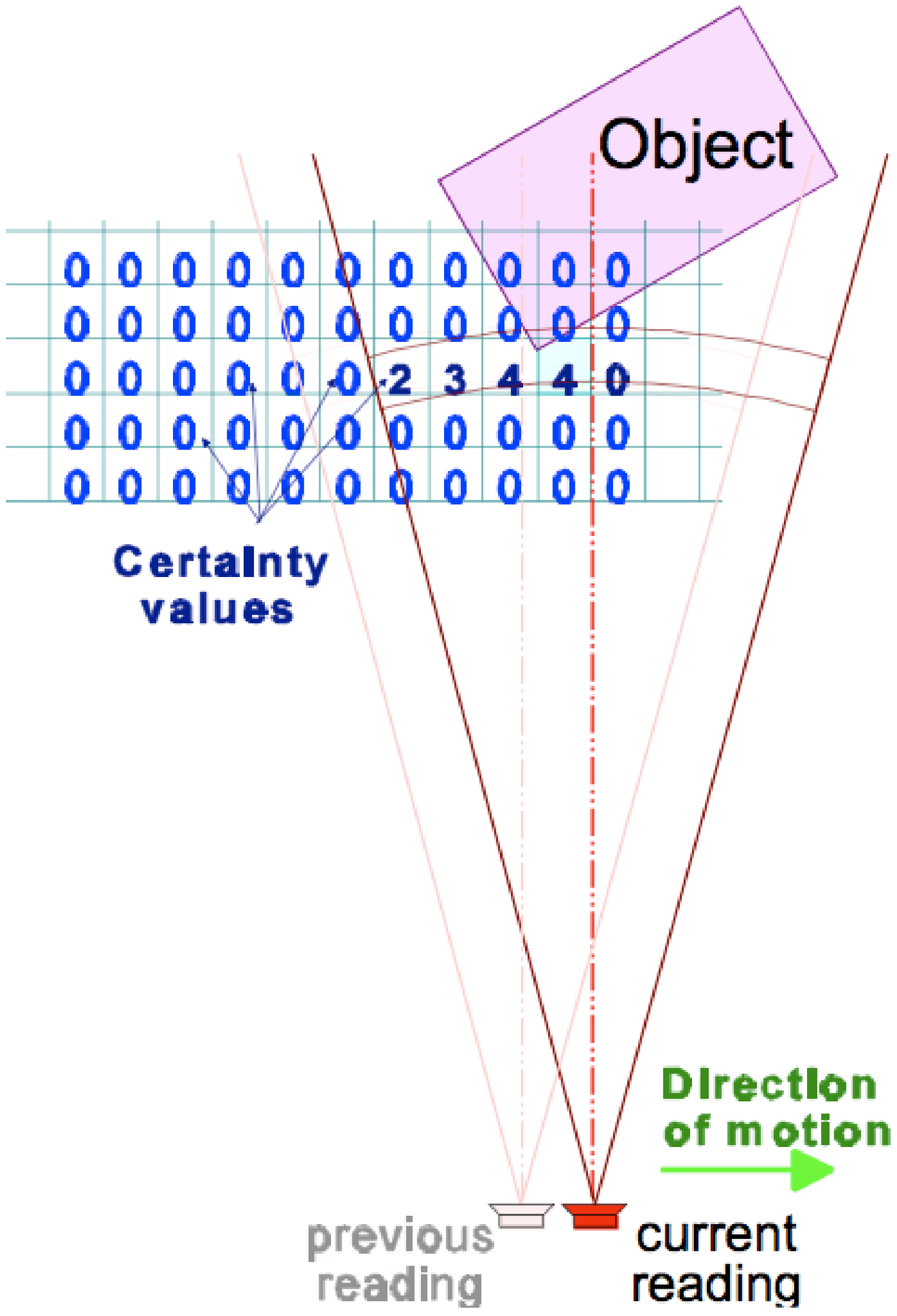
1. Occupancy Grid

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



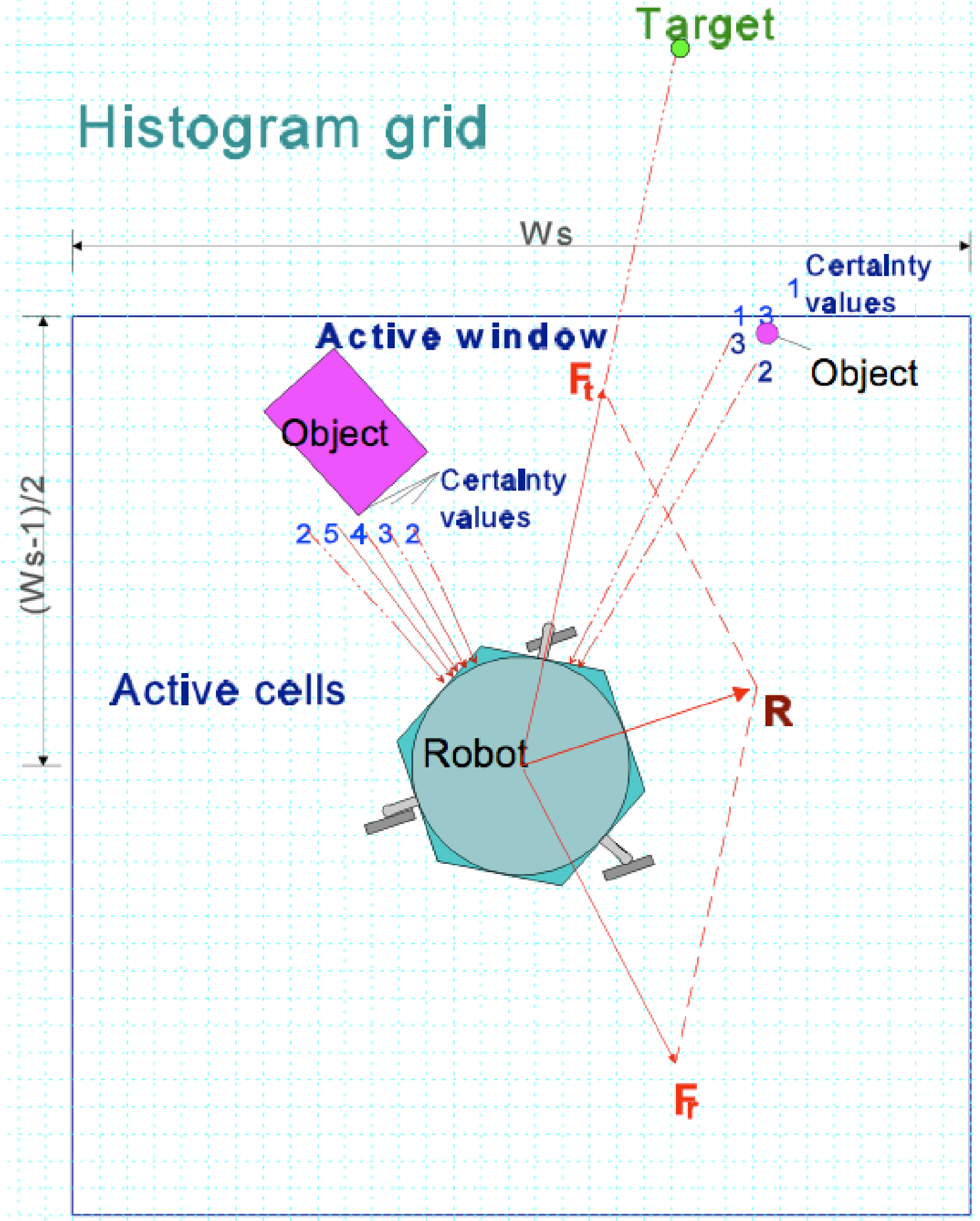
1. Occupancy Grid

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



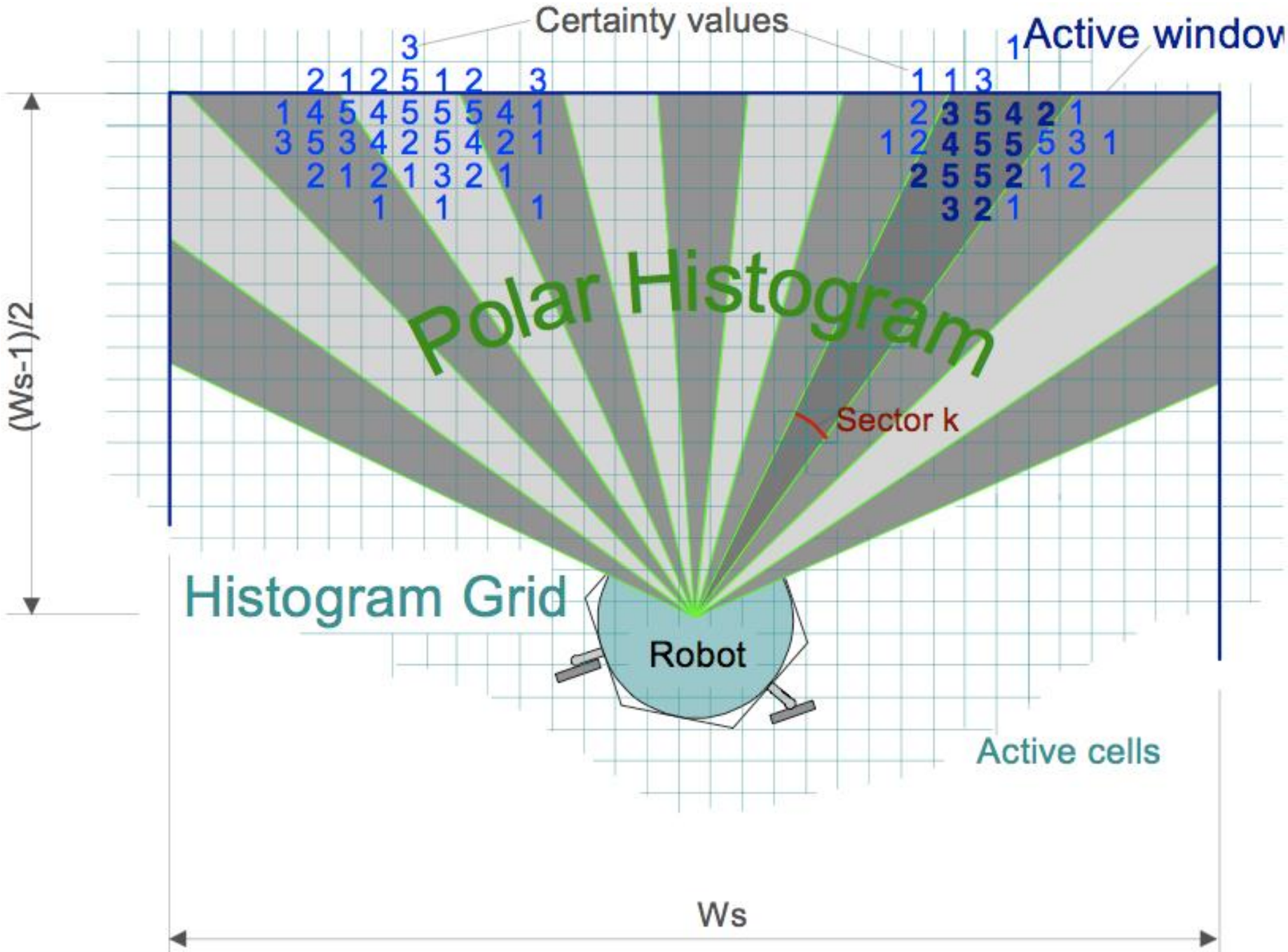
Virtual Force Field:

- Active window Ws×Ws around the robot
- Grid alone used to define a "virtual force field"



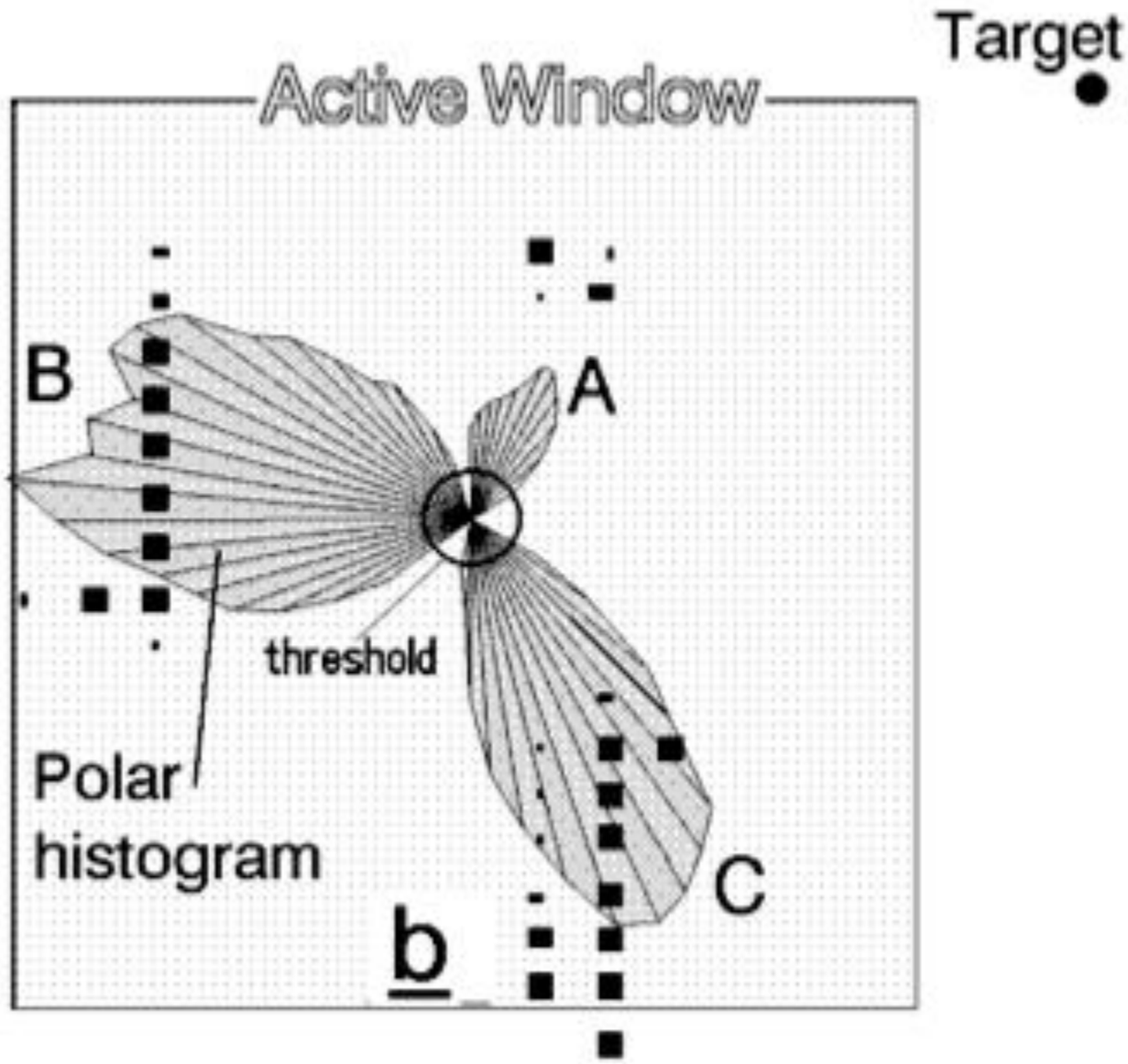
2. Polar Histogram

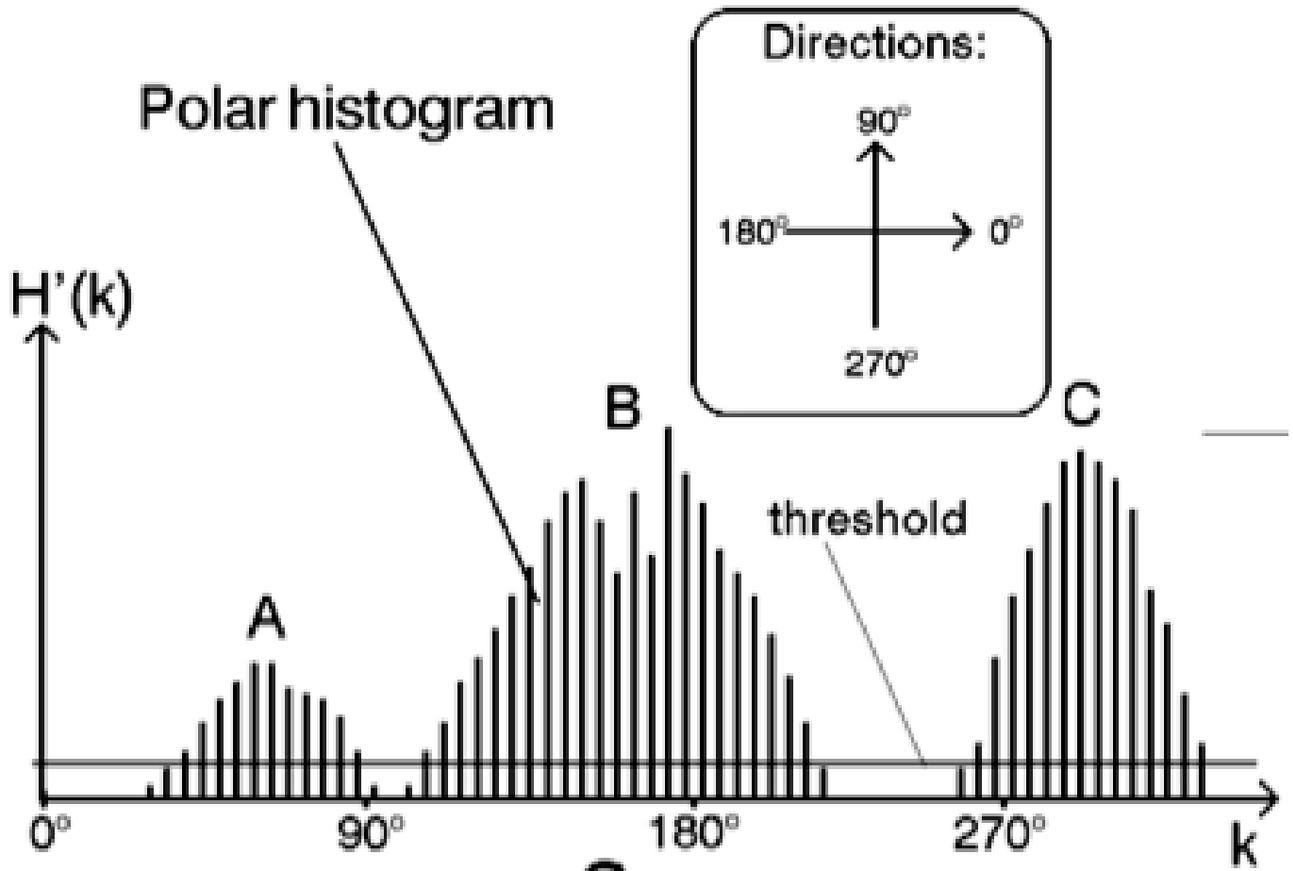
- Aggregate obstacles from occupancy grid according to direction from robot.



2. Polar Histogram

- Weight by occupancy, and inversely by distance.

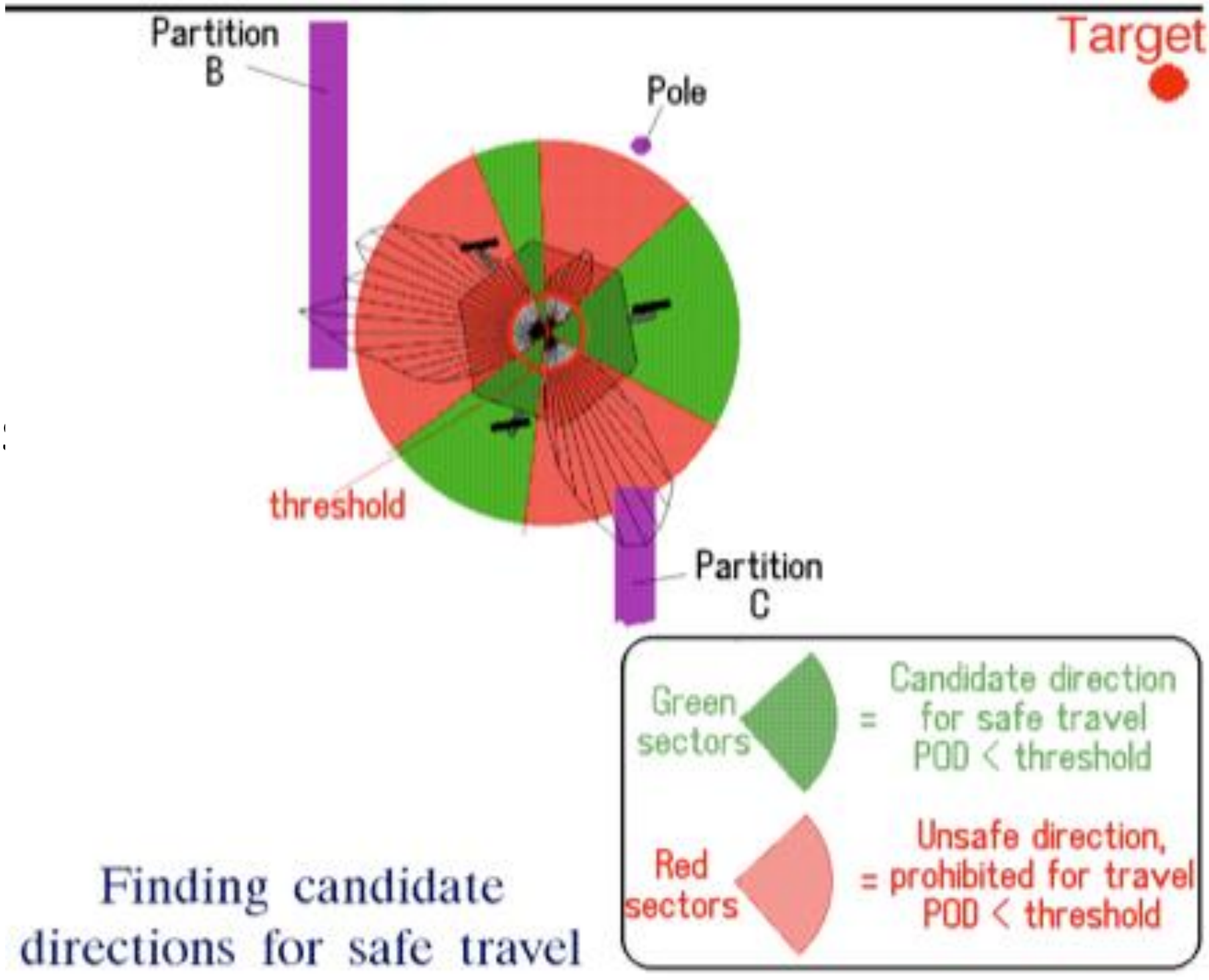




3. Directions for Safe Travel

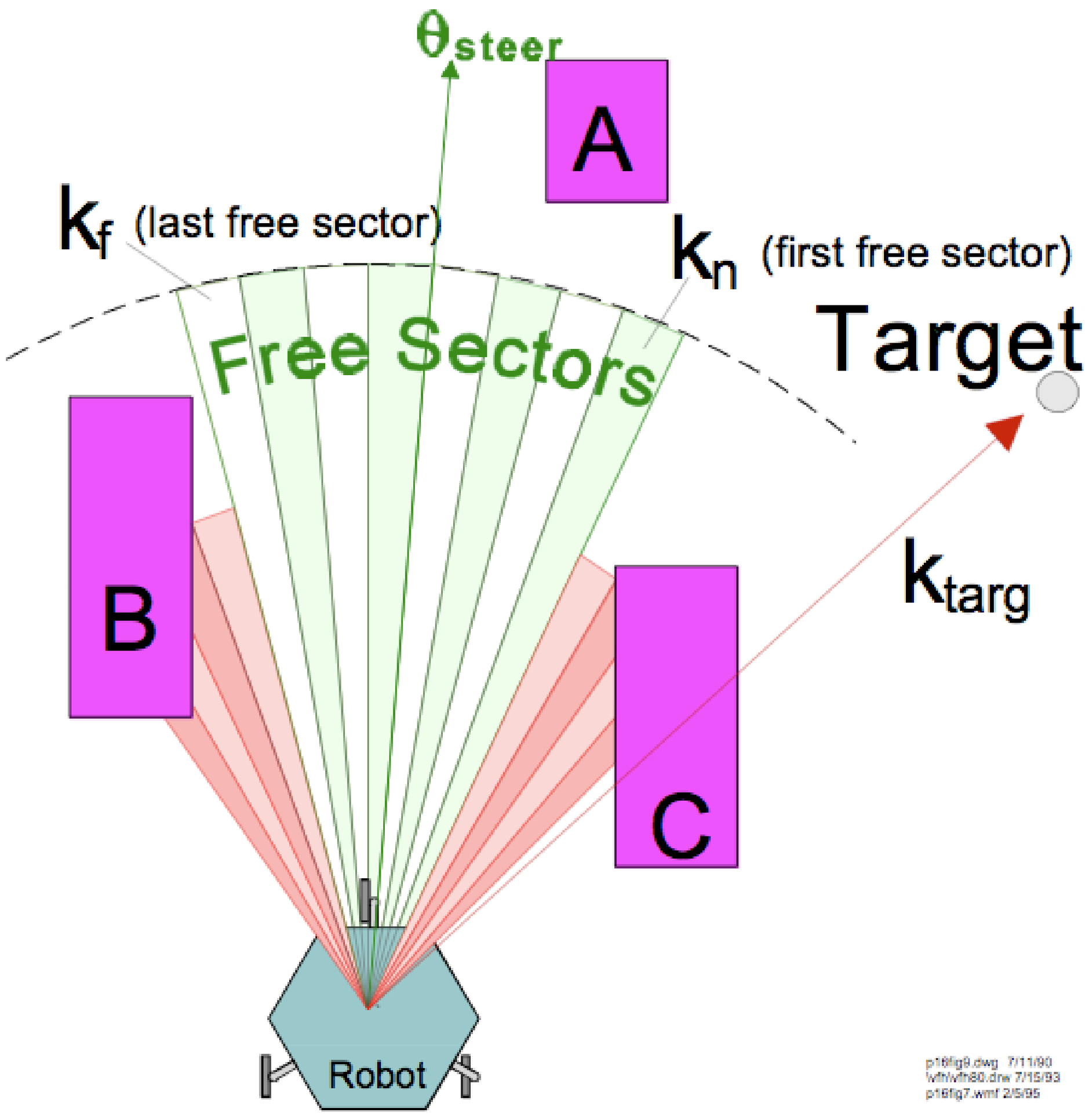
- Threshold determines safety.

- Multiple level of noise elimination.



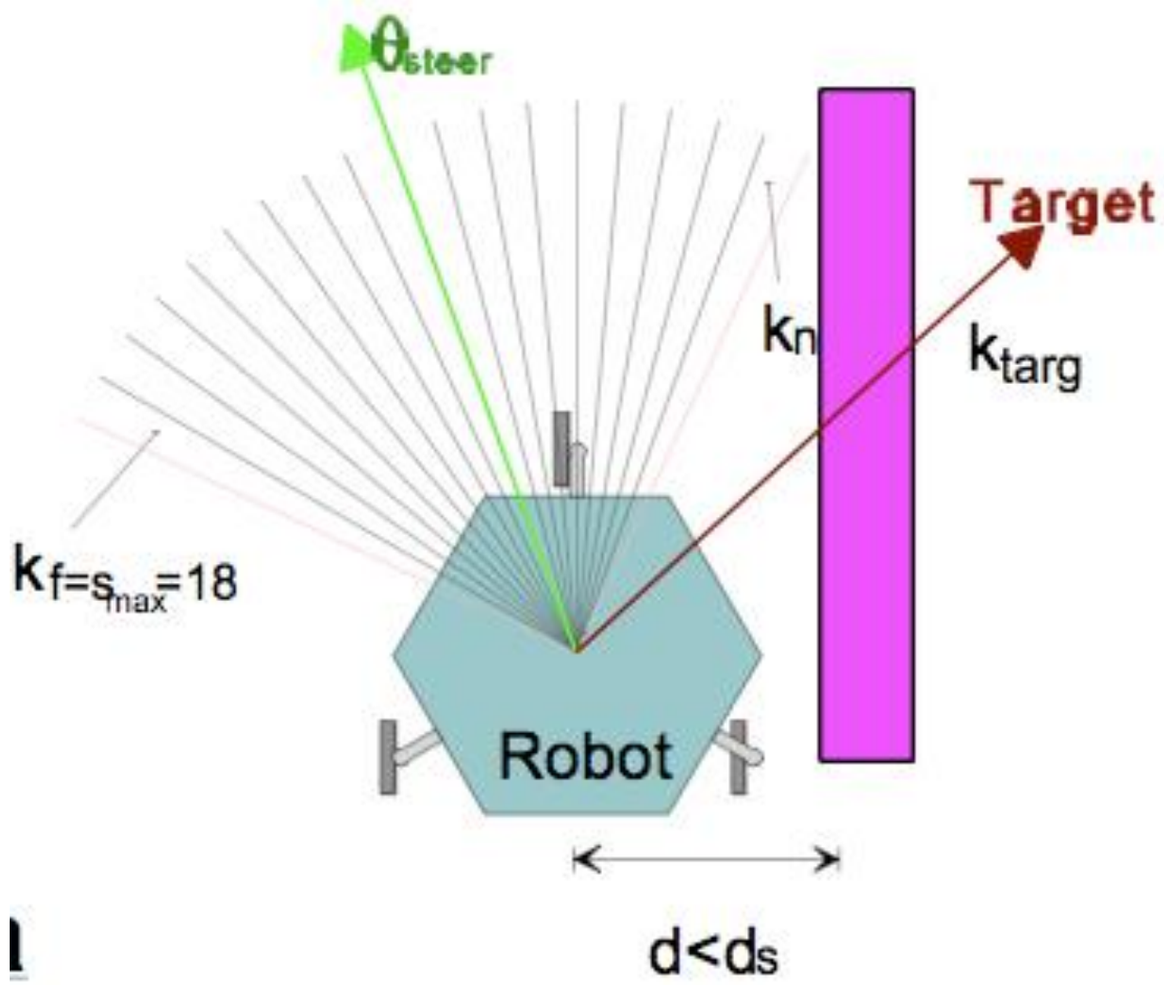
Finding candidate directions for safe travel

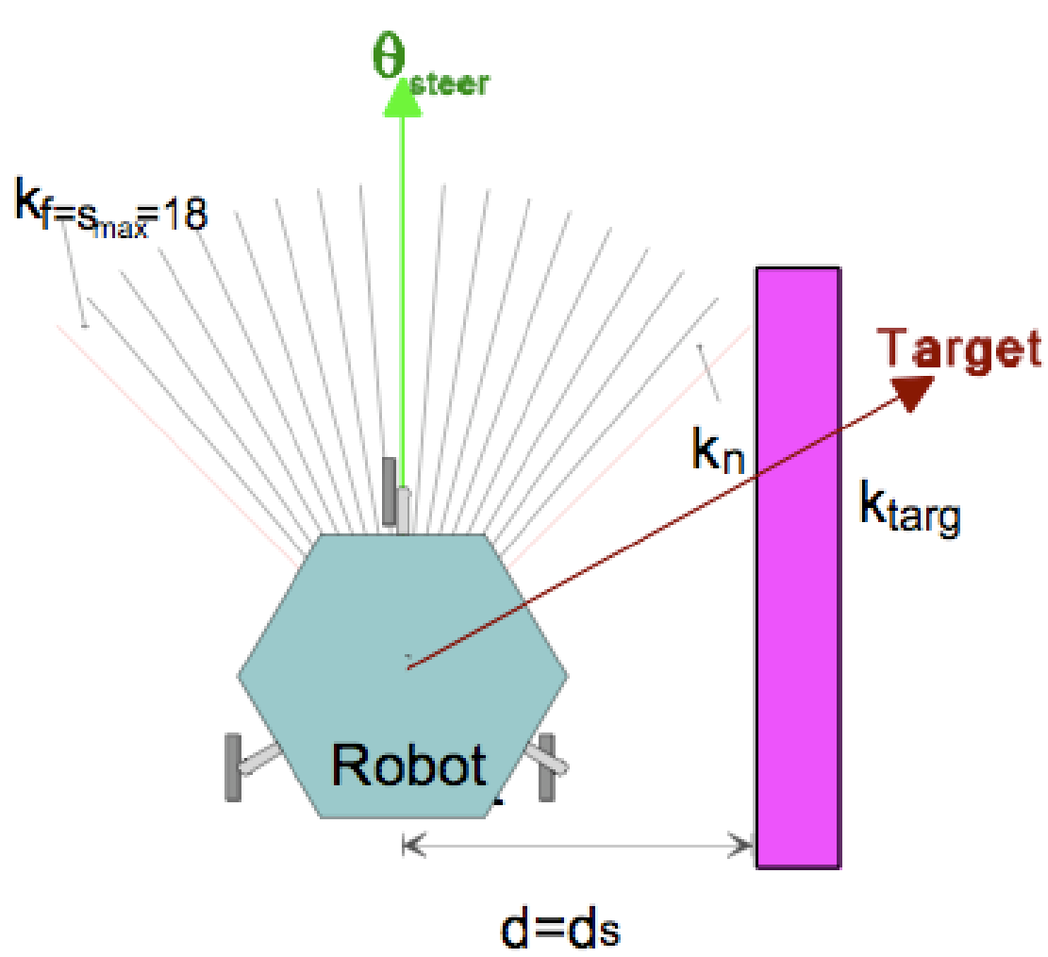
3. Steer to center of safe sector

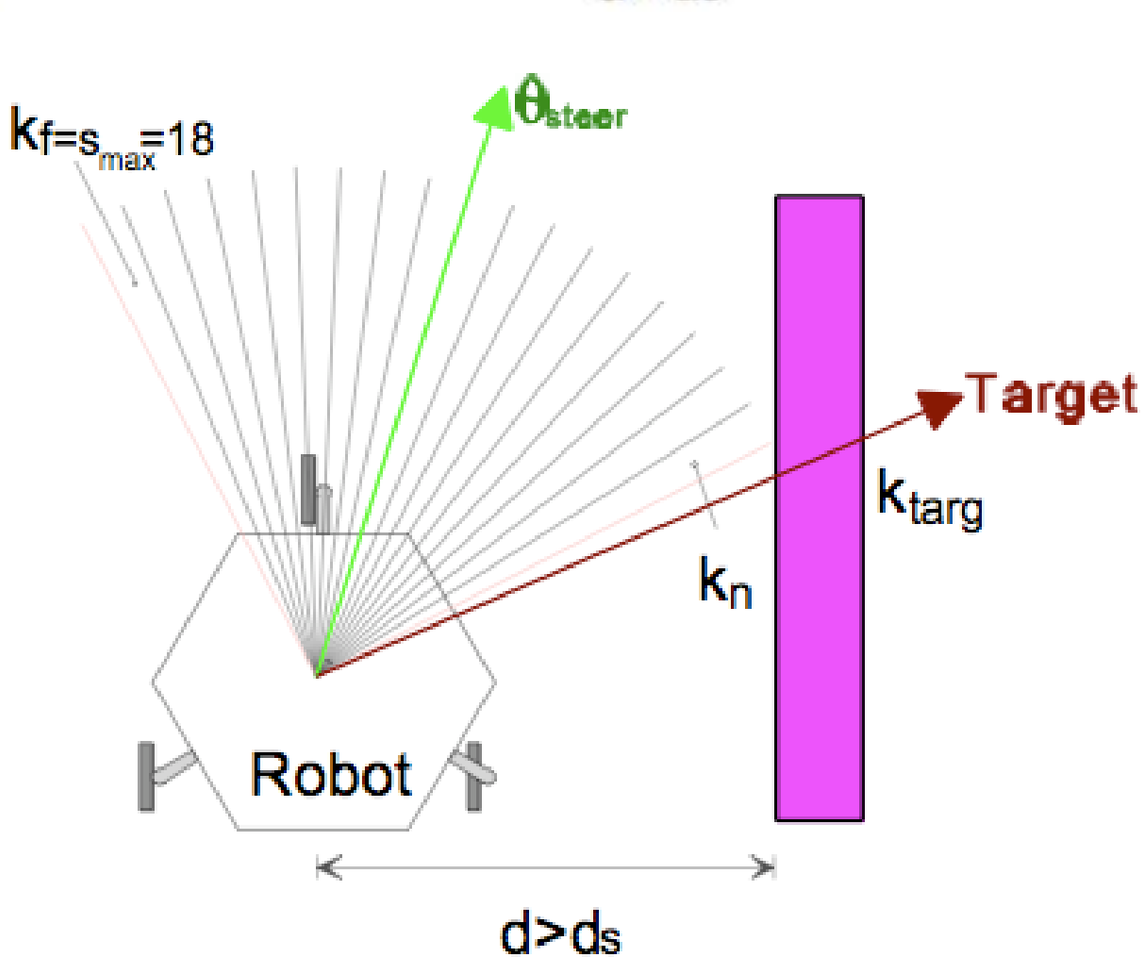


Leads to natural wall-following

- Threshold determines offset from wall.

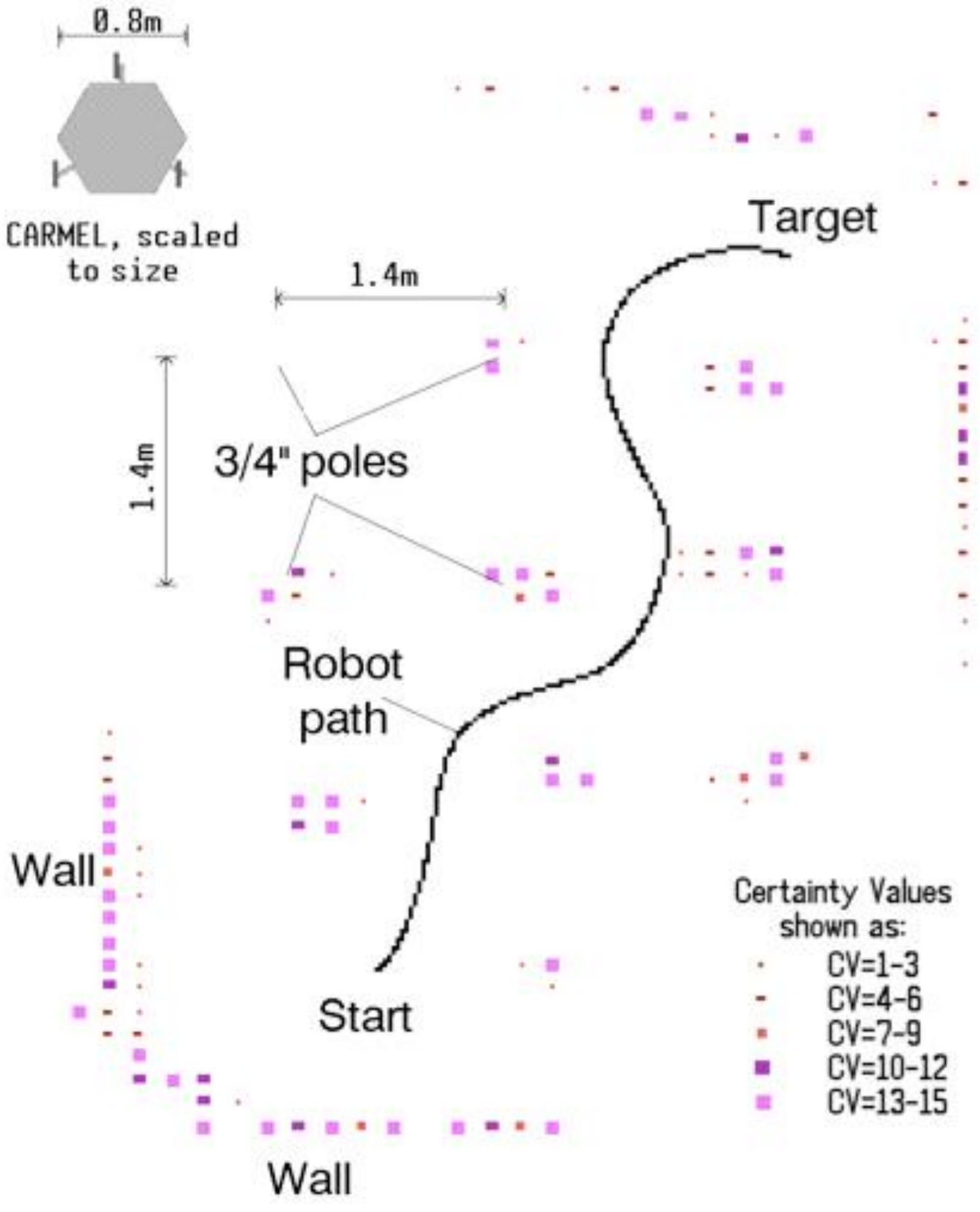






Smooth, Natural Wandering Behavior

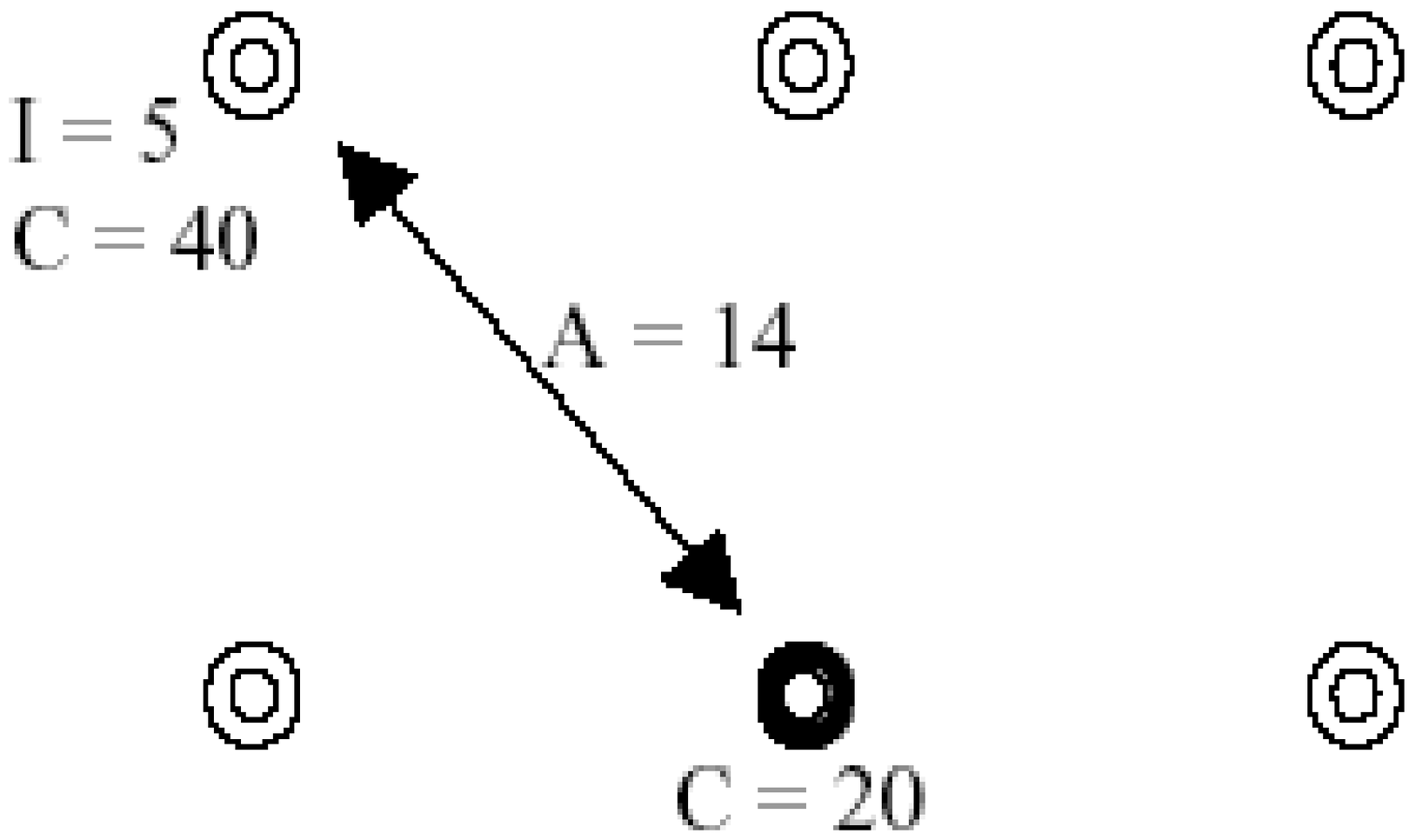
- Potentially quite fast!
- 1 m/s or more!

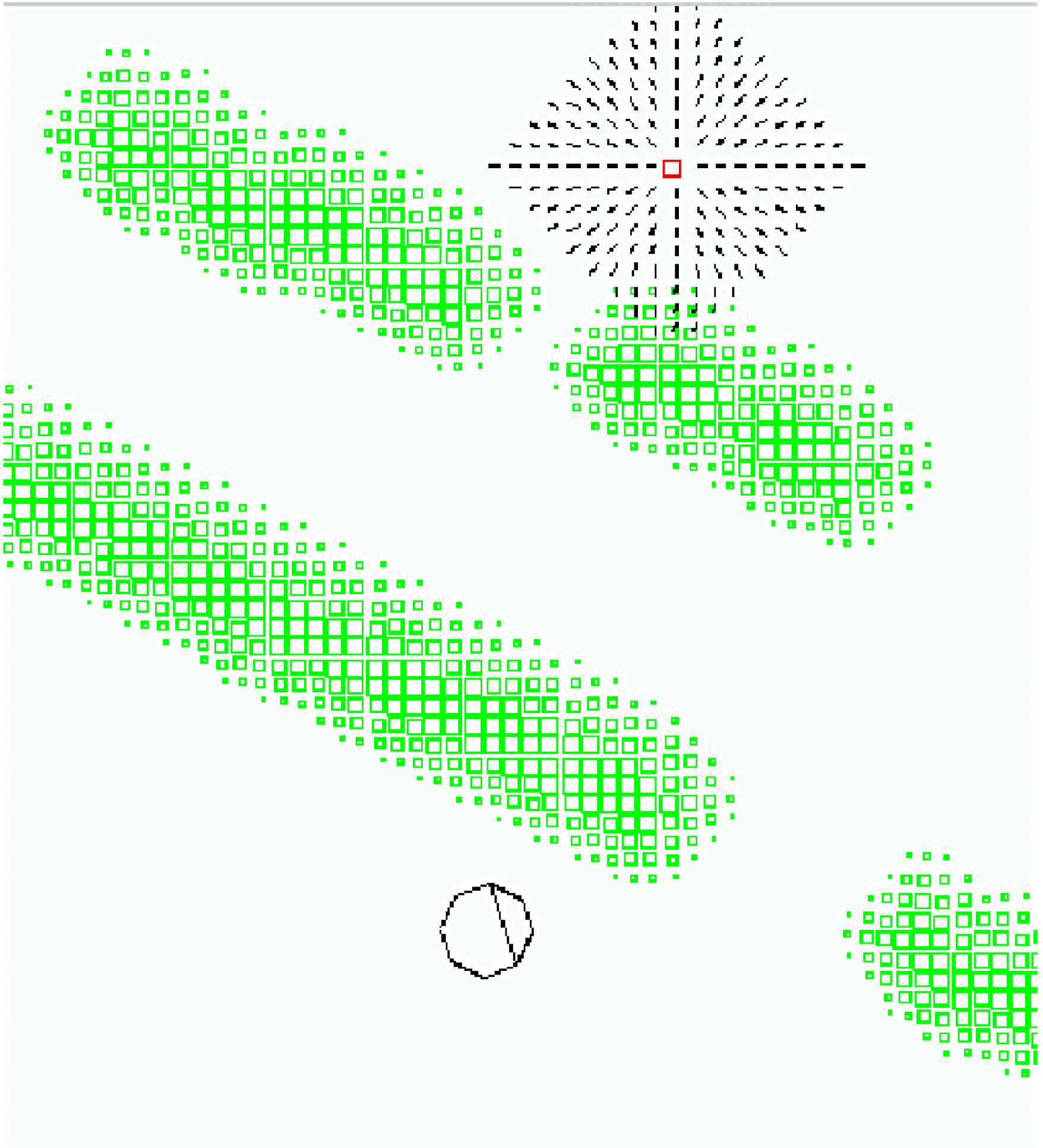


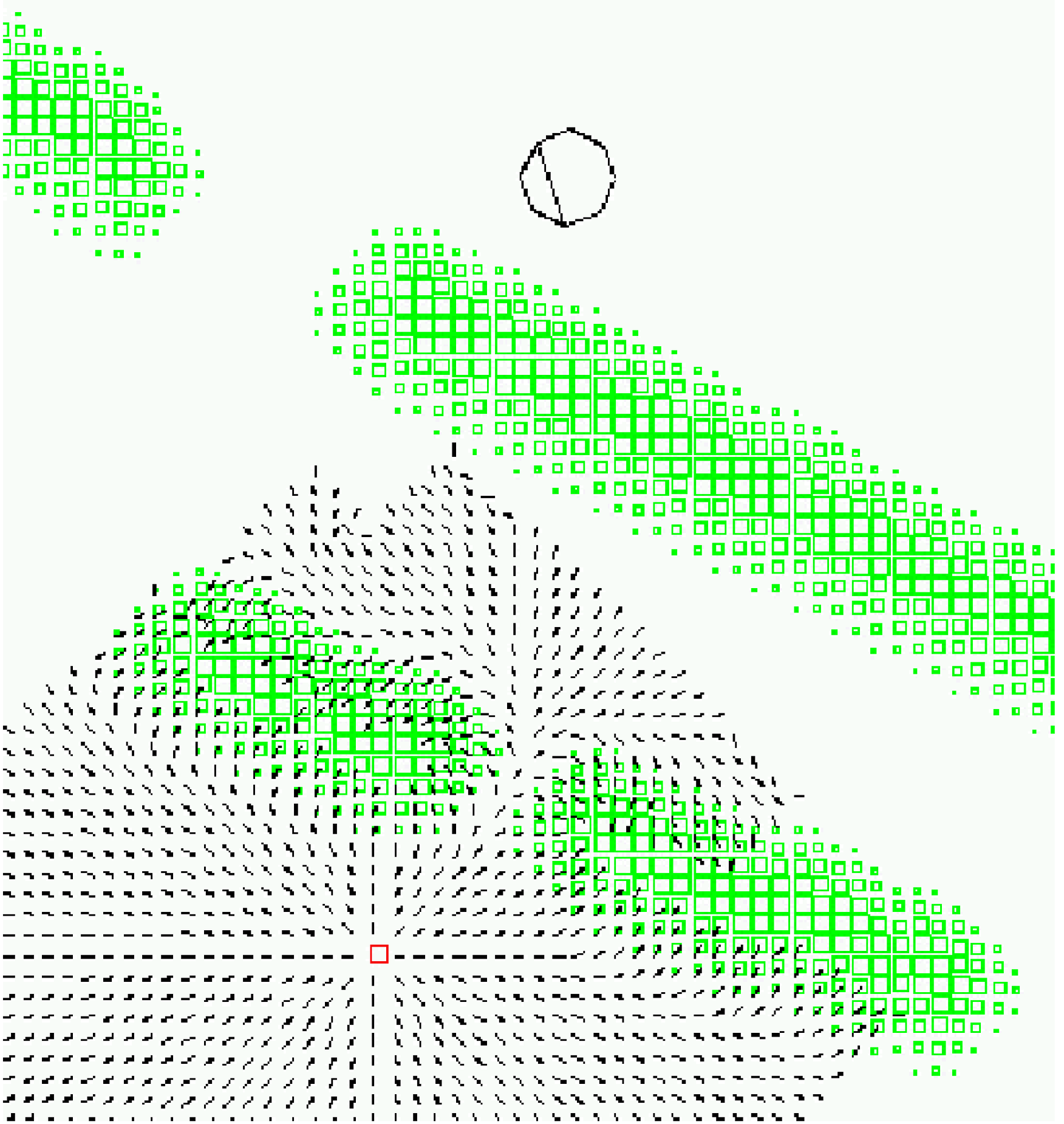
Wavefront Algorithm

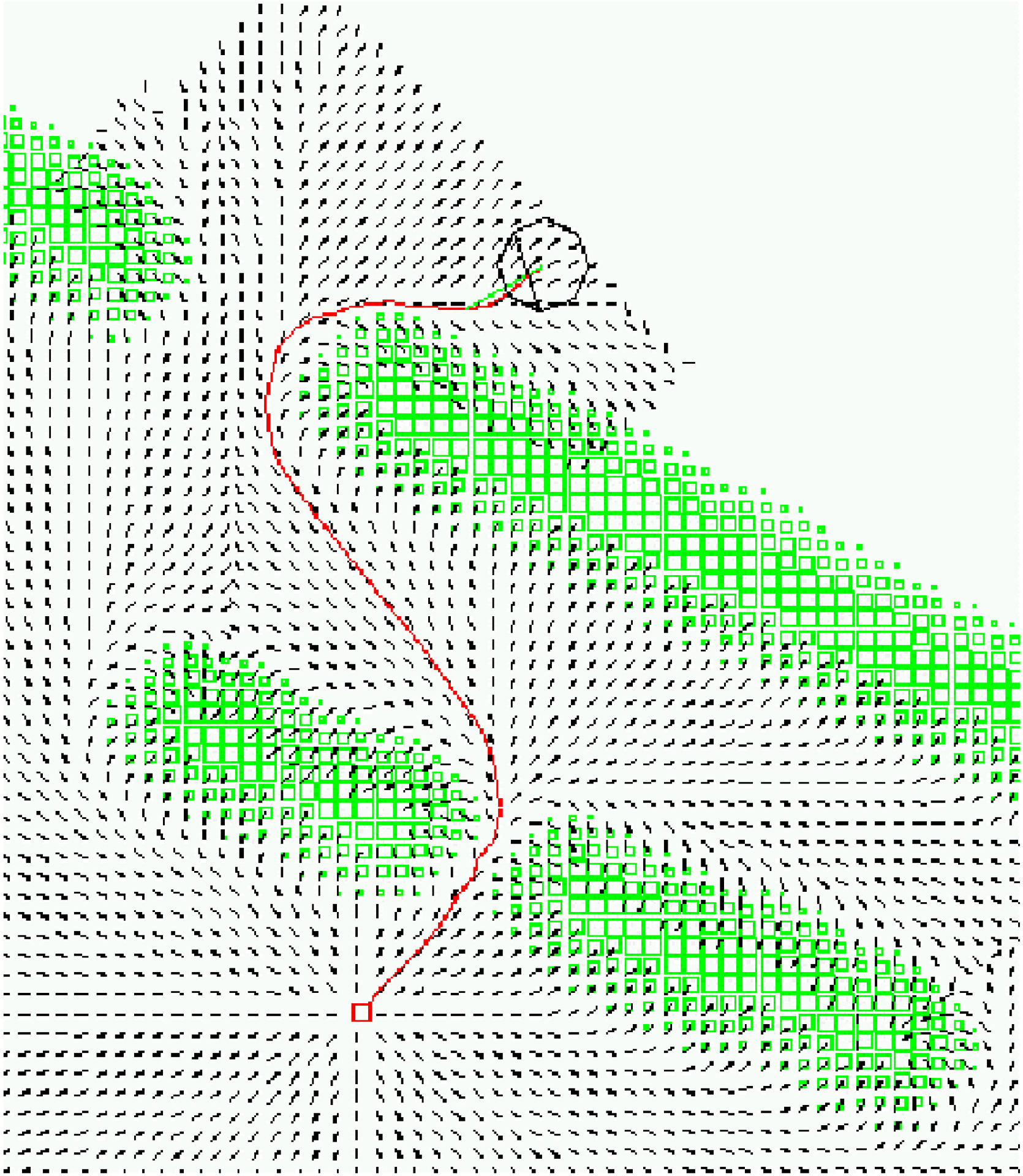
- The wavefront algorithm considers all routes to the goal with the same cost $N(p)$.
- The *wavefront* algorithm propagates from the goal to the current location.
  - An active point updates costs of its 8 neighbors.
  - A point becomes active if its cost decreases.
  - Continue to the robot's current position.

Wavefront Propagation









Wavefront Algorithm

- Recalculates $N(p)$ at 10 Hz
- 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.

Summary

- To localize or not?
- Noise, noise, noise
- Odometry
- Map representations
- Path Planners
- Potential fields