DD2410

Lecture slides

Navigation

Navigation

- Where am I?
  - See **Localization**, and **Mapping and SLAM**
- Where do I need to go?
  - High level reasoning not covered in this course
- How do I get there?
  - Global planning (see **Planning**)
  - Local planning (focus here)

Navigation

- Control is often decomposed into high level functions (often called behaviors or skills)
  - wall following
  - exploration
  - door traversal, . . .

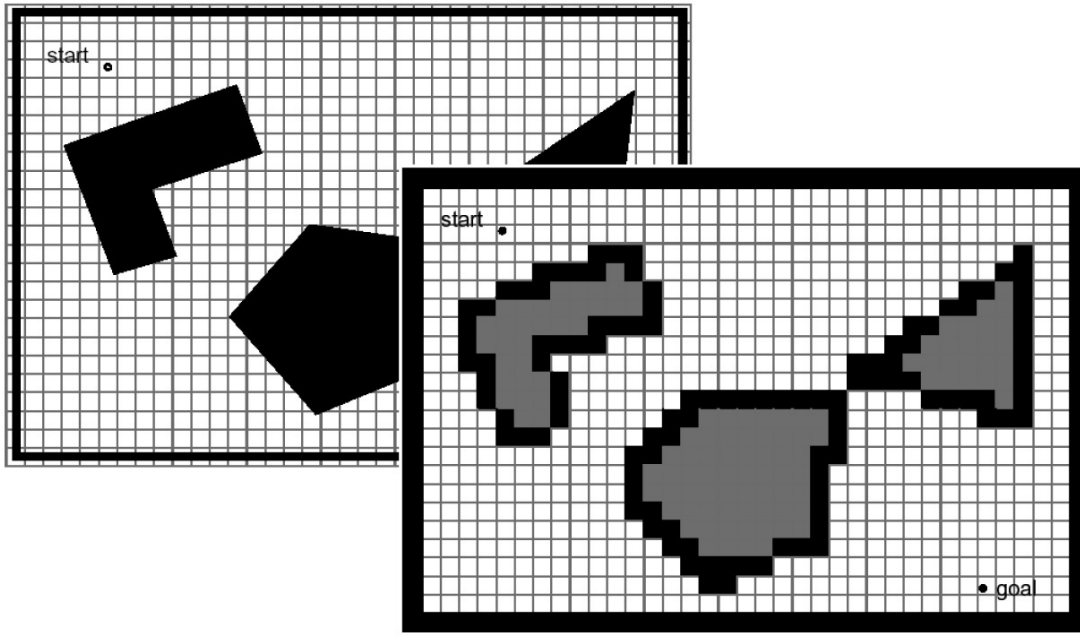
- Three parts of "how do I get there"
  - Global motion planning (deliberate)
  - Obstacle avoidance / local planning (reactive)
  - The low level control of the motion

Global motion planning

- Typical assumption: there is an adequate model/map of the environment for navigation
  - the model could be topological, metric or a mixture
- Steps:
  - Generate a representation of the map for planning
  - Populate the map with a “distance metric”
  - Perform search from the present configuration to the goal configuration

Typical map representation

- Occupancy grids
- Easy to implement and plan in
- Explicitly represents both obstacles and free space (a landmark map does not!)



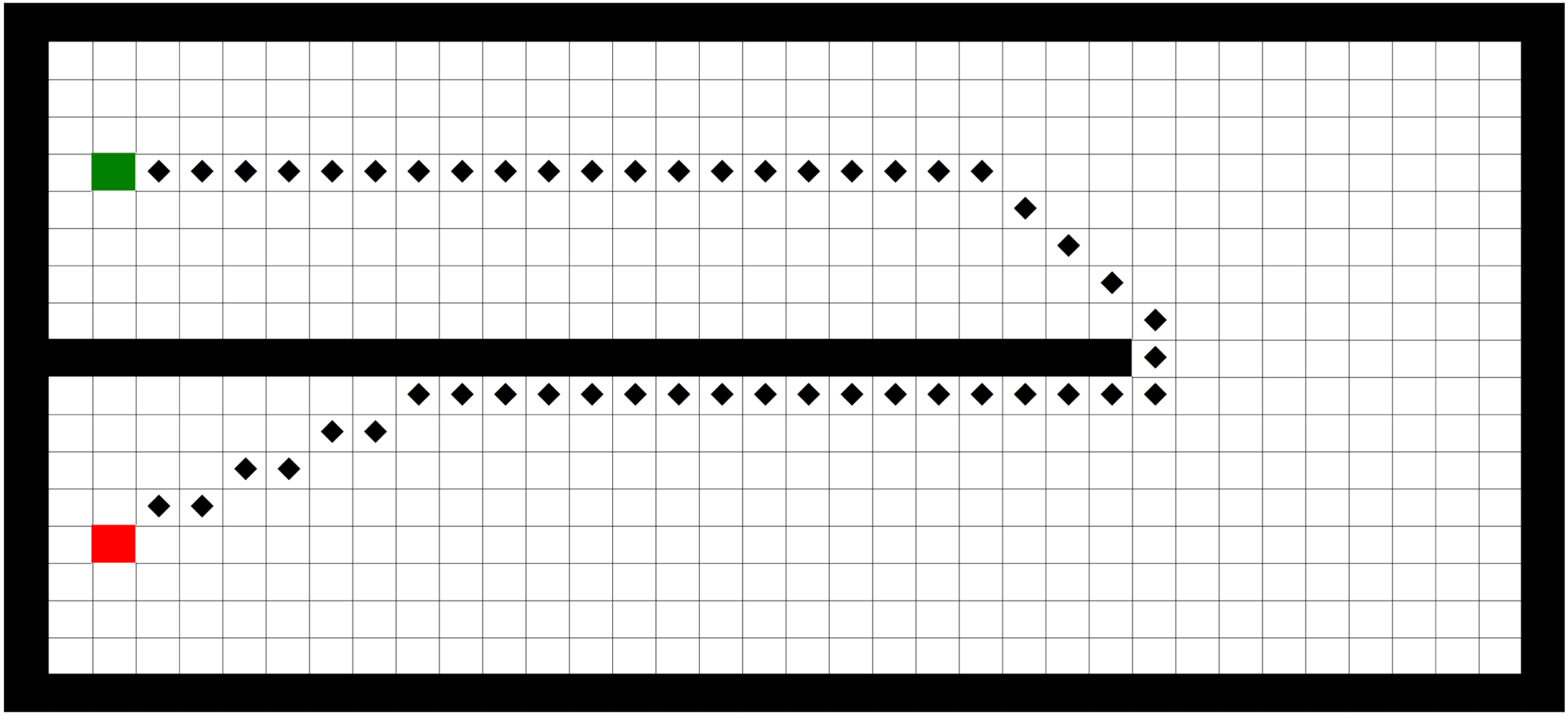
Some practical issues with grid based planning

- Path close to obstacles
- Generated paths are aligned to the grid and often not smooth

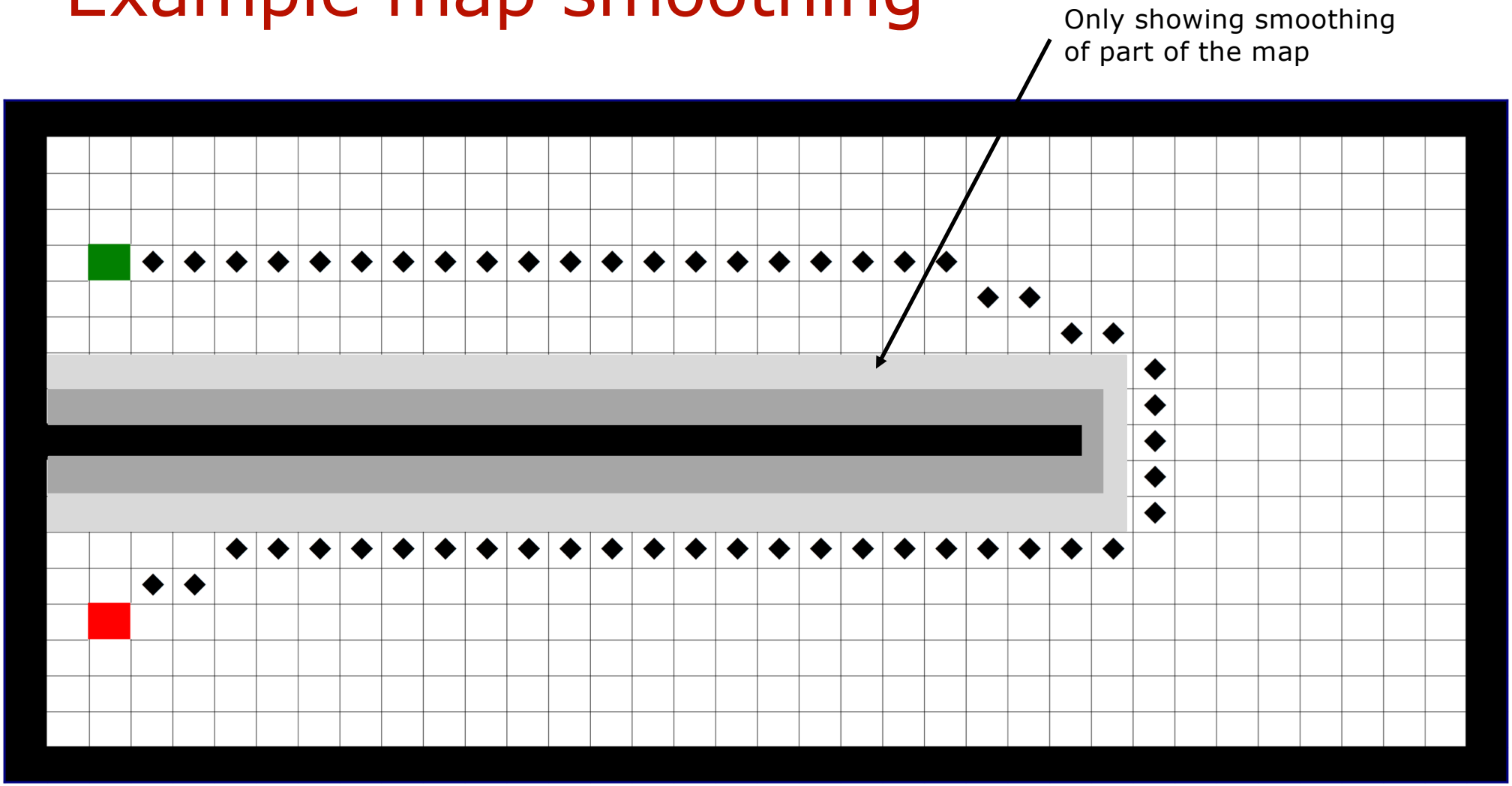
Too close to obstacles

- Smooth the grid with, e.g., a Gaussian kernel
  - Adds costs to be near obstacles (but does not make it impossible to be there
  - → Increases clearance

Example map smoothing



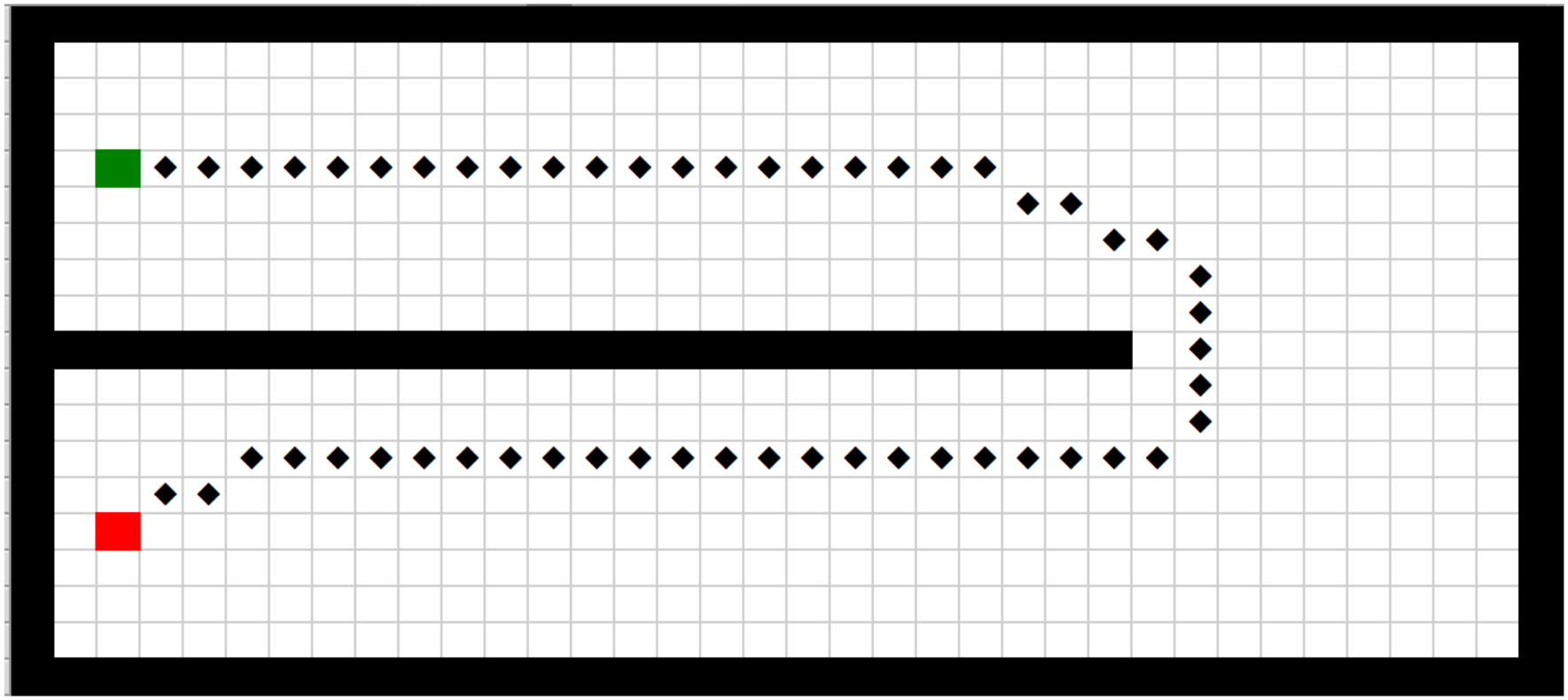
Example map smoothing



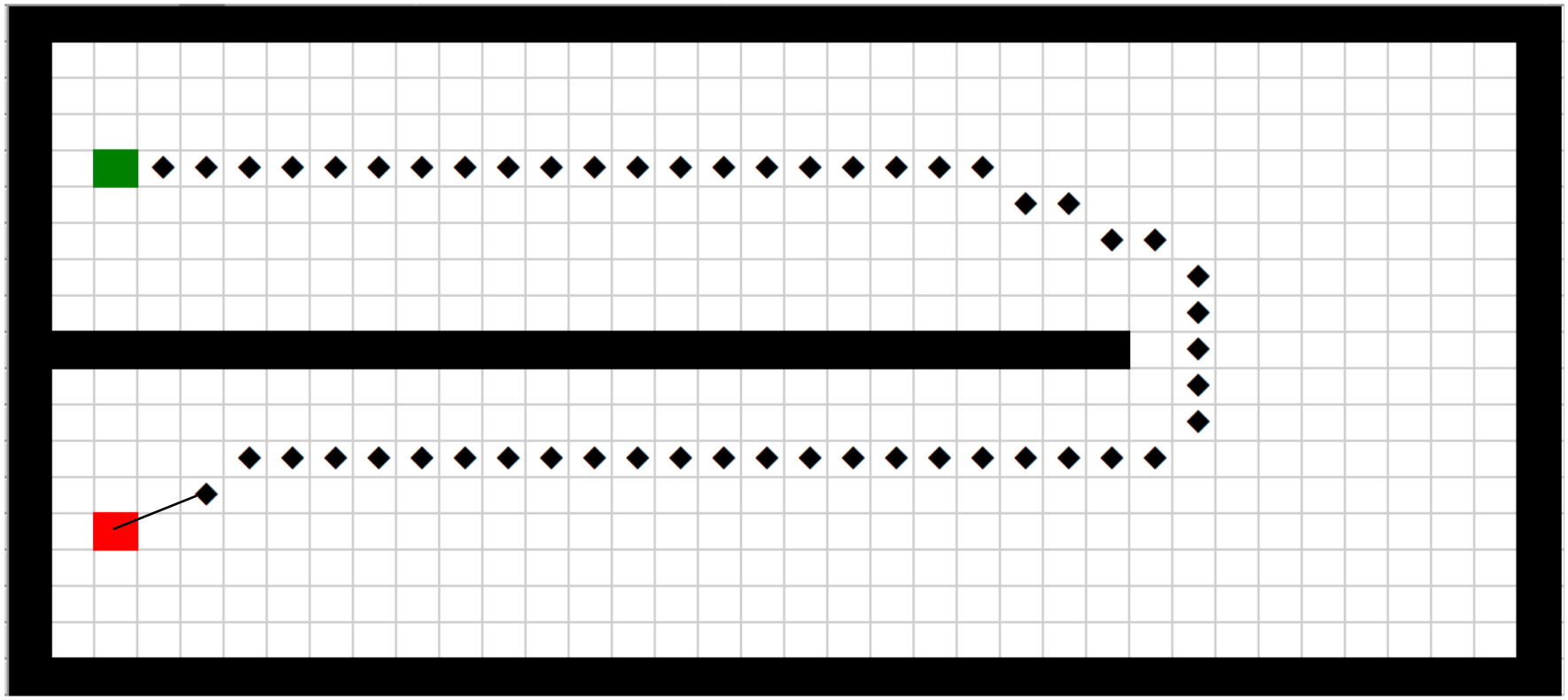
Path smoothing

- Notice how the path is not at all smooth
- Following the path like this would result many short segments with sharp turns in between and would take a lot of time
- Two ideas
  - Look for direct connection between nodes without collision
  - Modify initial path with nonlinear minimization

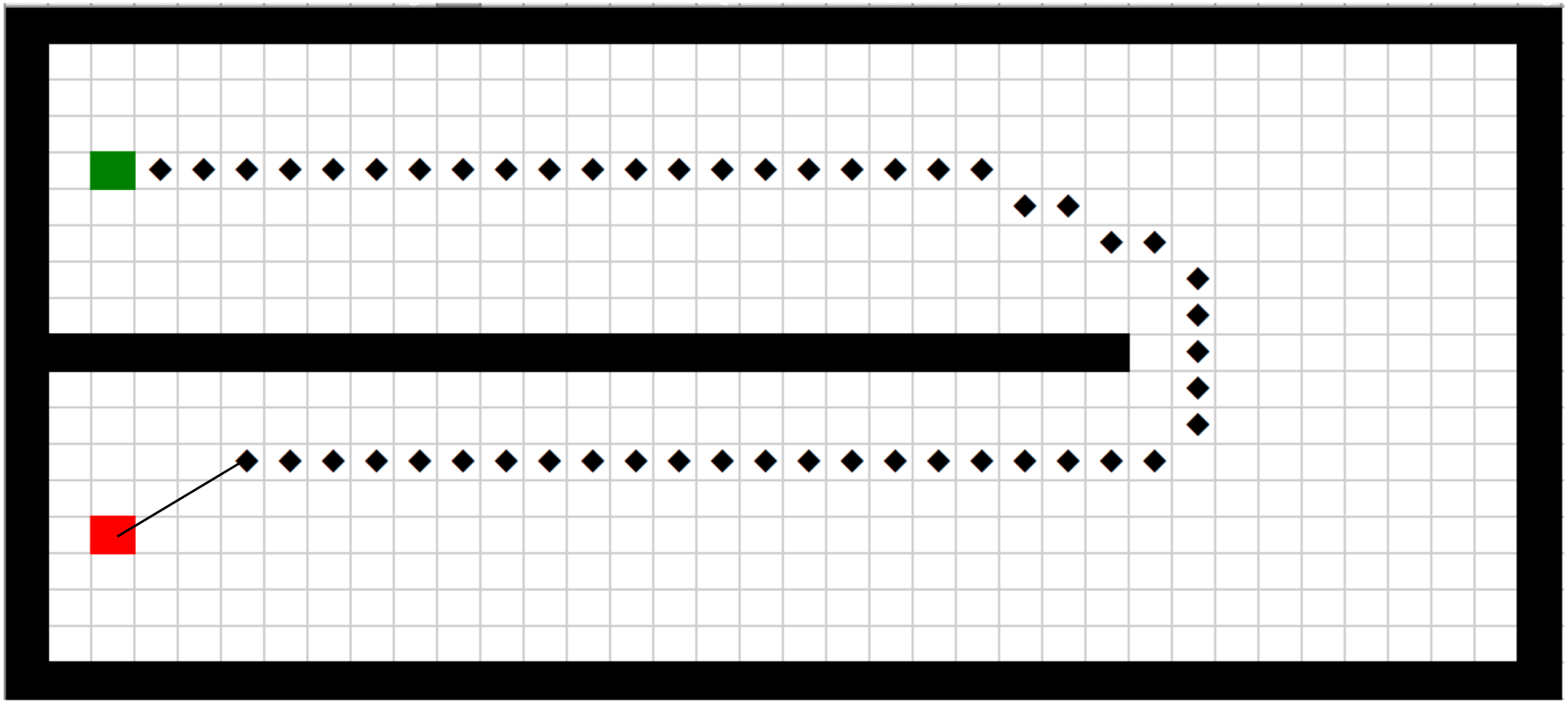
Example path smoothing



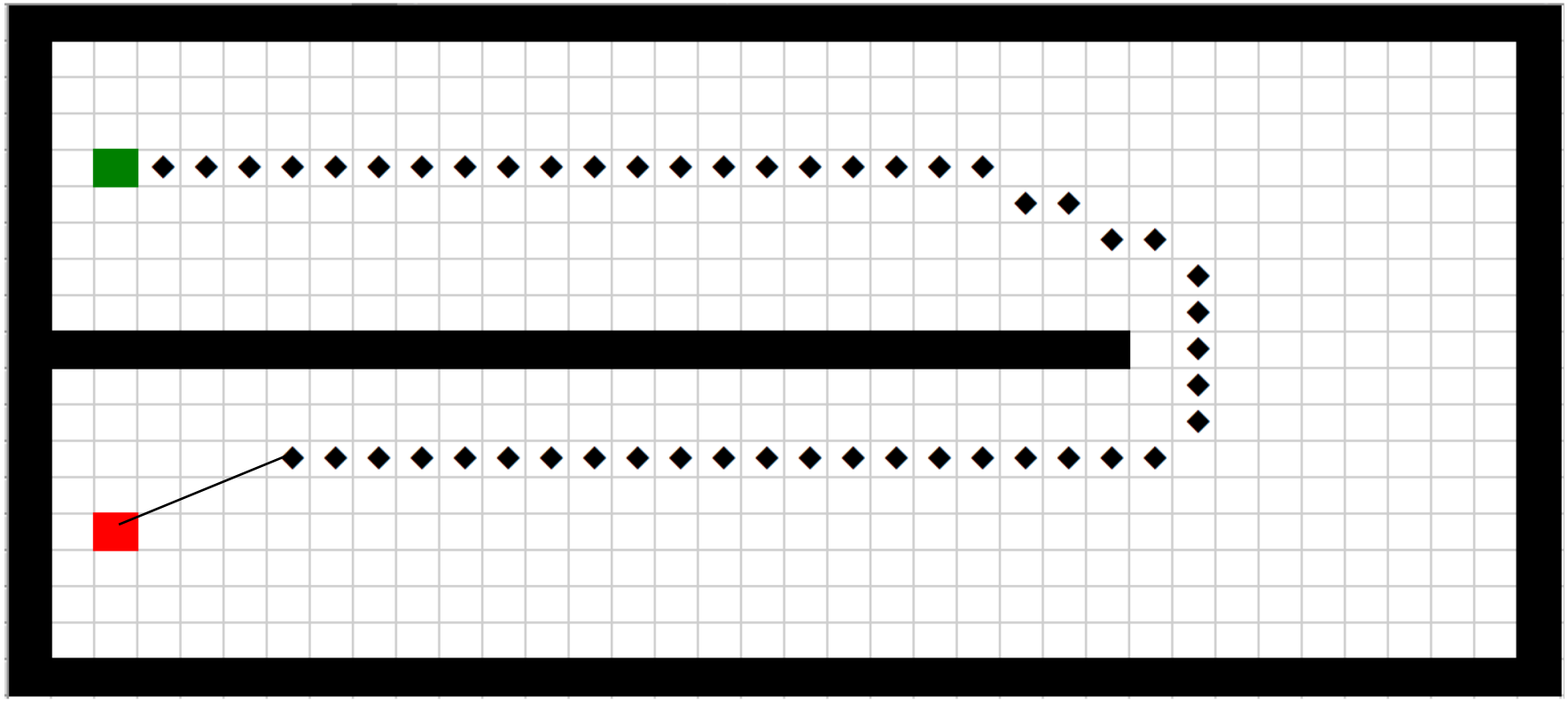
Example path smoothing



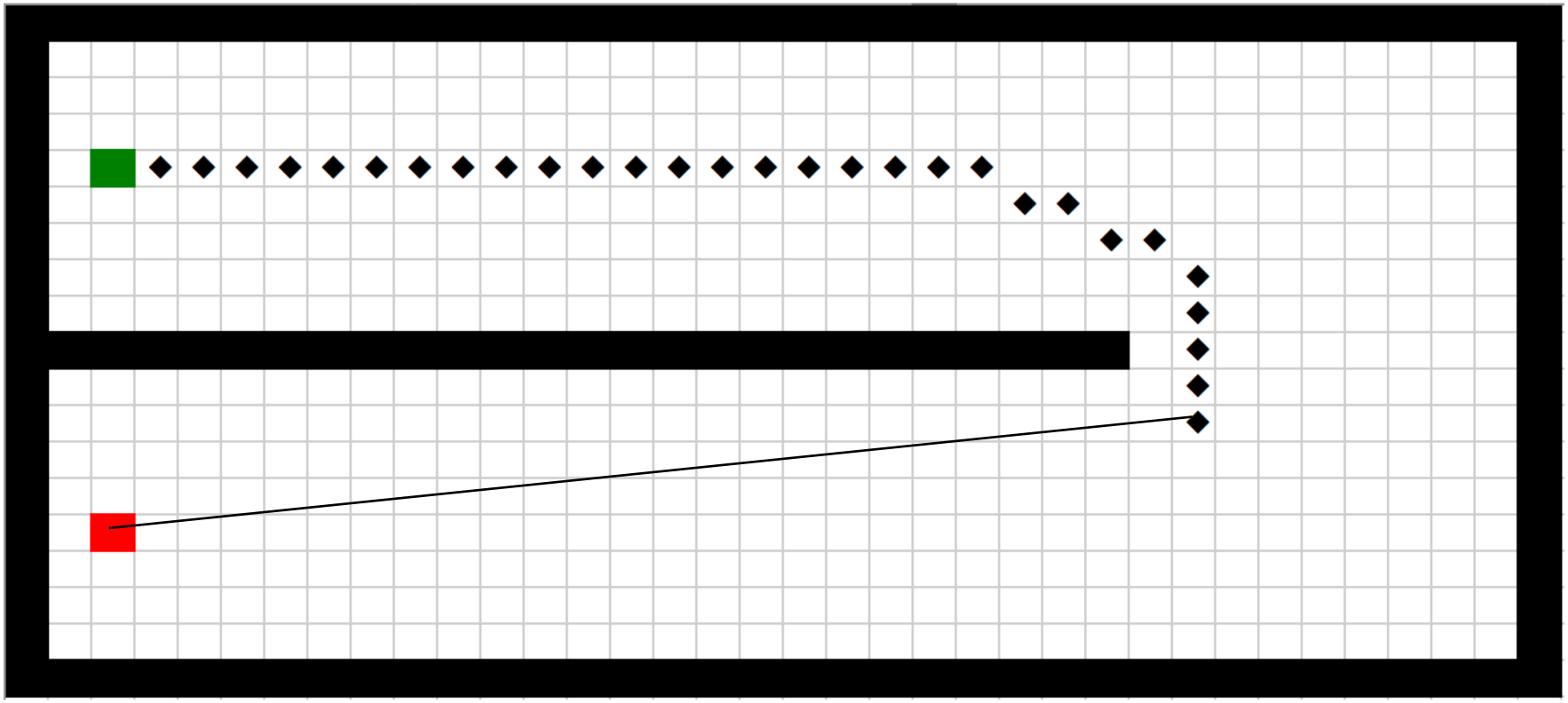
Example path smoothing



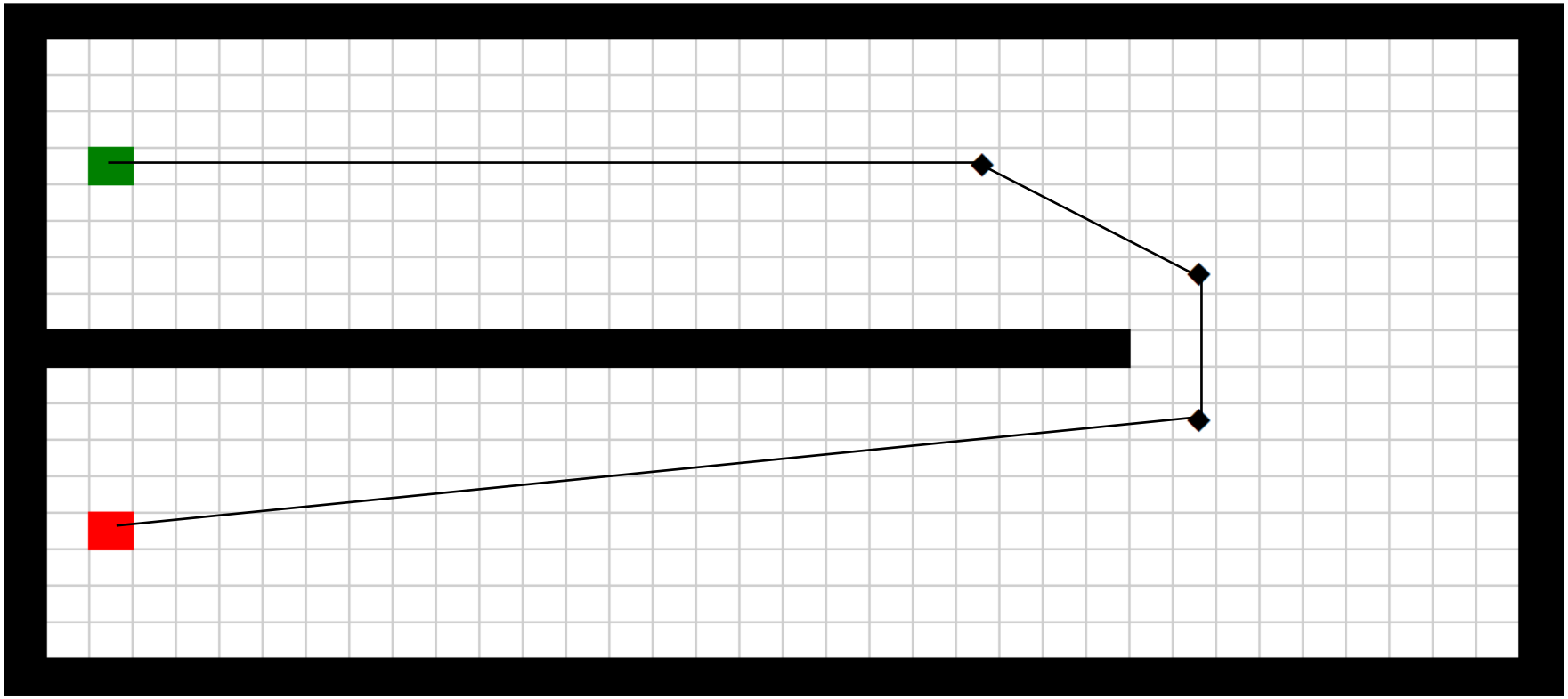
Example path smoothing



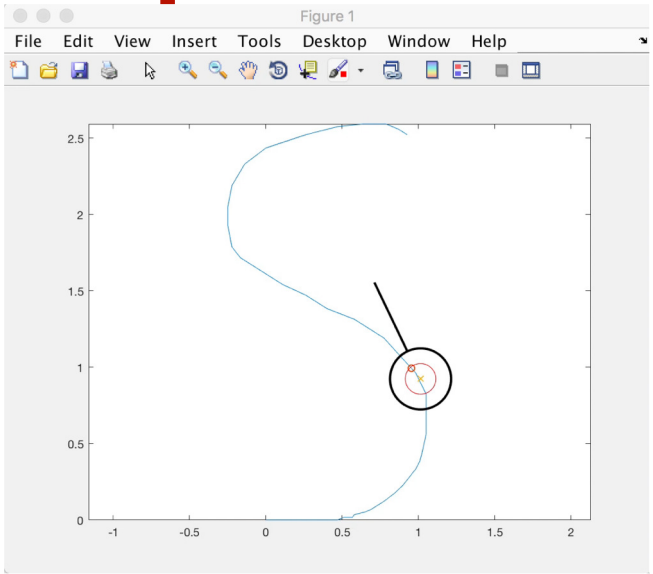
Example path smoothing



Example path smoothing

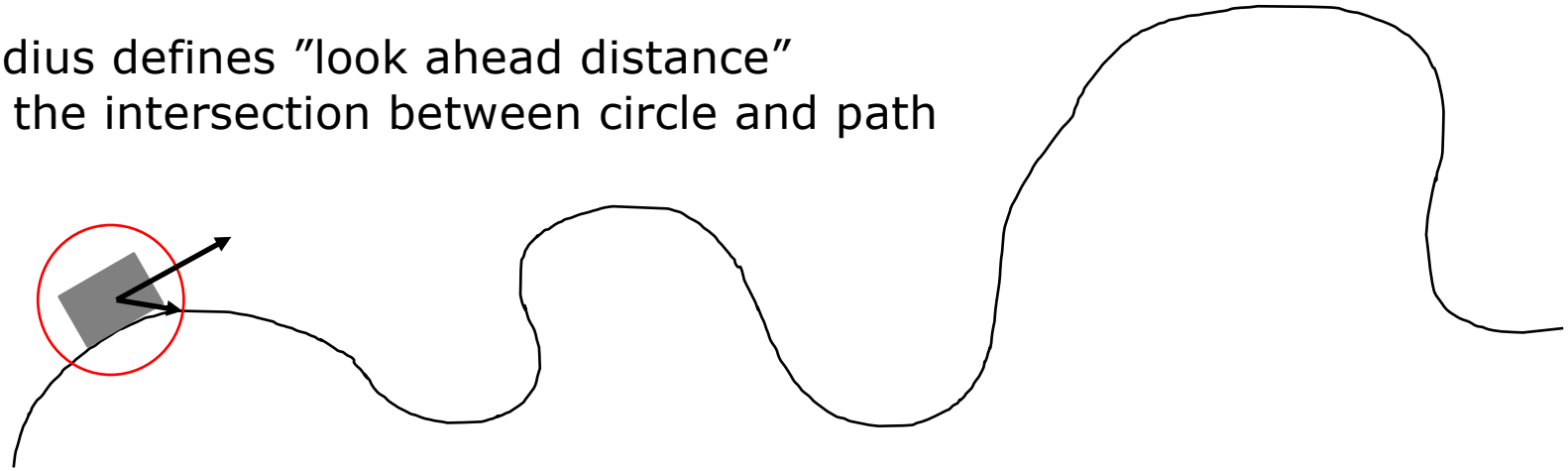


Controlling the robot to follow a path



- Finding the path is only the first step
- Need to follow it
- Rich literature on this topic
- Basic algorithm: Pure pursuit

Circle radius defines ”look ahead distance”
Steer to the intersection between circle and path



Obstacle avoidance

- In general the environment is not fully known or modeled
- The environment might be dynamically changing
  $\rightarrow$ Need to be able modify the plan online
- Many algorithms
  - from re-planning (using planning methods from before) to reactive strategies

Obstacle avoidance

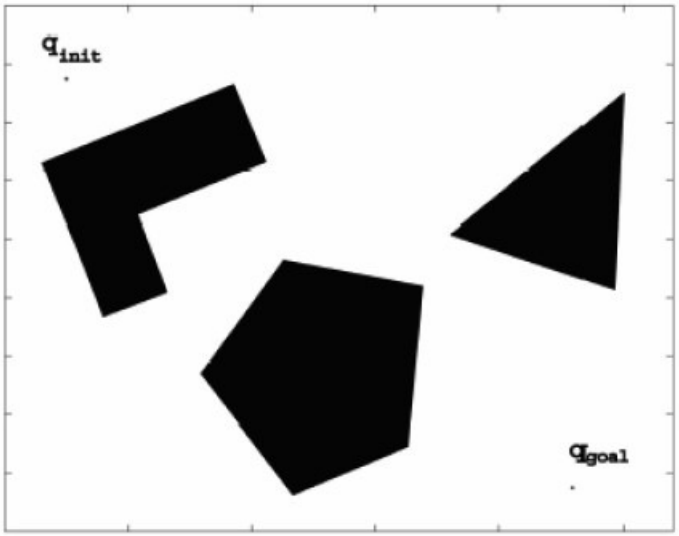
- Obstacle avoidance relies on
  - Information about the goal position
  - Robot position
  - Recent sensory information (typically a local map)
  - NOT the entire map!

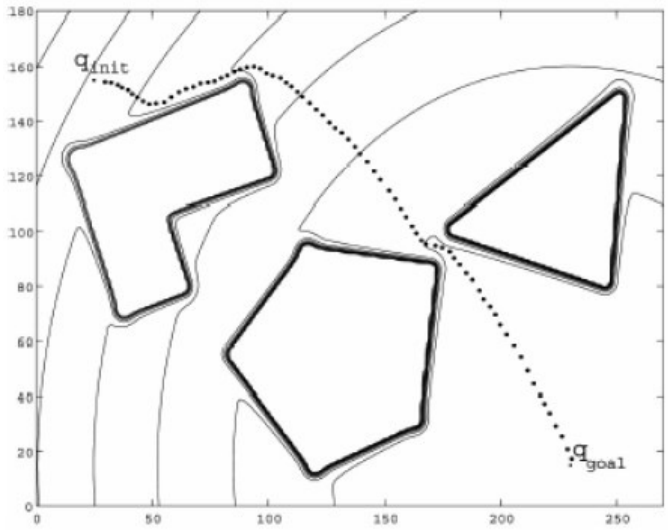
- Why use a local map and not just the last sensor reading which should be the most up to date?

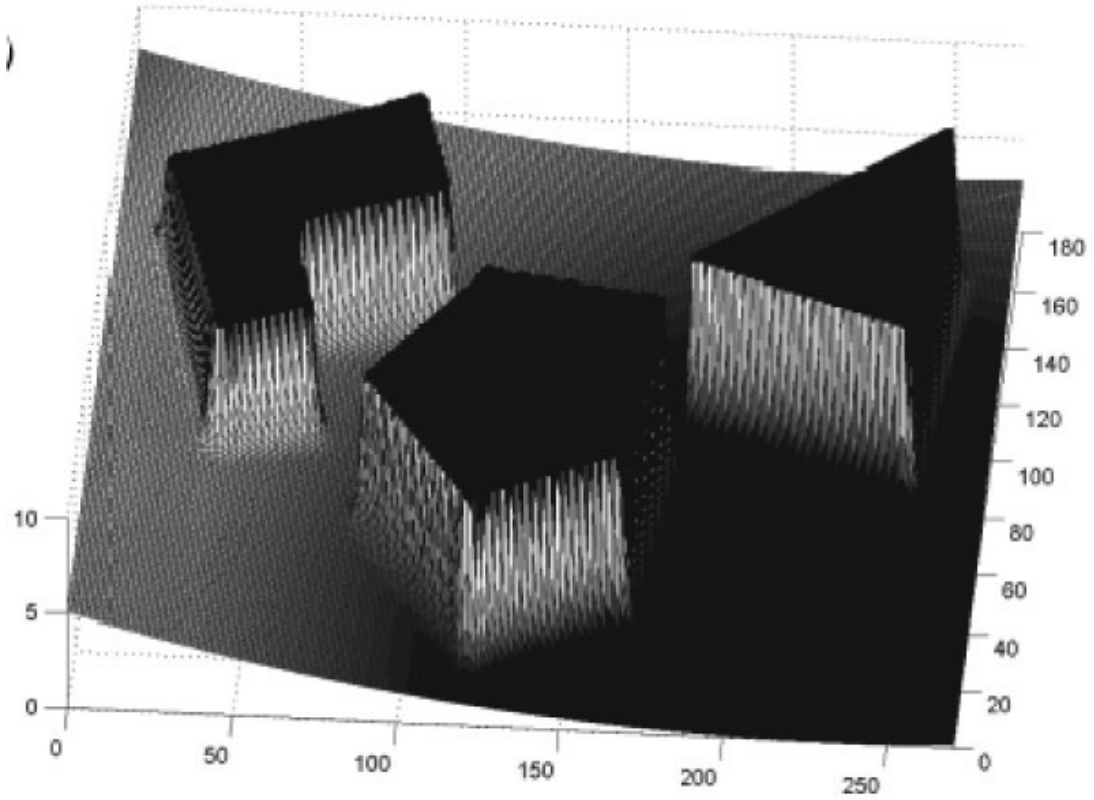
Potential field method

- Think of the robot as a particle in a potential field U(x)
- The goal serves as an attractor
- Obstacles gives repulsive forces
- With a differential potential field the force is given by
- $F(x) = -\text{grad}(U(x))$
- Basic idea: Robot is attracted towards the goal and repulsed by the obstacles

Potential field example

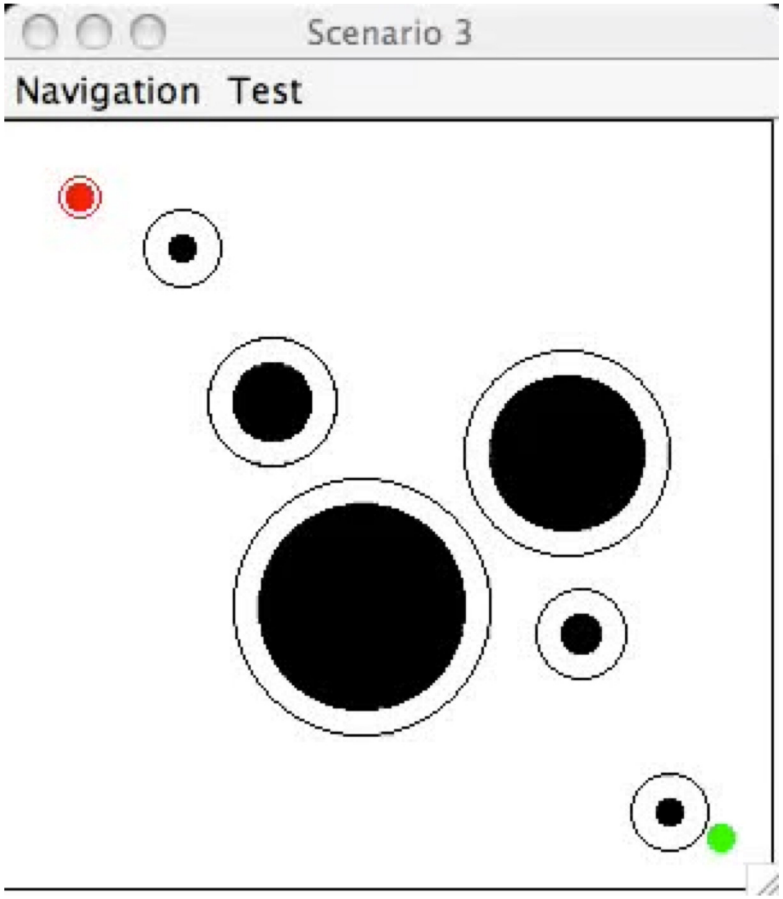


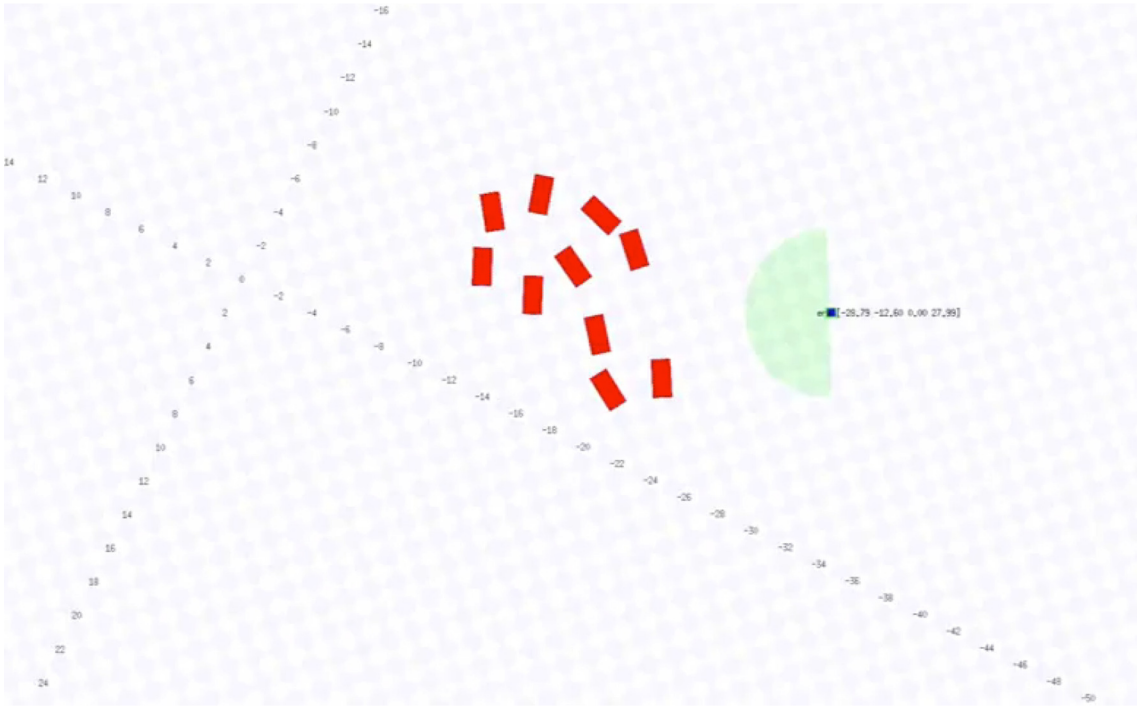




Robot rolls like a ball towards the goal.

Examples





https://www.youtube.com/watch?v=K9nU8xIiIMI

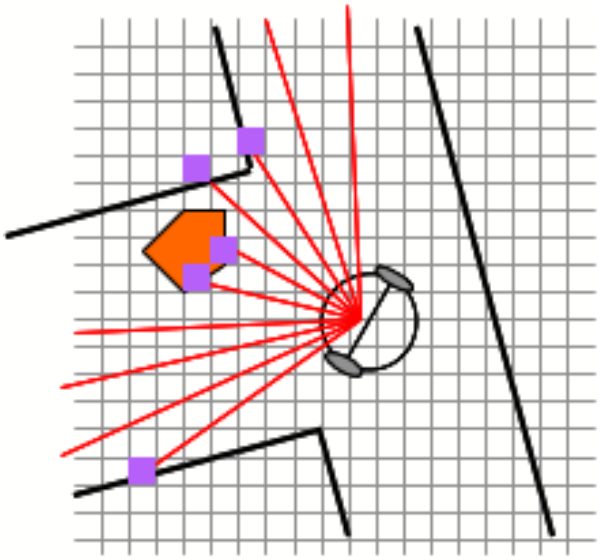
https://www.youtube.com/watch?v=UVUTcZisA94

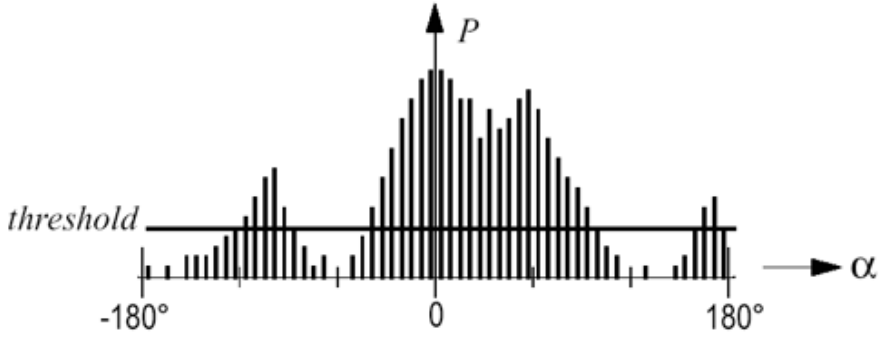
Potential field characteristics

- Pros:
  - Easy to implement
  - Applicable to many problems (arms, multi-robot, etc)
  - Mathematically appeling
- Cons:
  - Motion often not smooth
  - Often leads to oscillating motion
  - Parameter tuning problems
  - Some local minima issues

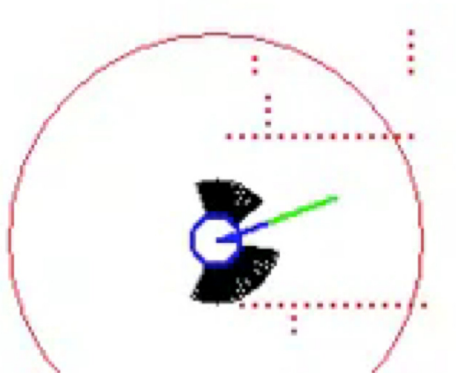
Vector field histogram (VFH)

- Calculate "obstacle forces" similar to potential field
- Generate polar histogram by adding forces in sectors
- Threshold to get binary diagram
- Find passable directions
- Select best direction



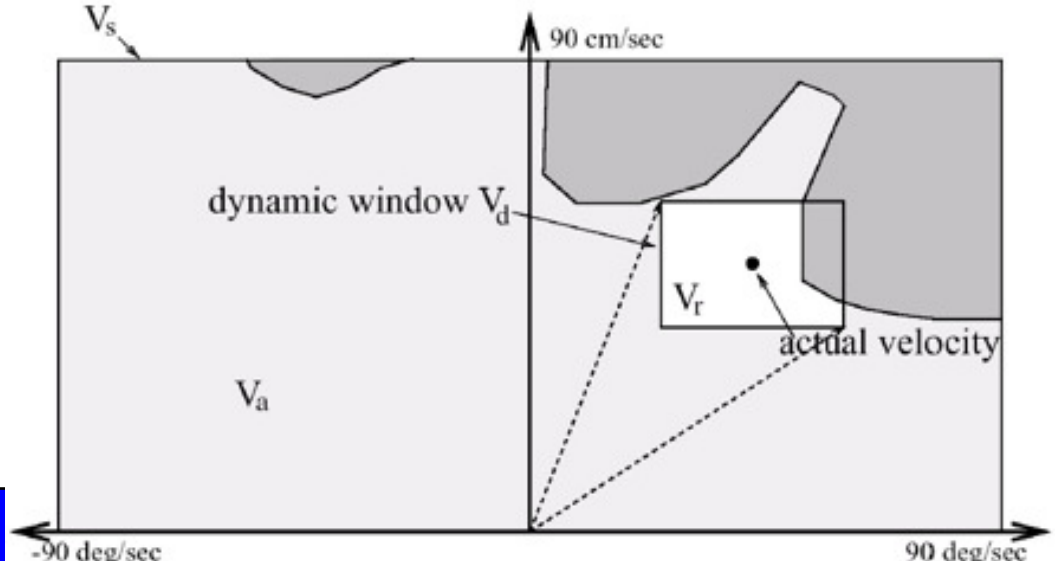


VFH

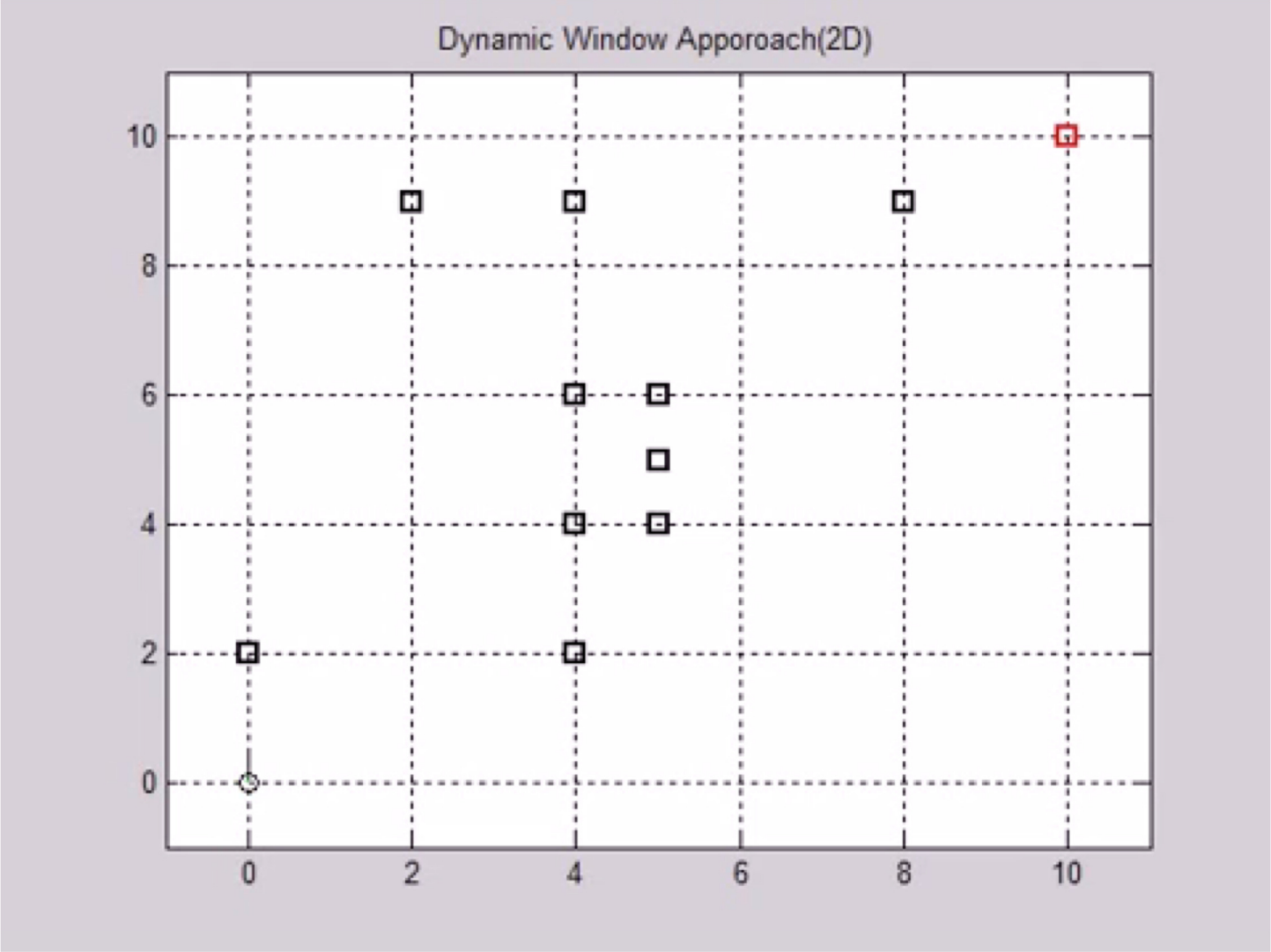


Dynamic window approach (DWA)

- Consider velocity space (v, ω)
- Search among velocities in "reachable" window around current velocity given dynamic constraints
- Optimize over heading, speed and distance to obstacles to ensure fast and safe motion



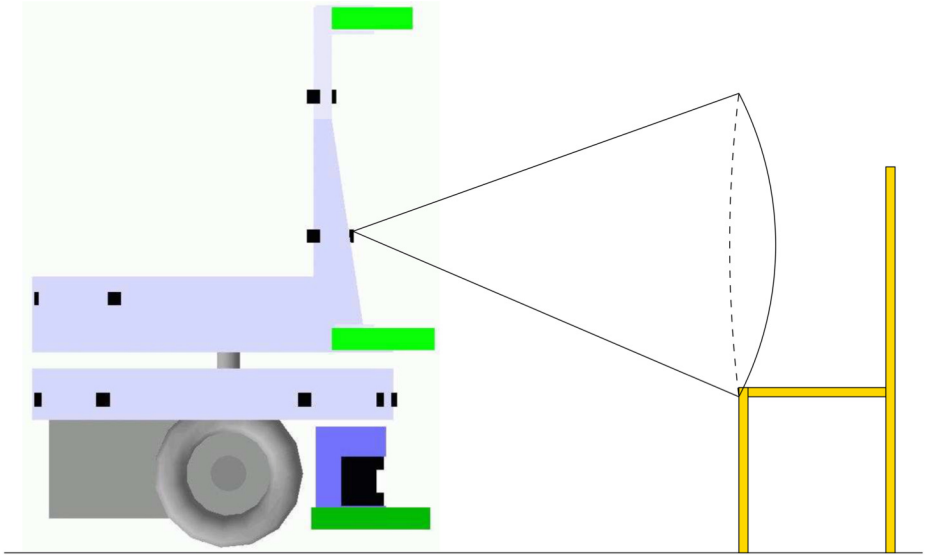
Dynamic window approach (DWA)

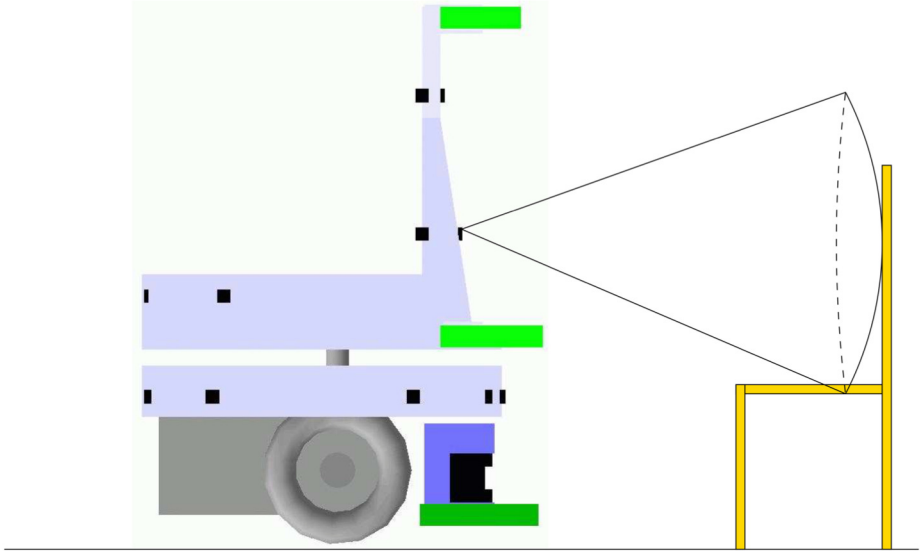


Obstacle avoidance in tele-op

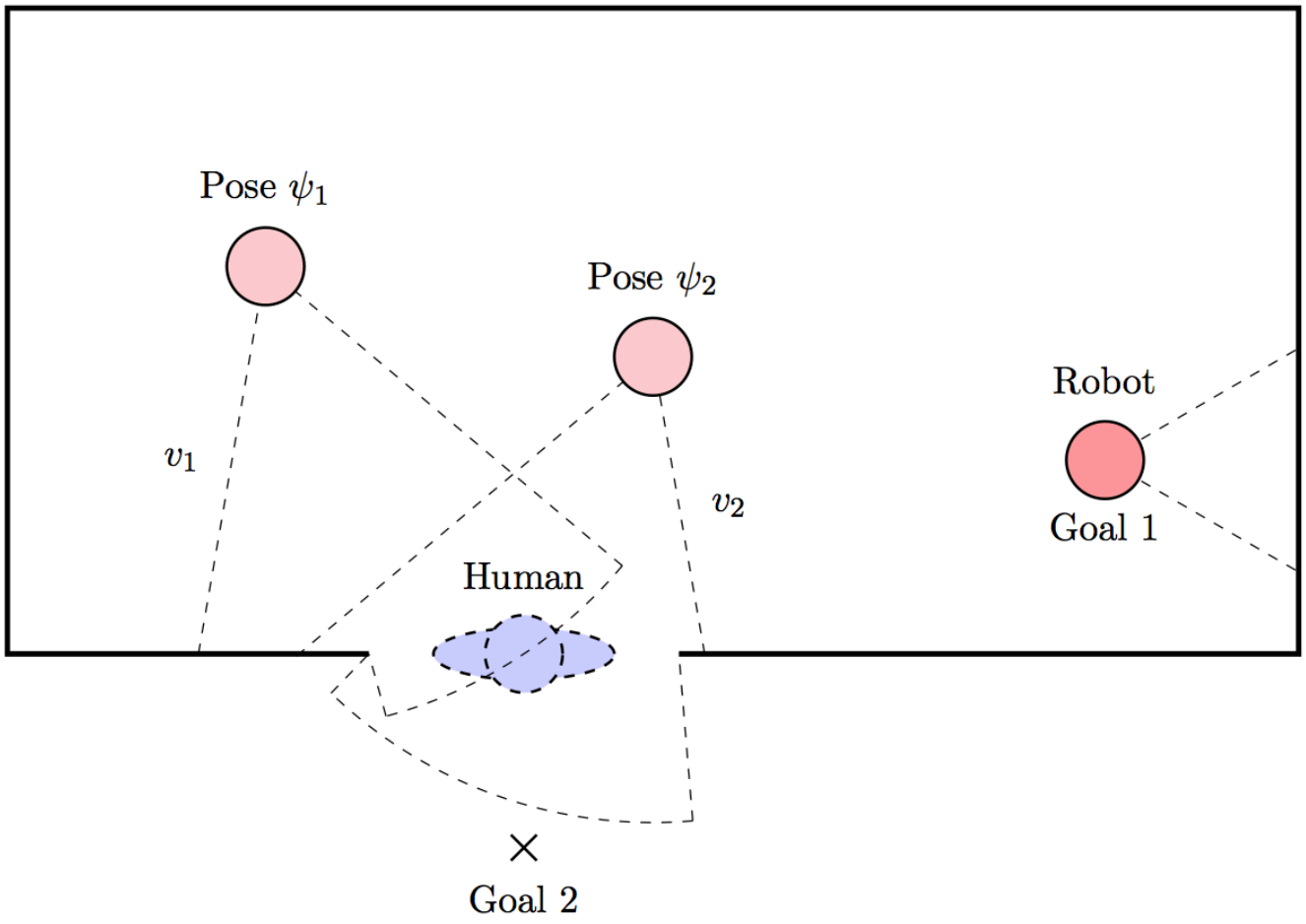
Local map

- Accumulates information from the nearby surrounding of the robot
- Typically an occupancy grid (in 2D or 3D)
- Example below: Without the local map we would probably run into the chair





Local map:
The world is dynamic, deal with it!



The robot is at pose 1 and move to goal 1. It seen the human in the door.

At goal 1 it is given a new target, goal 2. The sensor no longer sees the door opening but the map says that it is closed.

We are stuck!

Image by Lukas Gratwohl

Trade-off in local mapping

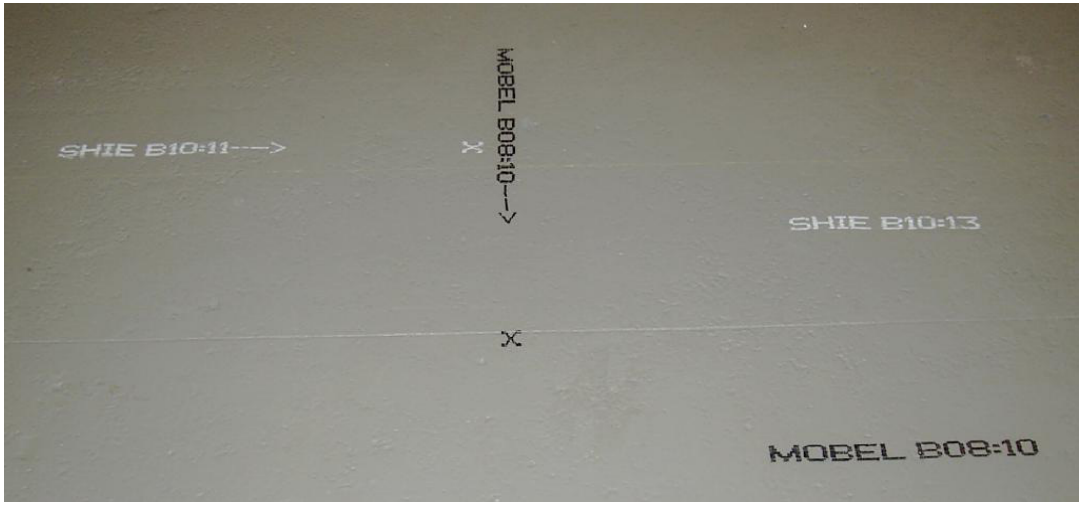
- Trade-off between i) remembering what we no longer see and ii) not being able to get rid of stuff that is not there anymore
  - Type I: the chair (2 slides ago)
  - Type II: an outlier from the sensor causes the robot to get stuck, a dynamic obstacle (human in previous slide)

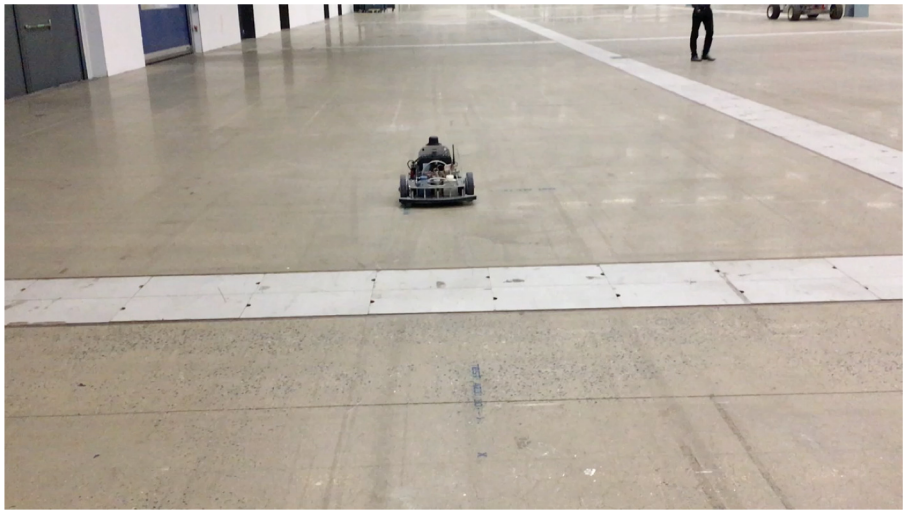
Local mapping

- Need to have some mechanism for obstacles to go away. Some approaches:
  - Let the sensor both add and remove information.
    - Add obstacles at the distance that the sensor registers the obstacle but clear the map in front of that. Good in that you only remove when your sensor tells you it is safe to remove, but you need to look at the area and you might never go there because planning says there is no point.
  - Add forgetting to the map.
    - The obstacle "probability" decreases with time. Good because you do not need to look at the area for problems to "go away", but bad because you might remove correct information and you risk repeating the same mistake over and over again. "Cool, let me try to go over here. Shit that was not possible [now either]"

Case study: Harry Plotter and friends

Obstacle avoidance hard?
Local map?
Main challenge?



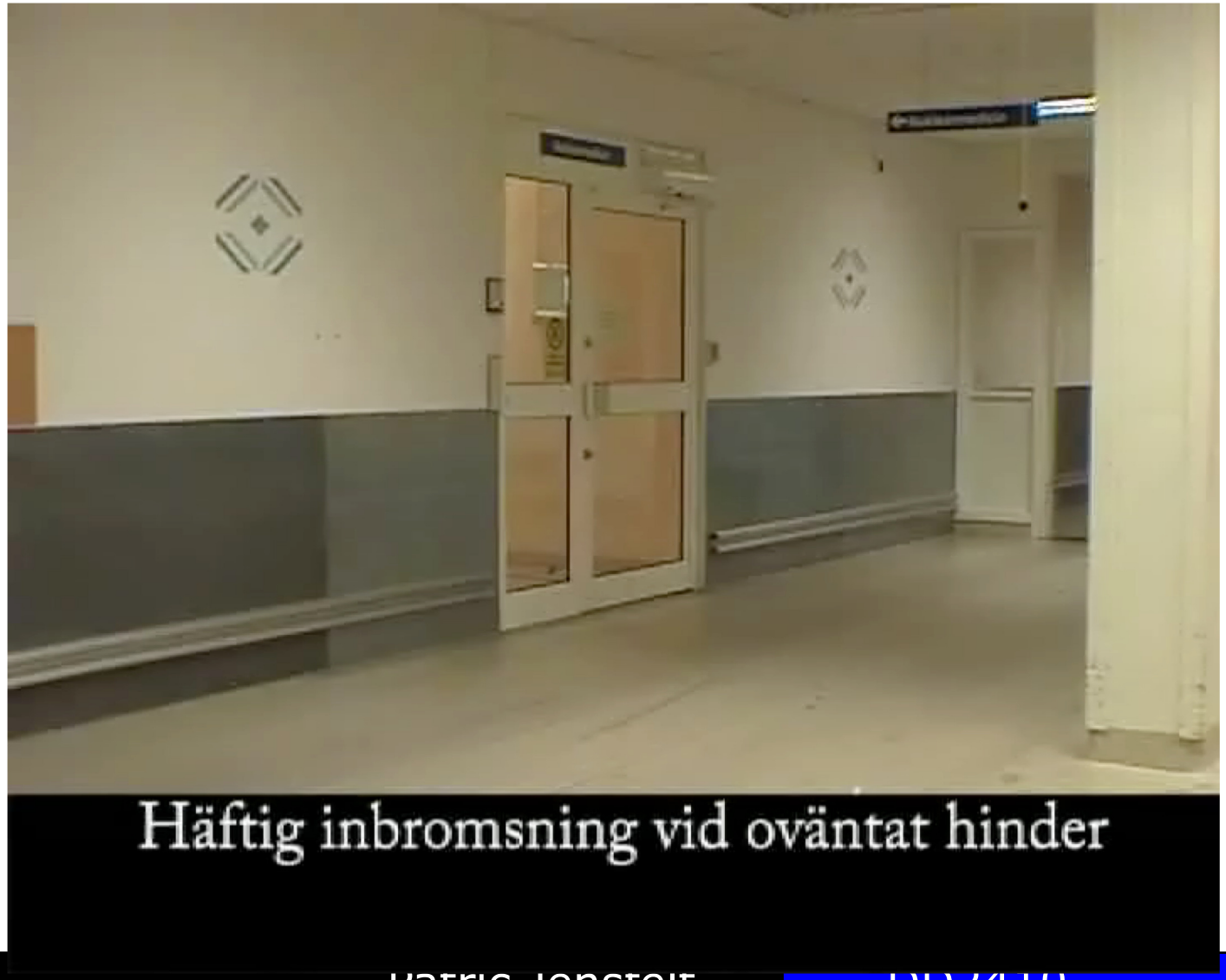


P. Jensfelt, E. Förell and P. Ljunggren,
"Automating the Marking Process for Exhibitions and Fairs",
Robotics and Autonomous Magazine, 14:3, 2007

Stay safe

- Typically need one more layer in the motion control
  - Global motion planning
  - Obstacle avoidance
  - **Emergency stop (reduce damage)**

Example E-Stop



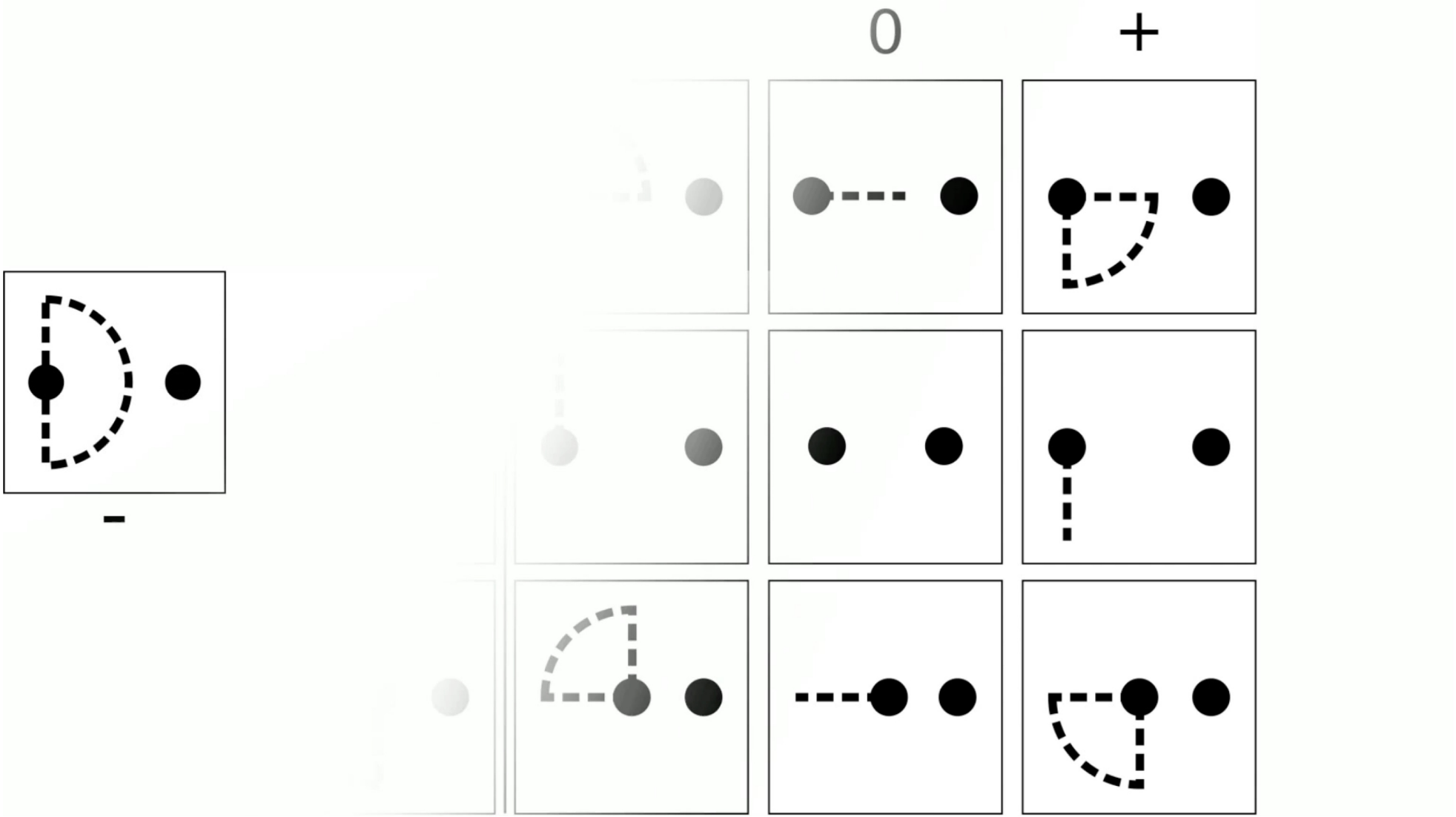
Social navigation

- The motion of the robot = body language
  - Need to communicate safety
- Read body language of other agents
  - Predict motion of people

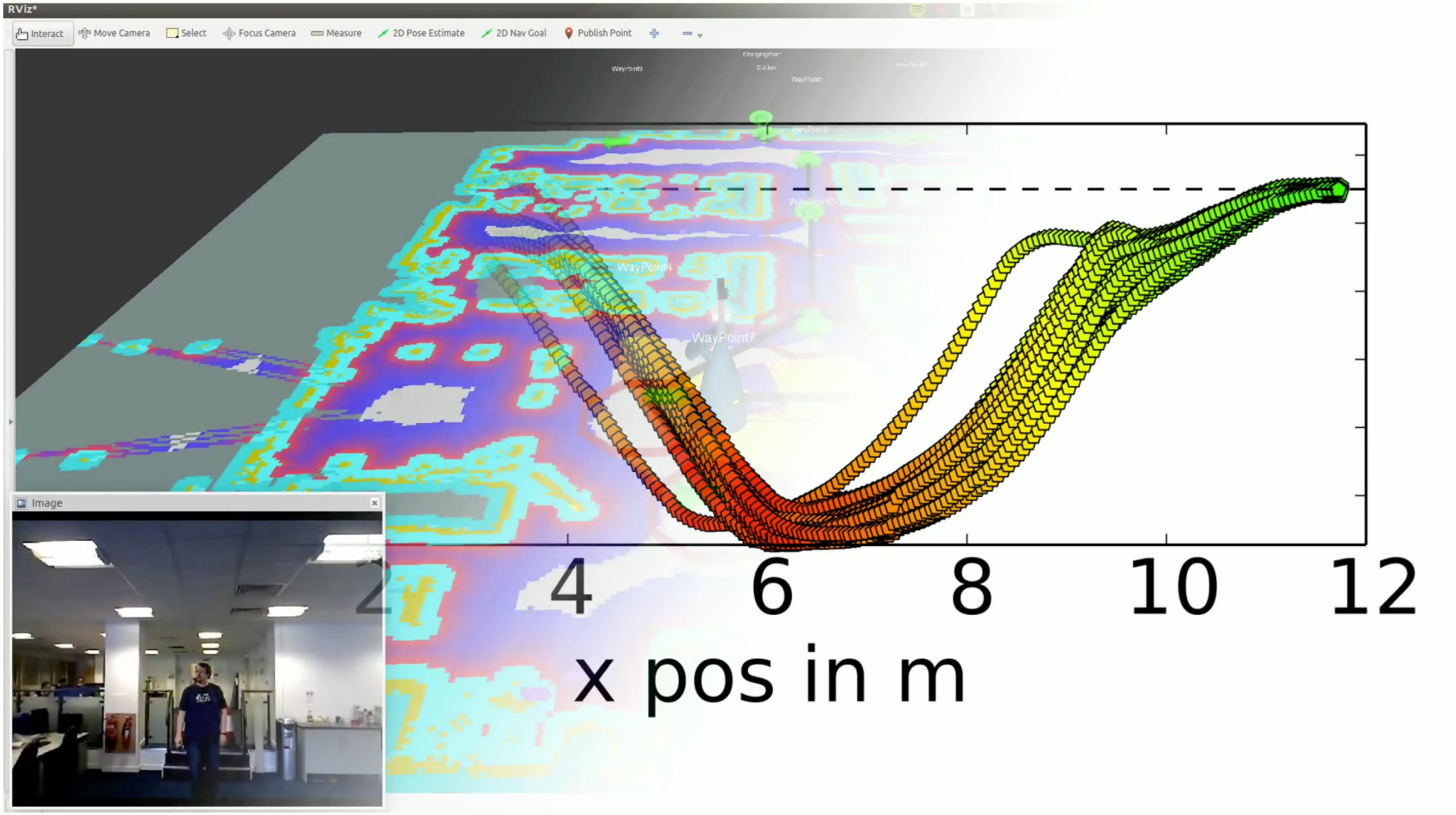
Fast, predictable and safe motion

Example: Social navigation

Example: Social navigation



Example: Social navigation

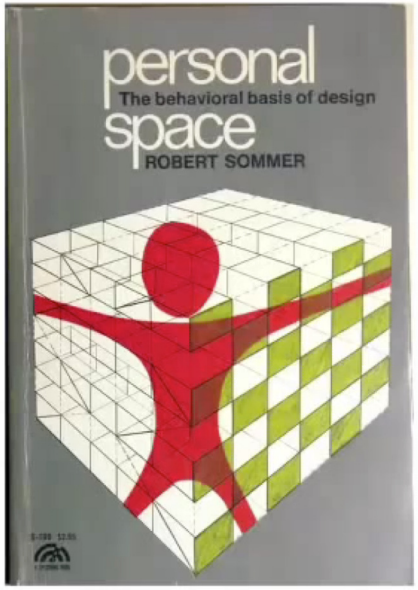


Proxemics

https://www.youtube.com/watch?v=4OFAm-VHATw

Proxemics

Personal Space



"An area with an invisible boundary surrounding a person's body into which intruders may not come"

(Sommer, 1969)

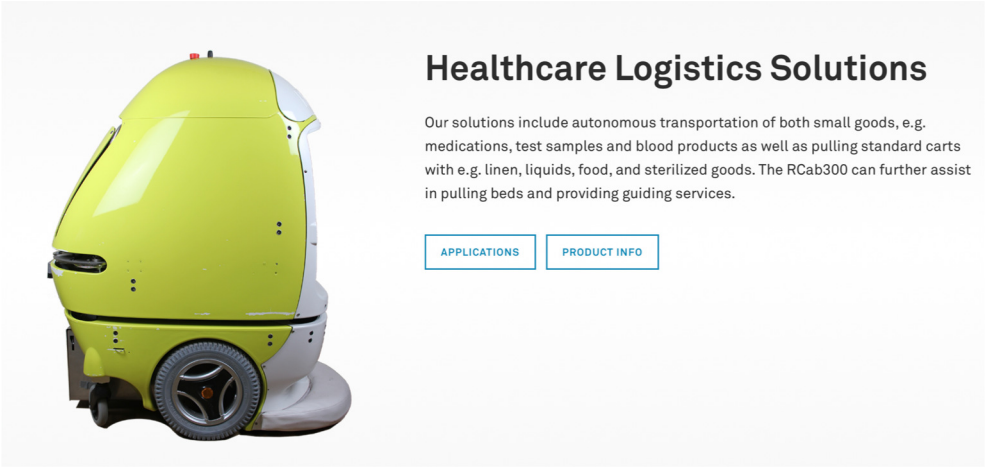
UCIRVINE | University of California

https://www.youtube.com/watch?v=4OFAm-VHATw

System use cases

RCab300

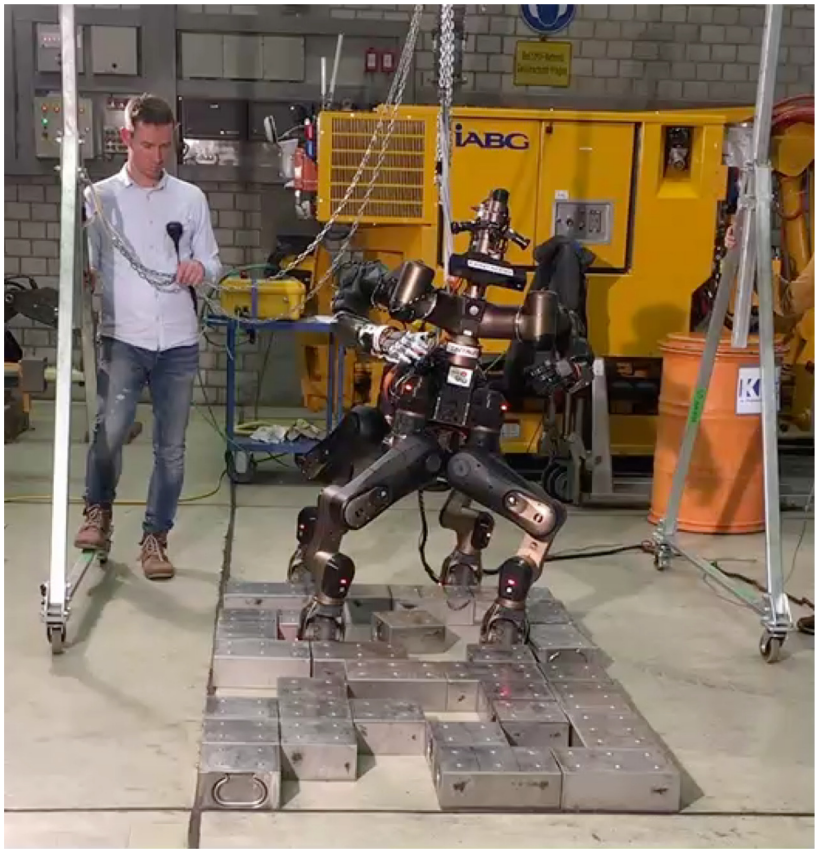
- Deliver goods within a hospital
- Challenges?



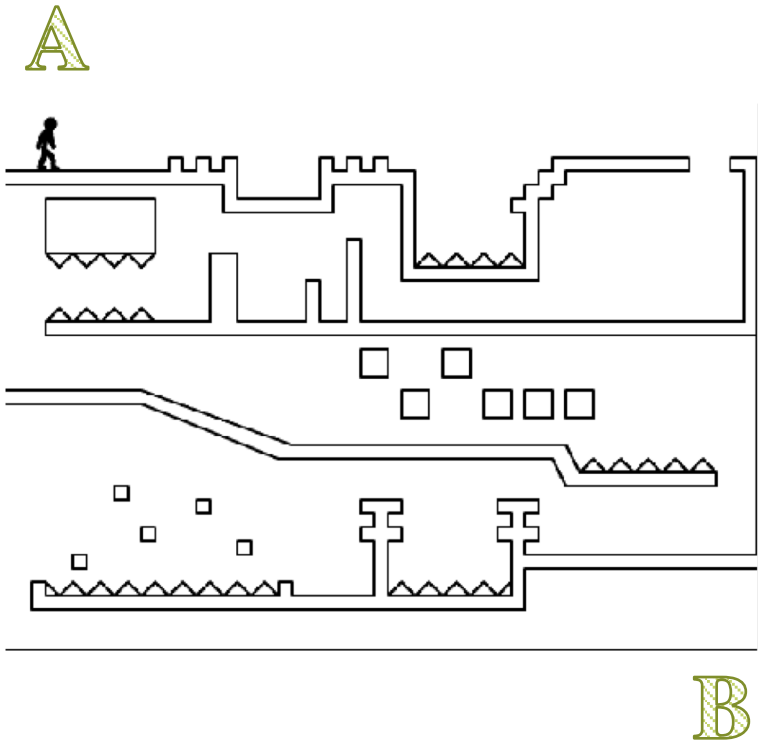
Prototype

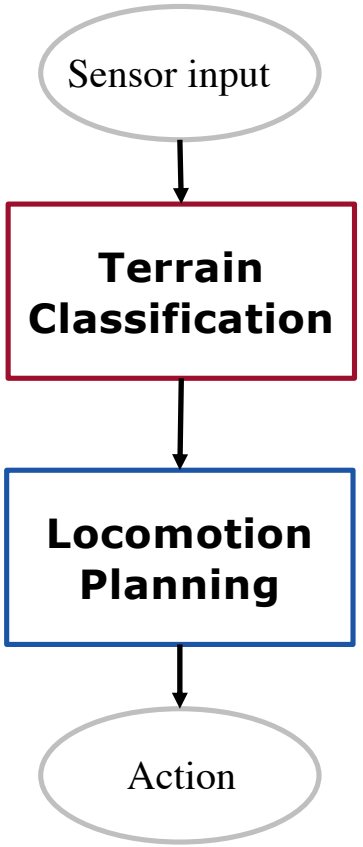
https://unibap.com/product/healthcare-logistics-solutions/

H2020 Project ”CENTAURO”



Going beyond free/occupied





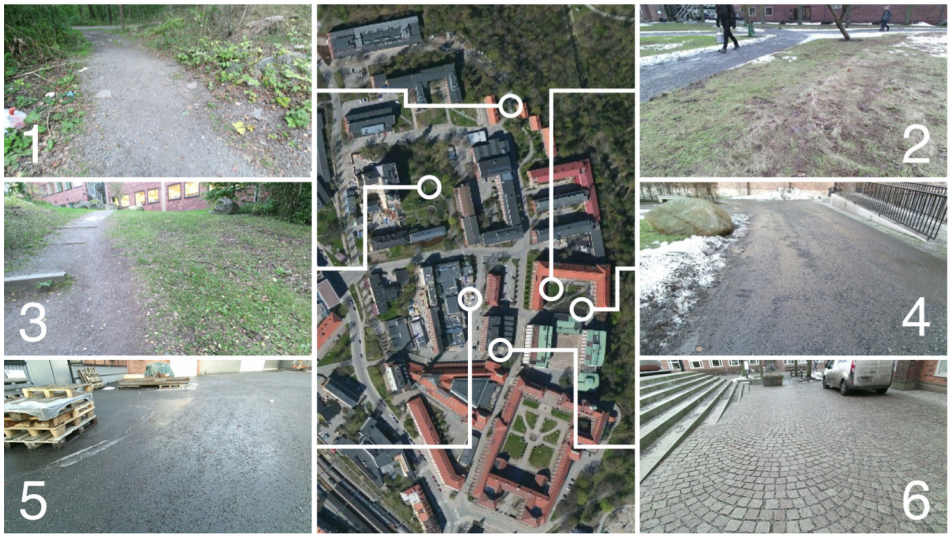

Why terrain classification?


Autonomous navigation!

Our robot and sensor configuration

- Vehicle: iRobot ATRV
  - Four wheels, skid-steering
- Lidar: Velodyne VLP-16
  - 16 rotating laser beams
  - 360° horizontal FOV
  - 30° vertical FOV
- Camera: Microsoft Kinect v2
  - RGB at 540 x 960 resolution

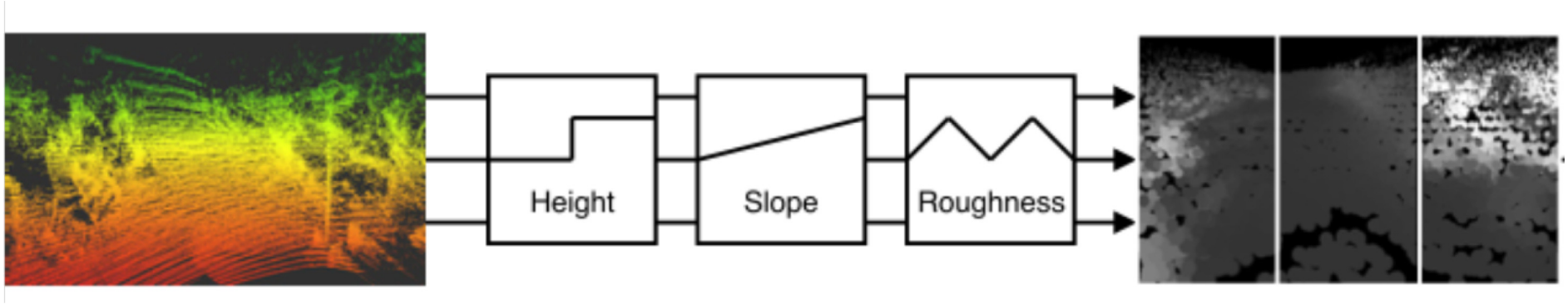
Target environment



- Six unique locations
- Summer and winter conditions
- Urban, semi-structured, and off-road
- Collected point clouds and images
  - 200m trajectory
  - 10 equidistant samples
- Classes:
  - Road, sidewalk, ground
  - Vegetation, grass
  - Vehicles, people, buildings
  - **Also**: snow / ice, stone / rock

Geometric features from registered point clouds

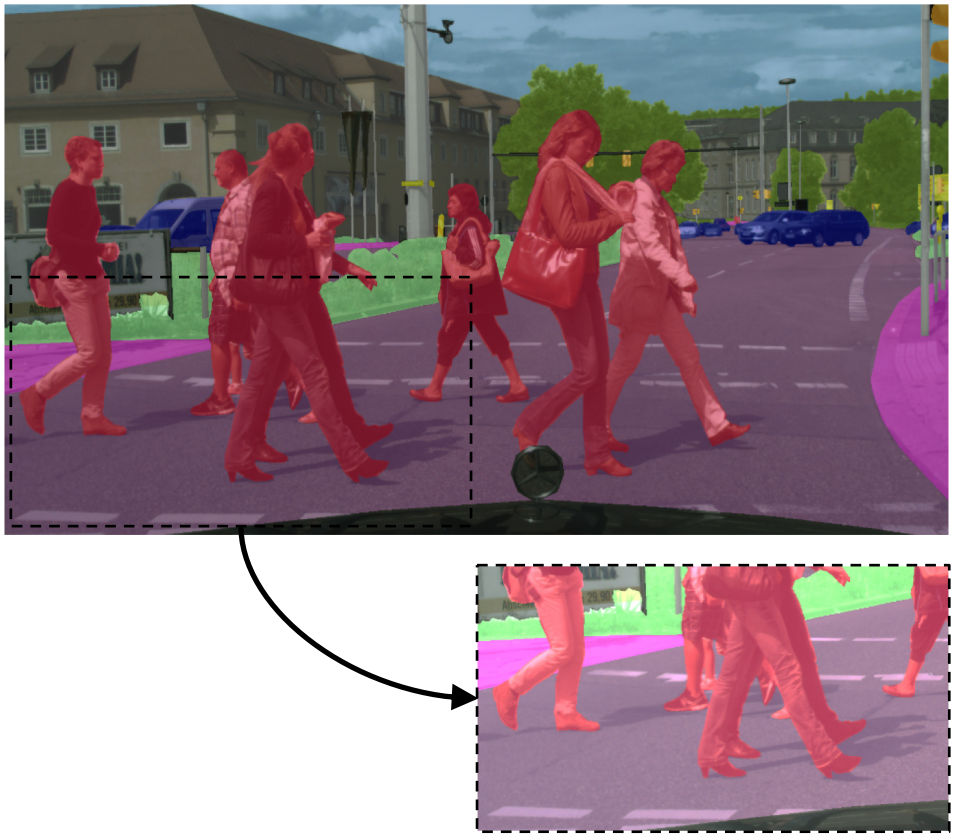
- Three features: height difference, slope, and roughness
- For every terrain patch, compute:
  - **Height difference**: easy
  - **Slope**: fit plane to points in terrain patch and use coefficients
  - **Roughness**: apply Difference of Normals* operator at multiple radii



The baseline model suffers from urban biases

- Trained on CityScapes* dataset
  - Geared towards autonomous navigation
  - Has almost all classes we need (34 total)
  - ~3,000 training, 500 validation images
- Severe **road** bias
  - Also hood bias in earlier model!
- Oblivious to **snow** class

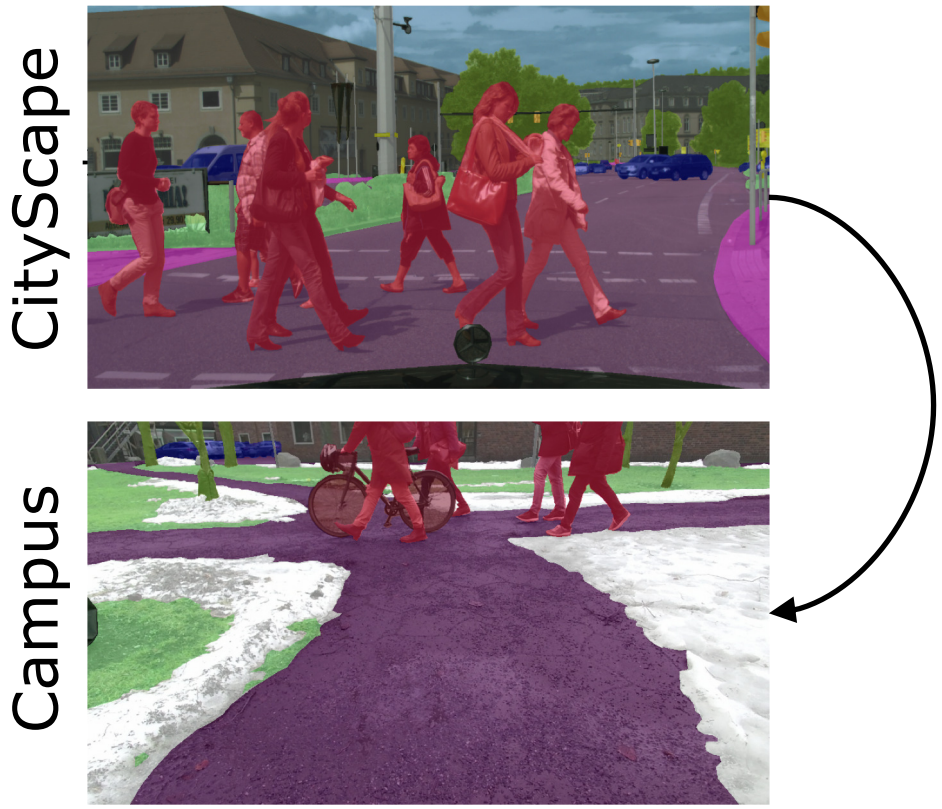
Data augmentation strategy to overcome biases



(exaggerated adjustment)

- Re-train on CityScapes only
- Downsample to 512 x 1024 (factor 2)
- Crop 256 x 512 patch from lower image
  - Not interested in upper regions, i.e. building, sky, etc.
- Random adjustments of...
  - Brightness
  - Contrast
  - Saturation

Fine-tuning strategy for environmental adaptation

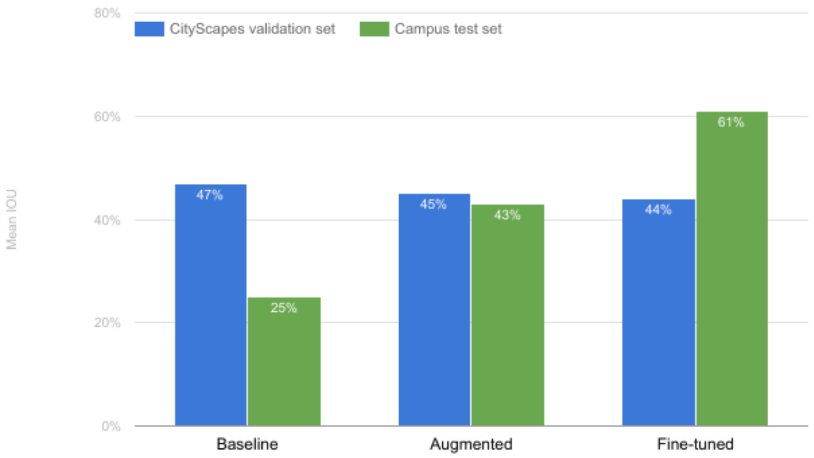


- Train on CityScapes and campus mixed
  - Source to target dataset ratio 9:1
- Repurpose unused classes
  - *caravan* for *snow*
  - *trailer* for *stone*
- Keep augmentation scheme
- Reduce learning rate by factor 10

Data augmentation and fine-tuning alleviates biases

- Learned to recognize **snow**
- All other biases gone
- Problems with motion **blur**

Quantitative results for visual segmentation



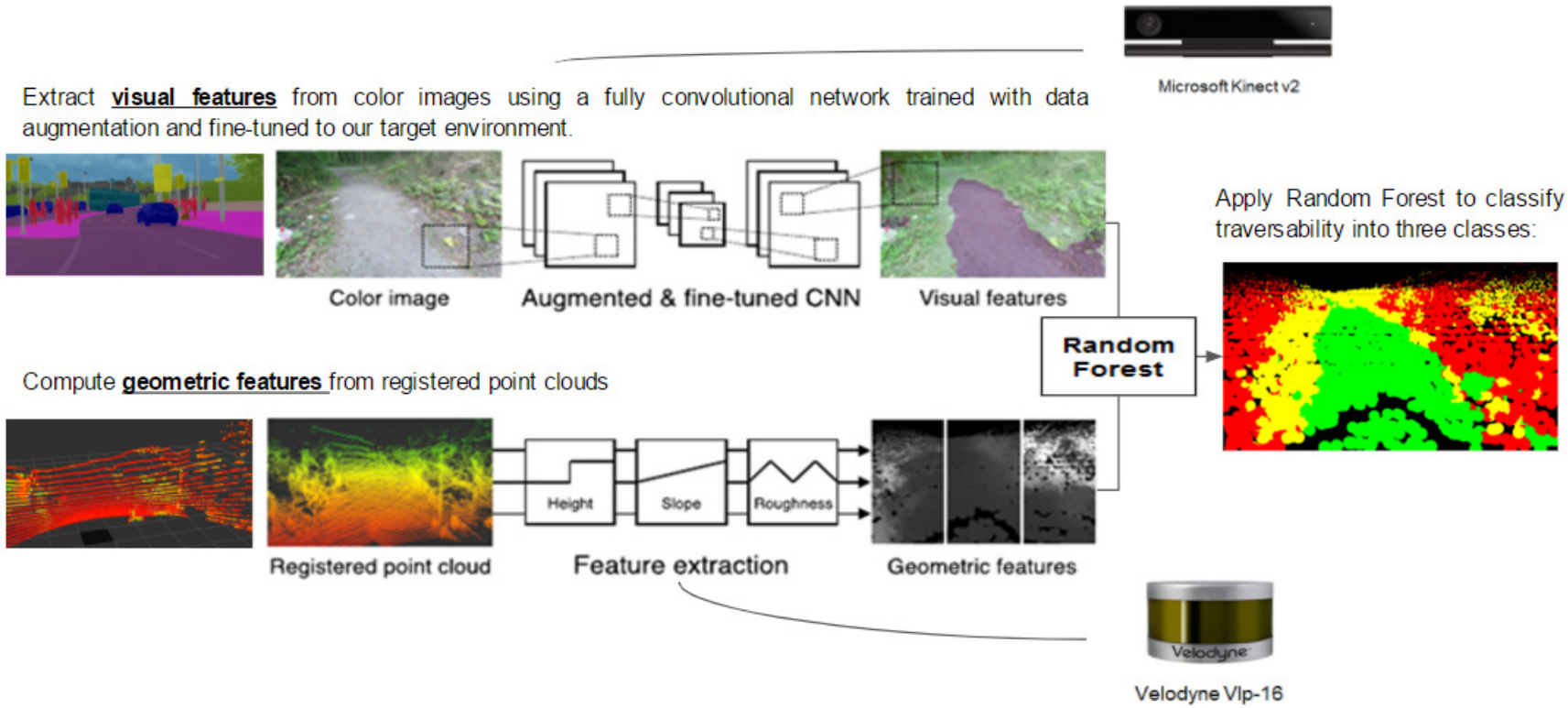
- Performance jumps!

- Baseline model
- Worst performance
- Urban biases!

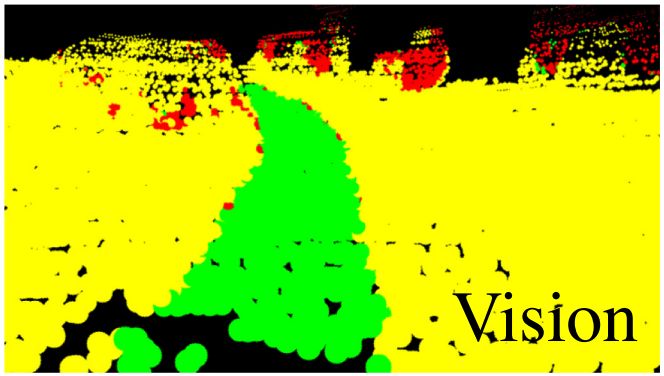
- Augmented model
- On par with CityScapes performance
- Oblivious to snow

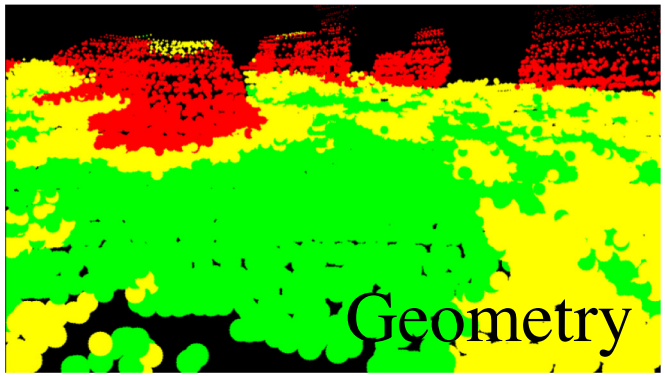
- Fine-tuned model
- Tops CityScapes performance
- Suitable for autonomous navigation!

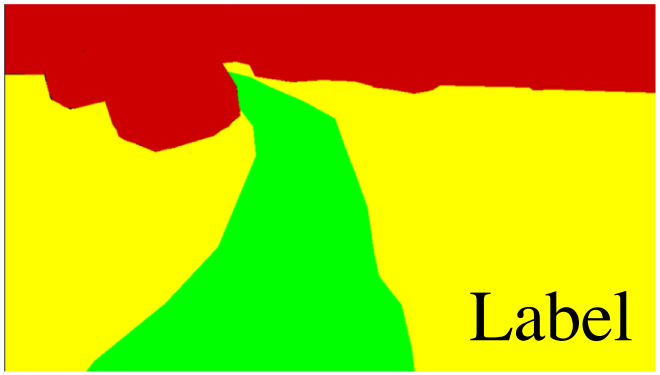
Architecture

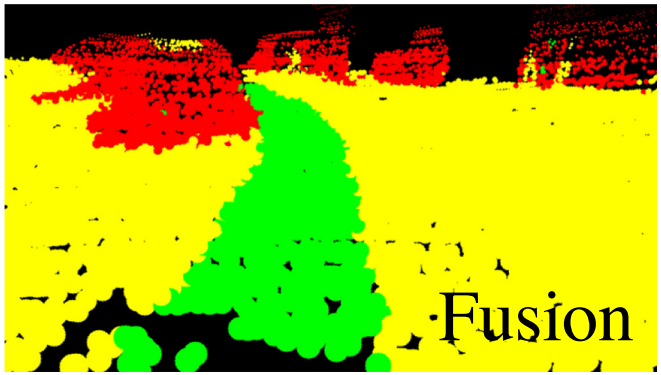


Qualitative results for vision, geometry, and fusion









The terrain classification system in action

Road marking

- Challenges?

Project "STRANDS"

- Goal: Long-term autonomy (100 days)
- Challenges?