Negative Information and Line Observations for Monte Carlo Localization

Todd Hester Peter Stone

Learning Agents Research Group
Department of Computer Sciences
The University of Texas at Austin

IEEE International Conference on Robotics and Automation, 2008

The Problem

Mobile Robot Localization

Maintain estimate of global position and orientation over time

- Given map of fixed landmark locations
- Not SLAM

The Problem

Mobile Robot Localization

Maintain estimate of global position and orientation over time

- Given map of fixed landmark locations
- Not SLAM

Challenging Platform

Typical Platform

- Wheeled robot
- Range-finding sensors

- Sony Aibo ERS-7
  - Color CMOS Camera in nose
    - Narrow field-of-view (56^o)
    - 30 YCrCb frames per second
  - Quadruped
  - 576MHz processor
    - All on-board processing

Challenging Platform

Our Platform

- Legged robot
- Vision-based sensors

- Sony Aibo ERS-7
  - Color CMOS Camera in nose
    - Narrow field-of-view (56^o)
    - 30 YCrCb frames per second
  - Quadruped
  - 576MHz processor
    - All on-board processing

Goal

Desiderata

- Navigate to specific point quickly
- Remain localized while colliding
- Recover quickly from kidnappings

Approach

- Begin with baseline MCL algorithm
- Add Negative Information
- Add Line Observations

Significant improvement over baseline

Goal

Desiderata

- Navigate to specific point quickly
- Remain localized while colliding
- Recover quickly from kidnappings

Approach

- Begin with baseline MCL algorithm
- Add Negative Information
- Add Line Observations

Significant improvement over baseline

Goal

Desiderata

- Navigate to specific point quickly
- Remain localized while colliding
- Recover quickly from kidnappings

Approach

- Begin with baseline MCL algorithm
- Add Negative Information
- Add Line Observations

Significant improvement over baseline

Outline

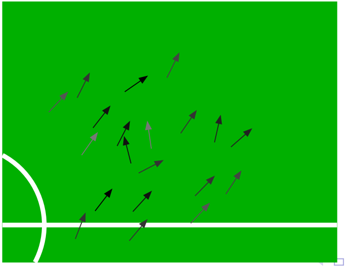
Outline

Method: Particle Filtering

- Estimate $p(h_T | o_T, a_{T-1}, o_{T-1}, a_{T-2}, \ldots, a_0)$: Distribution of poses given observations and actions
- Represented by finite set of samples: particles
  - Each is a hypothesis: $\langle \langle x, y, \theta \rangle, p \rangle$
- Weighted average to get single estimate of pose and confidence

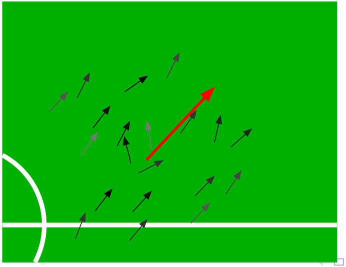
Method: Particle Filtering

- Estimate $p(h_T | o_T, a_{T-1}, o_{T-1}, a_{T-2}, \ldots, a_0)$:
  Distribution of poses given observations and actions
- Represented by finite set of samples: particles
  - Each is a hypothesis: $\langle\langle x, y, \theta\rangle, p\rangle$
- Weighted average to get single estimate of pose and confidence



Method: Particle Filtering

- Estimate $p(h_T | o_T, a_{T-1}, o_{T-1}, a_{T-2}, \ldots, a_0)$: Distribution of poses given observations and actions
- Represented by finite set of samples: particles
  - Each is a hypothesis: $\langle\langle x, y, \theta\rangle, p\rangle$
- Weighted average to get single estimate of pose and confidence

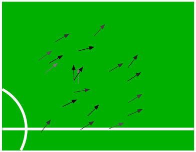


Observation Update

- Need sensor model: $p(o|h)$
  - Predicts observations given pose hypothesis using map
- Update each particle when robot sees something
  - Compute similarity for each observed landmark in frame
    - Use angles and distances to landmarks
    - Difference in measured and expected values
  - Compute product of similarities
  - Adjust probability closer to new value

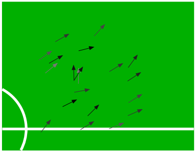
Observation Update

- Need sensor model: $p(o|h)$
  - Predicts observations given pose hypothesis using map
- Update each particle when robot sees something
  - Compute similarity for each observed landmark in frame
    - Use angles and distances to landmarks
    - Difference in measured and expected values
  - Compute product of similarities
  - Adjust probability closer to new value



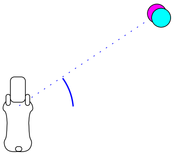
Observation Update

- Need sensor model: $p(o|h)$
  - Predicts observations given pose hypothesis using map
- Update each particle when robot sees something
  - Compute similarity for each observed landmark in frame
    - Use angles and distances to landmarks
    - Difference in measured and expected values
  - Compute product of similarities
  - Adjust probability closer to new value



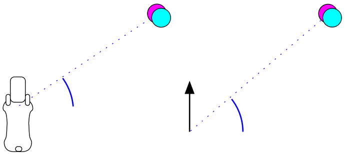
Observation Update

- Need sensor model: $p(o|h)$
  - Predicts observations given pose hypothesis using map
- Update each particle when robot sees something
  - Compute similarity for each observed landmark in frame
    - Use angles and distances to landmarks
    - Difference in measured and expected values
  - Compute product of similarities
  - Adjust probability closer to new value



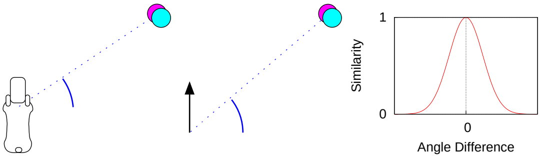
Observation Update

- Need sensor model: $p(o|h)$
  - Predicts observations given pose hypothesis using map
- Update each particle when robot sees something
  - Compute similarity for each observed landmark in frame
    - Use angles and distances to landmarks
    - Difference in measured and expected values
  - Compute product of similarities
  - Adjust probability closer to new value



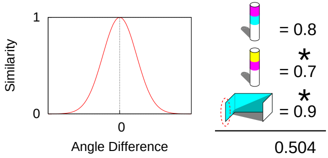
Observation Update

- Need sensor model: $p(o|h)$
  - Predicts observations given pose hypothesis using map
- Update each particle when robot sees something
  - Compute similarity for each observed landmark in frame
    - Use angles and distances to landmarks
    - Difference in measured and expected values
  - Compute product of similarities
  - Adjust probability closer to new value



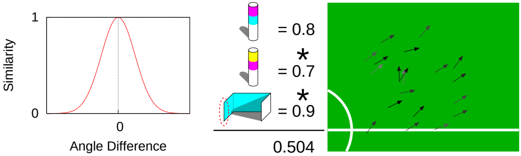
Observation Update

- Need sensor model: $p(o|h)$
  - Predicts observations given pose hypothesis using map
- Update each particle when robot sees something
  - Compute similarity for each observed landmark in frame
    - Use angles and distances to landmarks
    - Difference in measured and expected values
  - Compute product of similarities
  - Adjust probability closer to new value



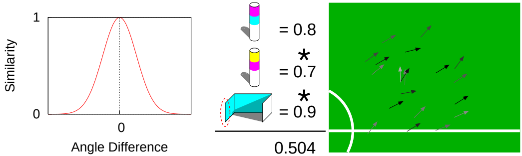
Observation Update

- Need sensor model: $p(o|h)$
  - Predicts observations given pose hypothesis using map
- Update each particle when robot sees something
  - Compute similarity for each observed landmark in frame
    - Use angles and distances to landmarks
    - Difference in measured and expected values
  - Compute product of similarities
  - Adjust probability closer to new value



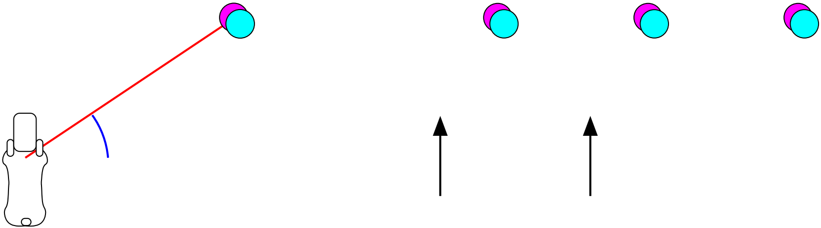
Observation Update

- Need sensor model: $p(o|h)$
  - Predicts observations given pose hypothesis using map
- Update each particle when robot sees something
  - Compute similarity for each observed landmark in frame
    - Use angles and distances to landmarks
    - Difference in measured and expected values
  - Compute product of similarities
  - Adjust probability closer to new value



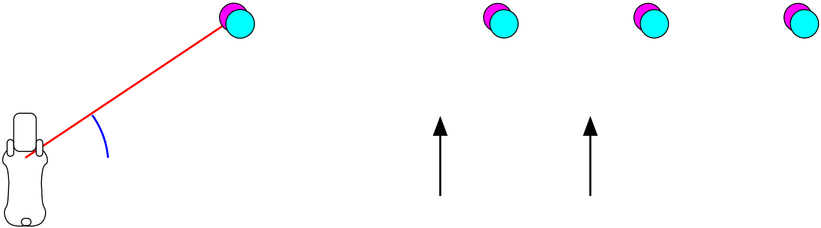
Ambiguous Landmarks

- What if the landmarks are ambiguous?
- Update each particle based on most likely landmark
  - Compute similarity for each possible landmark
  - Update particle using most likely landmark (most similar)



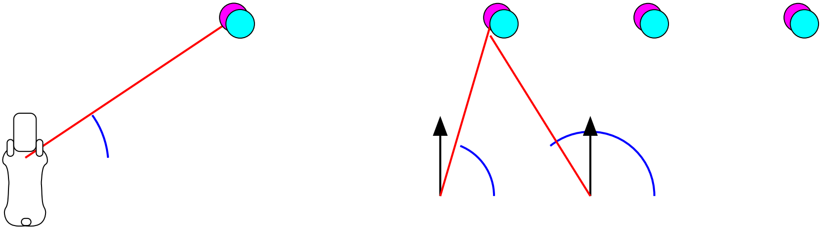
Ambiguous Landmarks

- What if the landmarks are ambiguous?
- Update each particle based on most likely landmark
  - Compute similarity for each possible landmark
  - Update particle using most likely landmark (most similar)



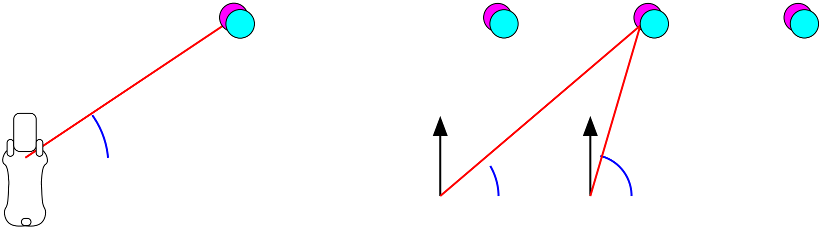
Ambiguous Landmarks

- What if the landmarks are ambiguous?
- Update each particle based on most likely landmark
  - Compute similarity for each possible landmark
  - Update particle using most likely landmark (most similar)



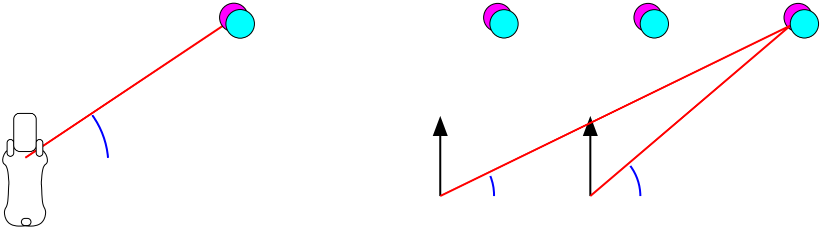
Ambiguous Landmarks

- What if the landmarks are ambiguous?
- Update each particle based on most likely landmark
  - Compute similarity for each possible landmark
  - Update particle using most likely landmark (most similar)



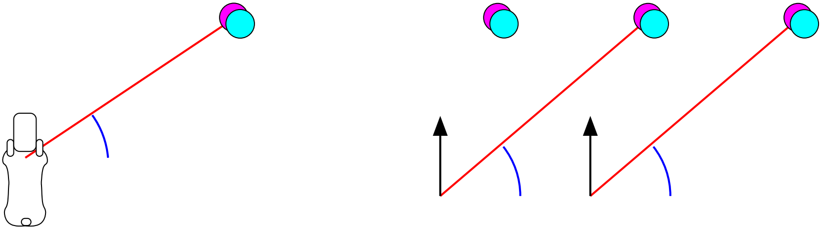
Ambiguous Landmarks

- What if the landmarks are ambiguous?
- Update each particle based on most likely landmark
  - Compute similarity for each possible landmark
  - Update particle using most likely landmark (most similar)



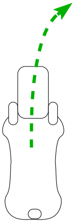
Ambiguous Landmarks

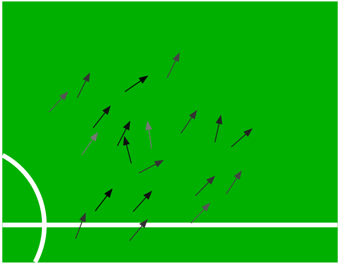
- What if the landmarks are ambiguous?
- Update each particle based on most likely landmark
  - Compute similarity for each possible landmark
  - Update particle using most likely landmark (most similar)



Motion Update

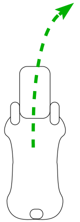
- Need motion model: $p(h'|h, a)$
  - Predict new pose given previous hypothesis and action
- Update each particle when robot moves
  - Use odometry velocities to translate particles

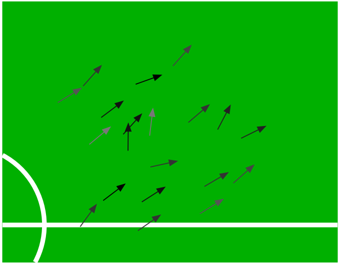




Motion Update

- Need motion model: $p(h'|h, a)$
  - Predict new pose given previous hypothesis and action
- Update each particle when robot moves
  - Use odometry velocities to translate particles

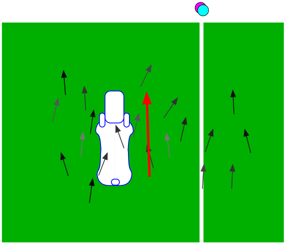




Outline

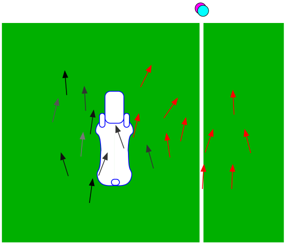
Negative Information

- Expect to see a landmark, but do not see it
- How it works
  - Particles expect to see a landmark
  - Robot does not see the landmark
  - Update the particles with a lower probability



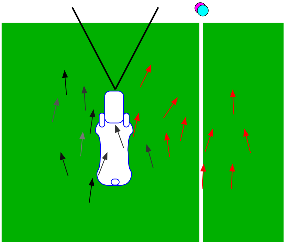
Negative Information

- Expect to see a landmark, but do not see it
- How it works
  - Particles expect to see a landmark
  - Robot does not see the landmark
  - Update the particles with a lower probability



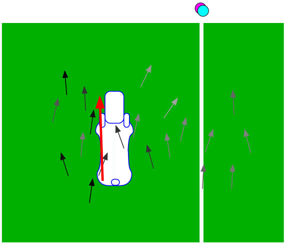
Negative Information

- Expect to see a landmark, but do not see it
- How it works
  - Particles expect to see a landmark
  - Robot does not see the landmark
  - Update the particles with a lower probability



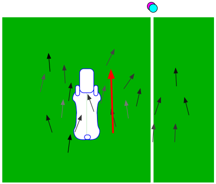
Negative Information

- Expect to see a landmark, but do not see it
- How it works
  - Particles expect to see a landmark
  - Robot does not see the landmark
  - Update the particles with a lower probability



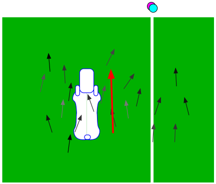
Previous work of Hoffmann et al

- For each particle, assume robot is at that particle's pose
  - For each landmark
    - Determine if landmark should be in field of view
    - Check if landmark was seen
    - If it was not seen,
    - Update particle with probability of missing an observation



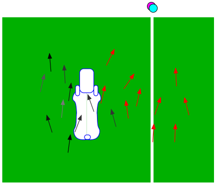
Previous work of Hoffmann et al

- For each particle, assume robot is at that particle's pose
  - For each landmark
    - Determine if landmark should be in field of view
    - Check if landmark was seen
    - If it was not seen,
    - Update particle with probability of missing an observation



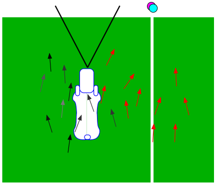
Previous work of Hoffmann et al

- For each particle, assume robot is at that particle's pose
  - For each landmark
    - Determine if landmark should be in field of view
    - Check if landmark was seen
    - If it was not seen,
    - Update particle with probability of missing an observation



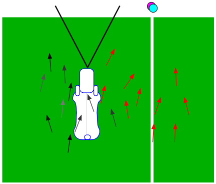
Previous work of Hoffmann et al

- For each particle, assume robot is at that particle's pose
  - For each landmark
    - Determine if landmark should be in field of view
    - Check if landmark was seen
    - If it was not seen,
    - Update particle with probability of missing an observation



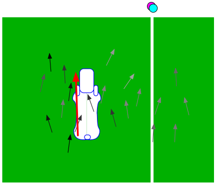
Previous work of Hoffmann et al

- For each particle, assume robot is at that particle's pose
  - For each landmark
    - Determine if landmark should be in field of view
    - Check if landmark was seen
    - If it was not seen,
    - Update particle with probability of missing an observation



Previous work of Hoffmann et al

- For each particle, assume robot is at that particle's pose
  - For each landmark
    - Determine if landmark should be in field of view
    - Check if landmark was seen
    - If it was not seen,
    - Update particle with probability of missing an observation



Our approach

- More robust to missed observations of landmarks that are in field of view
- For each particle and landmark
  - Keep counters of number of consecutive frames that:
    - The landmark has been expected to be seen
    - The landmark has not been
  - If both of these counters are over a threshold t,
    - Update particle with probability of missing t observations
- Generalizes Hoffmann et. al's approach
  - Equivalent with $t = 1$

Our approach

- More robust to missed observations of landmarks that are in field of view
- For each particle and landmark
  - Keep counters of number of consecutive frames that:
    - The landmark has been expected to be seen
    - The landmark has not been
  - If both of these counters are over a threshold t,
    - Update particle with probability of missing t observations
- Generalizes Hoffmann et. al's approach
  - Equivalent with $t = 1$

Our approach

- More robust to missed observations of landmarks that are in field of view
- For each particle and landmark
  - Keep counters of number of consecutive frames that:
    - The landmark has been expected to be seen
    - The landmark has not been
  - If both of these counters are over a threshold t,
    - Update particle with probability of missing t observations
- Generalizes Hoffmann et. al's approach
  - Equivalent with $t = 1$

Our approach

- More robust to missed observations of landmarks that are in field of view
- For each particle and landmark
  - Keep counters of number of consecutive frames that:
    - The landmark has been expected to be seen
    - The landmark has not been
  - If both of these counters are over a threshold t,
    - Update particle with probability of missing t observations
- Generalizes Hoffmann et. al's approach
  - Equivalent with $t = 1$

Our approach

- More robust to missed observations of landmarks that are in field of view
- For each particle and landmark
  - Keep counters of number of consecutive frames that:
    - The landmark has been expected to be seen
    - The landmark has not been
  - If both of these counters are over a threshold t,
    - Update particle with probability of missing t observations
- Generalizes Hoffmann et. al's approach
  - Equivalent with $t = 1$

Our approach

- More robust to missed observations of landmarks that are in field of view
- For each particle and landmark
  - Keep counters of number of consecutive frames that:
    - The landmark has been expected to be seen
    - The landmark has not been
  - If both of these counters are over a threshold t,
    - Update particle with probability of missing t observations
- Generalizes Hoffmann et. al's approach
  - Equivalent with $t = 1$

Our approach

- More robust to missed observations of landmarks that are in field of view
- For each particle and landmark
  - Keep counters of number of consecutive frames that:
    - The landmark has been expected to be seen
    - The landmark has not been
  - If both of these counters are over a threshold t,
    - Update particle with probability of missing t observations
- Generalizes Hoffmann et. al's approach
  - Equivalent with $t = 1$

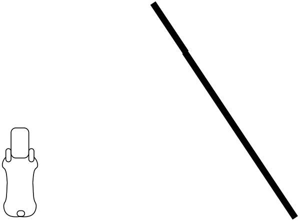
Our approach

- More robust to missed observations of landmarks that are in field of view
- For each particle and landmark
  - Keep counters of number of consecutive frames that:
    - The landmark has been expected to be seen
    - The landmark has not been
  - If both of these counters are over a threshold t,
    - Update particle with probability of missing t observations
- Generalizes Hoffmann et. al's approach
  - Equivalent with $t = 1$

Outline

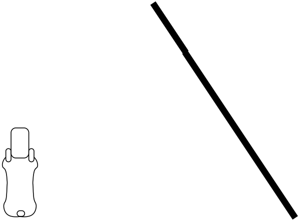
Line Observations

- How do we update the particles when we see a line?
  - Do not know where along the line we are
  - Provides information on distance and orientation to line



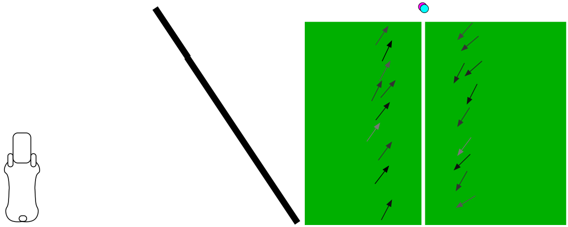
Line Observations

- How do we update the particles when we see a line?
  - Do not know where along the line we are
  - Provides information on distance and orientation to line



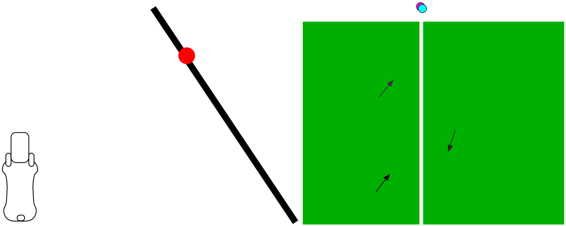
Line Observations

- How do we update the particles when we see a line?
  - Do not know where along the line we are
  - Provides information on distance and orientation to line



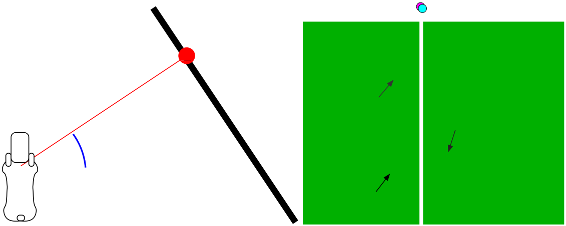
Our approach

- Find closest point on observed line
- Get distance and angle to this point
- Find closest point on line from particle pose
- Update using distance and angle as before
    - As if this were a point landmark



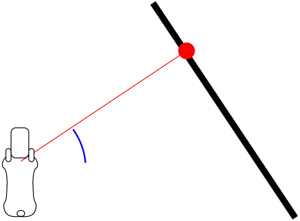
Our approach

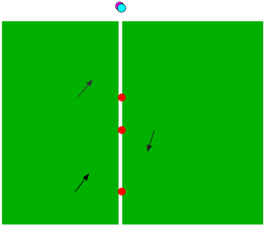
- Find closest point on observed line
- Get distance and angle to this point
- Find closest point on line from particle pose
- Update using distance and angle as before
  - As if this were a point landmark



Our approach

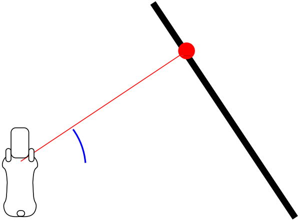
- Find closest point on observed line
- Get distance and angle to this point
- Find closest point on line from particle pose
- Update using distance and angle as before
  - As if this were a point landmark

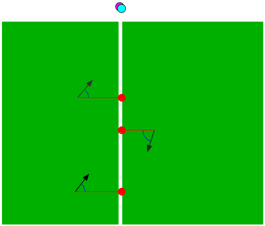




Our approach

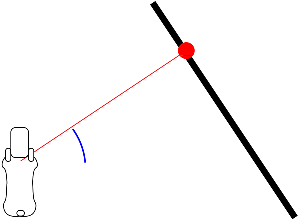
- Find closest point on observed line
- Get distance and angle to this point
- Find closest point on line from particle pose
- Update using distance and angle as before
  - As if this were a point landmark

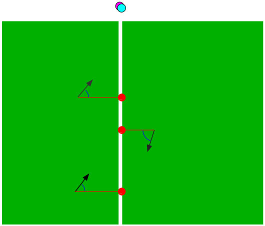




Our approach

- Find closest point on observed line
- Get distance and angle to this point
- Find closest point on line from particle pose
- Update using distance and angle as before
    - As if this were a point landmark



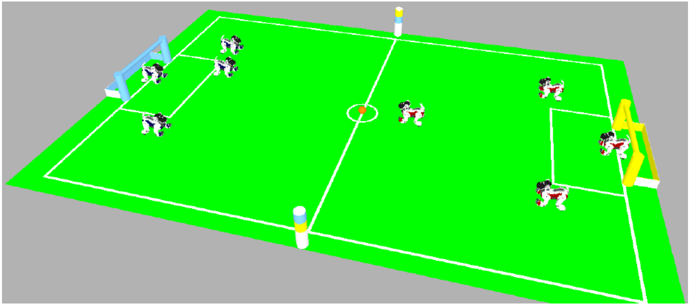


Outline

1. Introduction
   - Localization Problem
   - Particle Filtering
2. Enhancements
   - Negative Information
   - Line Observations
3. Empirical Results
   - Physical Robot Experiments
   - Simulation Experiments

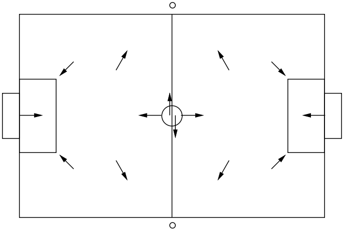
Environment

- RoboCup Legged League field
  - Size: roughly $3.6m \times 5.4m$
  - Landmarks: 2 beacons, 2 goals, 11 field lines



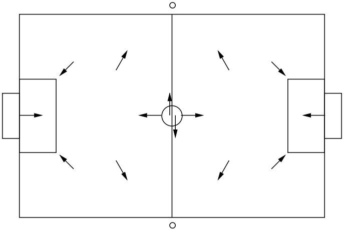
Localization Accuracy

- Visit sequence of 14 points and headings
- After stabilizing at a point, record
  - Actual position and orientation
  - Robot's belief of position and orientation
- Calculate error between actual and believed pose
- Significance results using one-tailed Student's t test



Localization Accuracy

- Visit sequence of 14 points and headings
- After stabilizing at a point, record
  - Actual position and orientation
  - Robot's belief of position and orientation
- Calculate error between actual and believed pose
- Significance results using one-tailed Student's t test



Localization Accuracy

Four Versions of Algorithm

1. Baseline (None)
2. Negative Information (NEG)
3. Line Observations (LINES)
4. Both enhancements (All)

- Average across 10 runs for each

Localization Accuracy

Four Versions of Algorithm

1. Baseline (None)
2. Negative Information (NEG)
3. Line Observations (LINES)
4. Both enhancements (All)

- Average across 10 runs for each

Results

Algorithm	Distance Error(cm)	p-value	Angle Error(deg)	p-value
None	17.67	–	5.39	–
NEG	15.57	0.103	5.16	0.358
LINES	13.62	0.010	5.08	0.381
BOTH	13.38	0.014	4.40	0.104

- Line Observations provide significant improvement
- Both enchancements provide significant improvement
- Some improvement in angular error

Results

Algorithm	Distance Error(cm)	p-value	Angle Error(deg)	p-value
None	17.67	–	5.39	–
NEG	15.57	0.103	5.16	0.358
LINES	13.62	0.010	5.08	0.381
BOTH	13.38	0.014	4.40	0.104

- Line Observations provide significant improvement
- Both enchancements provide significant improvement
- Some improvement in angular error

Results

Algorithm	Distance Error(cm)	p-value	Angle Error(deg)	p-value
None	17.67	–	5.39	–
NEG	15.57	0.103	5.16	0.358
LINES	13.62	0.010	5.08	0.381
BOTH	13.38	0.014	4.40	0.104

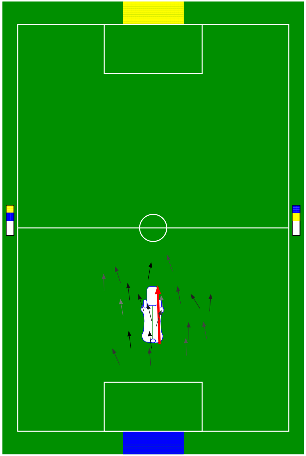
- Line Observations provide significant improvement
- Both enchancements provide significant improvement
- Some improvement in angular error

Outline

1. Introduction
   - Localization Problem
   - Particle Filtering
2. Enhancements
   - Negative Information
   - Line Observations
3. Empirical Results
   - Physical Robot Experiments
   - Simulation Experiments

Simulator

- Abstract noisy observations and movements
- Always know ground truth
- Perturbations repeatable
- Misses set fraction of observations



Kidnapped Robot

- Robot follows figure 8 path
  - Kidnapped once every 30 seconds
    - Placed at center of field at random orientation
- Measure position and angle error
  - Averaged over 2 hours (about 50 laps)
- Measure kidnap recovery time
  - Time for the robot to achieve error less than 20 cm and 20 degrees
- Significance results using one-tailed Student's t test

Kidnapped Robot

- Robot follows figure 8 path
  - Kidnapped once every 30 seconds
    - Placed at center of field at random orientation
- Measure position and angle error
  - Averaged over 2 hours (about 50 laps)
- Measure kidnap recovery time
  - Time for the robot to achieve error less than 20 cm and 20 degrees
- Significance results using one-tailed Student's t test

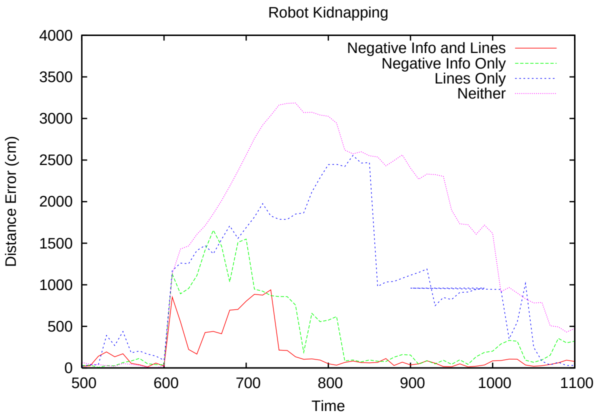
Kidnapped Robot

- Robot follows figure 8 path
  - Kidnapped once every 30 seconds
    - Placed at center of field at random orientation
- Measure position and angle error
  - Averaged over 2 hours (about 50 laps)
- Measure kidnap recovery time
  - Time for the robot to achieve error less than 20 cm and 20 degrees
- Significance results using one-tailed Student's t test

Kidnapped Robot

- Robot follows figure 8 path
  - Kidnapped once every 30 seconds
    - Placed at center of field at random orientation
- Measure position and angle error
  - Averaged over 2 hours (about 50 laps)
- Measure kidnap recovery time
  - Time for the robot to achieve error less than 20 cm and 20 degrees
- Significance results using one-tailed Student's t test

Example Kidnapping



- Kidnapping at time $t = 600$
- Localization Accuracy slowly recovers

Results

Algorithm	Distance Error(cm)	p-value	Angle Error(deg)	p-value	Recovery Time(sec)	p-value
None	46.04	–	15.8	–	7.76	–
NEG	42.29	$< 10^{-6}$	14.0	$< 10^{-7}$	7.53	0.35
LINES	41.19	$< 10^{-8}$	15.3	0.11	6.35	0.01
BOTH	36.74	$< 10^{-29}$	14.2	$< 10^{-5}$	5.86	$< 10^{-3}$

- Algorithm with both enhancements was significantly better than baseline
    - In distance error
    - In angular error
    - In kidnap recovery time
- Algorithm with both enhancements was better than all algorithms in distance error

Results

Algorithm	Distance Error(cm)	p-value	Angle Error(deg)	p-value	Recovery Time(sec)	p-value
None	46.04	–	15.8	–	7.76	–
NEG	42.29	$< 10^{-6}$	14.0	$< 10^{-7}$	7.53	0.35
LINES	41.19	$< 10^{-8}$	15.3	0.11	6.35	0.01
BOTH	36.74	$< 10^{-29}$	14.2	$< 10^{-5}$	5.86	$< 10^{-3}$

- Algorithm with both enhancements was significantly better than baseline
  - In distance error
  - In angular error
  - In kidnap recovery time
- Algorithm with both enhancements was better than all algorithms in distance error

Results

Algorithm	Distance Error(cm)	p-value	Angle Error(deg)	p-value	Recovery Time(sec)	p-value
None	46.04	–	15.8	–	7.76	–
NEG	42.29	$< 10^{-6}$	14.0	$< 10^{-7}$	7.53	0.35
LINES	41.19	$< 10^{-8}$	15.3	0.11	6.35	0.01
BOTH	36.74	$< 10^{-29}$	14.2	$< 10^{-5}$	5.86	$< 10^{-3}$

- Algorithm with both enhancements was significantly better than baseline
    - In distance error
    - In angular error
    - In kidnap recovery time
- Algorithm with both enhancements was better than all algorithms in distance error

Results

Algorithm	Distance Error(cm)	p-value	Angle Error(deg)	p-value	Recovery Time(sec)	p-value
None	46.04	–	15.8	–	7.76	–
NEG	42.29	$< 10^{-6}$	14.0	$< 10^{-7}$	7.53	0.35
LINES	41.19	$< 10^{-8}$	15.3	0.11	6.35	0.01
BOTH	36.74	$< 10^{-29}$	14.2	$< 10^{-5}$	5.86	$< 10^{-3}$

- Algorithm with both enhancements was significantly better than baseline
  - In distance error
  - In angular error
  - In kidnap recovery time
- Algorithm with both enhancements was better than all algorithms in distance error

Comparison to Hoffmann et al's Method

- Hoffmann et. al's Method
  - Update particles with negative information after one missed observation $t = 1$
- Our method
  - Update particles with negative information after landmark has been missed t times
  - For these experiments, $t = 5$

Comparison to Hoffmann et al's Method

- Hoffmann et. al's Method
  - Update particles with negative information after one missed observation $t = 1$
- Our method
  - Update particles with negative information after landmark has been missed t times
  - For these experiments, $t = 5$

Results

Algorithm	Distance Error (cm)	p-value	Angle Error(deg)	p-value
Our Method ($t = 5$)	36.74	–	14.2	–
Hoffmann ($t = 1$)	40.11	$< 10^{-5}$	15.5	$< 10^{-3}$

- Our method performed significantly better
  - In distance errors
  - And angular errors

Results

Algorithm	**Distance Error (cm)**	**p-value**	**Angle Error(deg)**	**p-value**
Our Method ($t = 5$)	36.74	–	14.2	–
Hoffmann ($t = 1$)	40.11	$< 10^{-5}$	15.5	$< 10^{-3}$

- Our method performed significantly better
  - In distance errors
  - And angular errors

Related Work

Negative Information

- Hoffmann et al [Robocup 2005, IROS 2005]
  - First implementation of negative information
  - Take occlusions into account
  - Experiments on still robot

Line Observations

- Röfer et al [Robocup 2003, 2005]
  - Discretize field into grids
  - Pre-calculate where line pixels should be in image for each grid cell
  - Update particles based on similarity in line pixel locations

Summary

- Monte Carlo Localization works well
- Want to to make use of all information available
  - Including Negative Information
  - And Line Observations
- Significantly better than baseline approach