Autonomous Sensor and Action Model Learning for Mobile Robots

Daniel Stronger

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

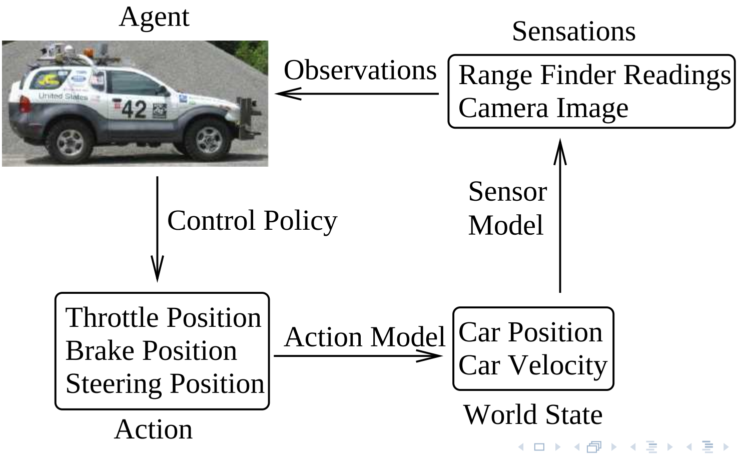
Dissertation Defense
June 19, 2008

Model Learning for Autonomous Robots

- **Goal:** To increase the effectiveness of autonomous mobile robots
- **Plan:** Enable mobile robots to **autonomously learn models** of their **sensors and actions**.

Action and Sensor Models

- Mobile robots rely on models of their actions and sensors



Action and Sensor Models

- Mobile robots rely on models of their actions and sensors

Agent

Sensations

Observations

Range Finder Readings
Camera Image

Control Policy

Sensor
Model

Throttle Position
Brake Position
Steering Position

Action Model

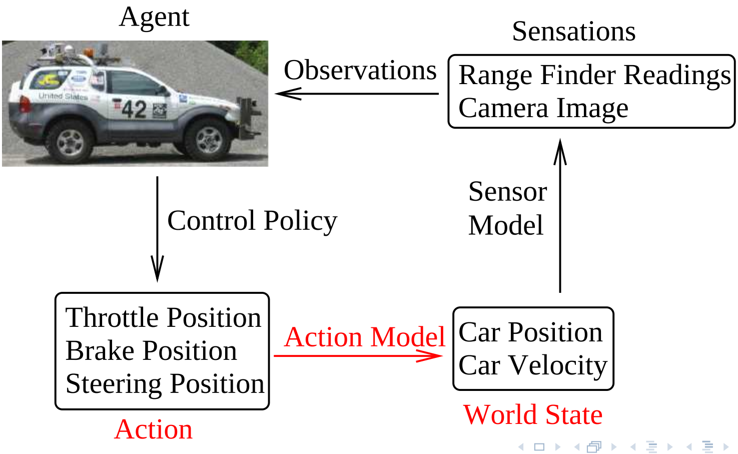
Car Position
Car Velocity

Action

World State

Action and Sensor Models

- Mobile robots rely on models of their actions and sensors



Overview

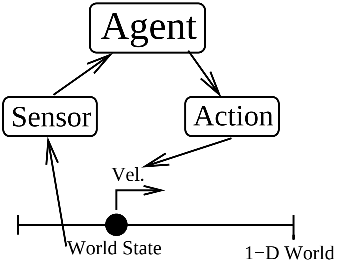
- Action and sensor models are typically calibrated manually: **laborious and brittle**
  - Robot in novel environment might encounter unfamiliar terrain or lighting conditions
  - Parts may wear down over time
- **Goal:** Start without accurate estimate of either model
- Technique is implemented and tested in:
  - One-dimensional scenario: Sony **Aibo** ERS-7
  - Aibo in two-dimensional area
  - Second robotic platform: an **autonomous car**

Outline

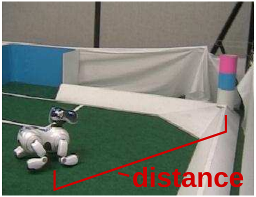
Test-bed Robotic Platform

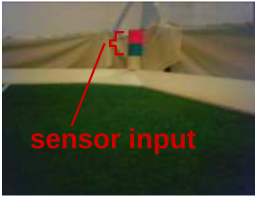
Example: Learning in One Dimension

- Consider a setting with the following properties:
  - The set of world states: **Continuous**, **one-dimensional**
  - **One sensor:** Readings correspond to world states
  - **Range of actions:** Correspond to rates of change
  - Actions and sensors suffer from **random noise**



Experimental Setup





- Sensor model maps **landmark height in image** to distance
  - Mapping derived from camera specs not accurate
- Action model maps **parametrized walking action**, W(x), to velocity
  - Parameter x corresponds to **attempted velocity**, not accurate because of friction and joint behavior

The Sensor and Action Models

- Each model informs an estimate of the world state:
  - The sensor model maps an observation to a world state

$$x_s(t_k) = S(obs_k)$$

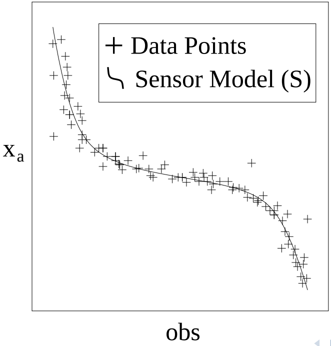
  - The action model maps an action $C(t)$ to a velocity

$$x_a(t) = x(0) + \int_0^t A(C(s))\,ds$$

- Goal is to learn two model functions, A and S
  - Use **polynomial regression** as a **function approximator**

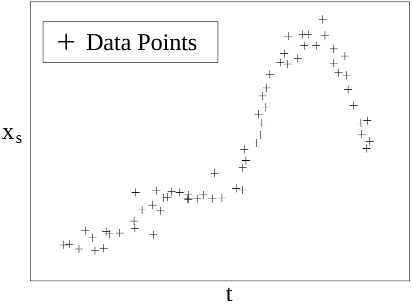
Learning a Sensor Model

- Assume a given action model is accurate
- Consider ordered pairs $(obs_k, x_a(t_k))$
- Fit polynomial to data



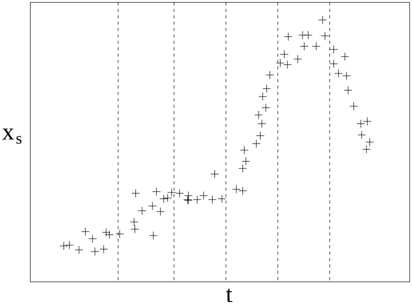
Learning an Action Model

- Assume a given sensor model is accurate
- Plot $x_s(t)$ against time



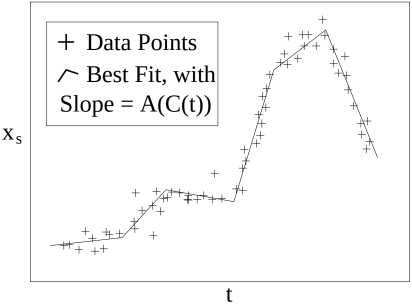
Learning an Action Model

- Assume a given sensor model is accurate
- Plot $x_s(t)$ against time



Learning an Action Model (cont.)

- Compute action model that minimizes the error
- Problem equivalent to another multivariate regression

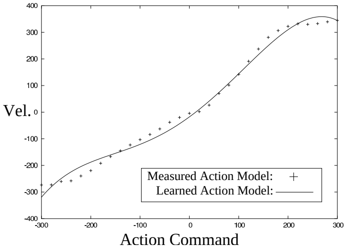


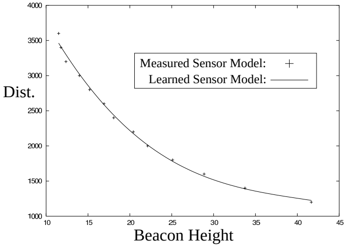
Learning Both Models Simultaneously

- Given very little to start with, learn both models
- Maintain **both state estimates**, $x_s(t)$ and $x_a(t)$
- Each one is used to fit the other model
- Both models grow in accuracy through bootstrapping process
- Use **weighted regression**
  - More **recent** points weighted higher
  - $w_i = \gamma^{n-i}$, $\gamma < 1$
  - Can still be computed incrementally

Learned Models

- Measure actual models with stopwatch and ruler
- Use optimal scaling to evaluate learned models





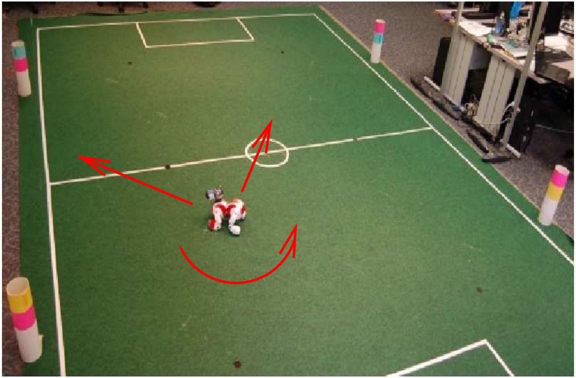
- Sensor model average error: 70.4 mm, 2.9% of range
- Action model average error: 29.6 mm/s, 4.9% of range

Learning in Two Dimensions

- Robot learns while traversing rectangular field
  - Combinations of forward, sideways, and turning motion
  - Field has four known color-coded cylindrical landmarks

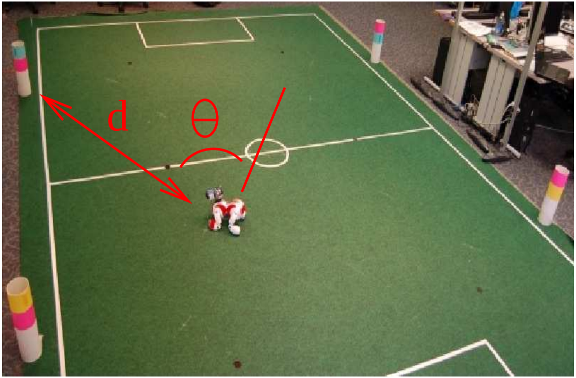
Learning in Two Dimensions

- Robot learns while traversing rectangular field
  - Combinations of forward, sideways, and turning motion
  - Field has four known color-coded cylindrical landmarks



Learning in Two Dimensions

- Robot learns while traversing rectangular field
  - Combinations of forward, sideways, and turning motion
  - Field has four known color-coded cylindrical landmarks



Aibo in Two Dimensions: Sensor Model

- Sensor model maps **distance to landmark** to distribution of observed **height in image**
- Model includes **polynomial function**, $f(\mathrm{dist}(s))$
- Also, the **variance of random noise** added to image heights
- Additional variance parameter for landmark's **horizontal angle**

Aibo in Two Dimensions: Action Model

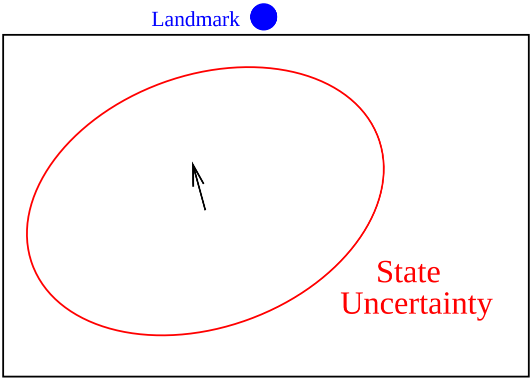
- Actions correspond to attempted combinations of **forward, sideways, and turning** velocities
  - Attempted velocities control step size, direction
  - Inaccuracies due to friction, joint behavior
  - Action model maps attempted velocities to **actual velocities**
  - Discrete set of 40 actions used

Challenges

- This problem presents many challenges:
  - How do we incorporate actions and sensations into the world state?
    - For state estimation, **Kalman filtering**
  - How do we determine what models are most consistent with the observed data?
    - For maximum likelihood parameter estimation, the **Expectation-Maximization (EM) algorithm**

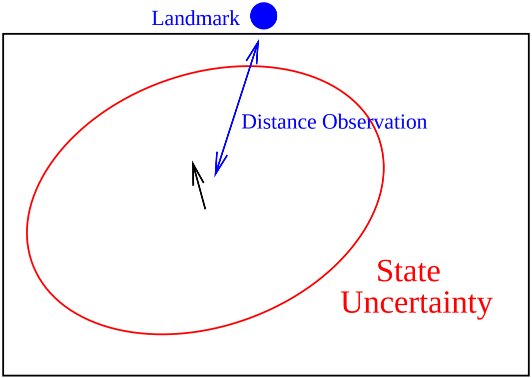
Kalman Filtering

- Kalman filter maintains successive state estimates
  - Represents mean and covariance of distribution



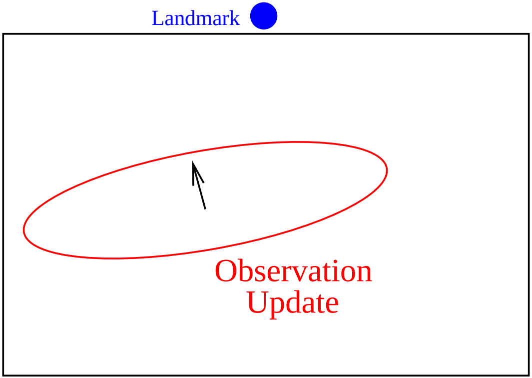
Kalman Filtering

- Kalman filter maintains successive state estimates
  - Represents mean and covariance of distribution



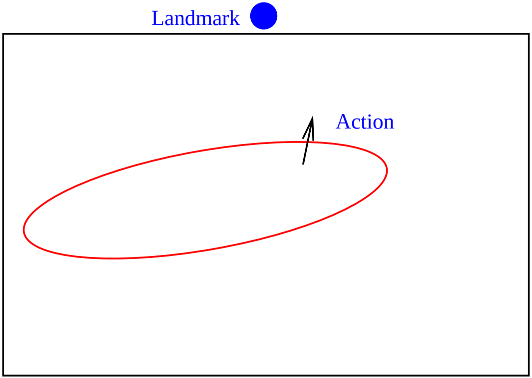
Kalman Filtering

- Kalman filter maintains successive state estimates
  - Represents mean and covariance of distribution



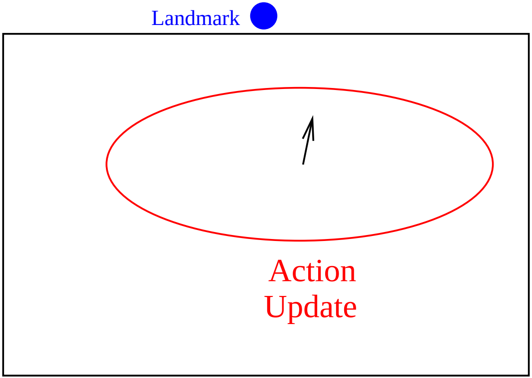
Kalman Filtering

- Kalman filter maintains successive state estimates
  - Represents mean and covariance of distribution



Kalman Filtering

- Kalman filter maintains successive state estimates
  - Represents mean and covariance of distribution



Maximum Likelihood Estimation

- **Known:** robot's actions and observations
- **Hidden variables:** world state over time
- Goal is to learn system **parameters**: action and sensor models
- **Approach:** Find models with **maximum likelihood** of producing observed data with the **EM Algorithm**
  - **E-step:** Given models, find probability **distribution** over world state
  - **M-step:** Given distribution, find maximum likelihood **models**
  - Alternate until convergence

Maximum Likelihood Estimation

- **Known:** robot's actions and observations
- **Hidden variables:** world state over time
- Goal is to learn system **parameters**: action and sensor models
- **Approach:** Find models with **maximum likelihood** of producing observed data with the **EM Algorithm**
  - **E-step:** Given models, find probability **distribution** over world state
  - **M-step:** Given distribution, find maximum likelihood **models**
  - Alternate until convergence

Maximum Likelihood Estimation

- **Known:** robot's actions and observations
- **Hidden variables:** world state over time
- Goal is to learn system **parameters**: action and sensor models
- **Approach:** Find models with **maximum likelihood** of producing observed data with the **EM Algorithm**
  - **E-step:** Given models, find probability **distribution** over world state
  - **M-step:** Given distribution, find maximum likelihood **models**
  - Alternate until convergence

Learning the Robot's Models

- The E-step determines a probability distribution over the robot's pose over time
  - The **Extended Kalman Filter and Smoother** (EKFS) approximates these distributions as multivariate Gaussians
- Definition of M-step: New parameters **maximize expected log likelihood** of observations with respect to E-step distribution
  - **Adapting the M-step** to learn these models is a contribution of this work

Adapting the M-step

- Given E-step distribution $\hat{\mathrm{p}}$ and observations O, find parameters λ that maximize $\mathrm{E}_{\hat{\mathrm{p}}}[\log \mathrm{p}(O|\lambda)]$
- Equivalently, find the action model, a, that maximizes:

$$\sum_{t=1}^{T} \underbrace{\int_{s_{t-1},s_t} \hat{\mathrm{p}}(s_{t-1}, s_t)}_{\text{expected}} \log \underbrace{\mathrm{p}(s_t|s_{t-1}, a)}_{\text{action likelihood}} \, ds_{t-1} ds_t$$

- and the sensor model, b, that maximizes:

$$\sum_{t=1}^{T} \underbrace{\int_{s_t} \hat{\mathrm{p}}(s_t)}_{\text{expected}} \log \underbrace{\mathrm{p}(o_t|s_t, b)}_{\text{observation likelihood}} \, ds_t$$

Learning the Sensor Model

- According to M-step, must find sensor model b that maximizes $\sum_{t=1}^{T} \int_{s_t} \hat{\mathrm{p}}(s_t) \log \mathrm{p}(o_t | s_t, b)\, ds_t$
- Equivalently, find sensor model function f that minimizes:

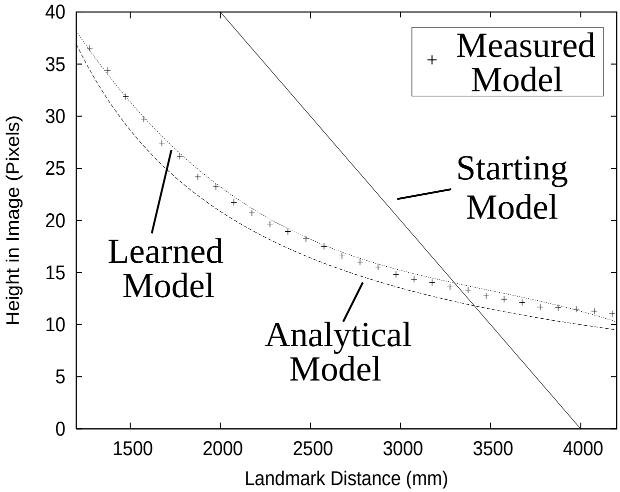
$$\sum_{t=1}^{T} \underbrace{\int_{s_t} \hat{\mathrm{p}}(s_t)}_{\text{weighted mean}} \underbrace{(f(\mathrm{dist}(s_t)) - o_t)^2}_{\text{squared error}}\, ds_t$$

- Minimize the error with weighted polynomial regression
  - Solution approximated by **drawing samples** from $\hat{\mathrm{p}}(s_t)$
- New **variances** are model's weighted mean square errors
- Additional derivation yields new **velocities** for each action

Experimental Validation

- Experiments were performed on the Aibo and in simulation
  - Simulator models the robot's pose over time with noisy actions and observations
- **Random actions** were chosen with certain constraints:
  - Each chosen action was executed for five seconds at a time
  - The robot stays on the field
  - Actions are evenly represented
- Actual action and sensor models were measured
  - Compared to learned models
- Time for data collection: **25 minutes**; Learning on real world data:**10 minutes**; On simulated data: **1 hour**

The Learned Sensor Model



Learned Standard Deviations

Std. Dev	Starting	Actual	Learned
Real Height (pix)	10.0	1.59	1.69
Real Angle (rad)	0.2	0.027	0.012
Sim. Height (pix)	10.0	1.0	0.980
Sim. Angle (rad)	0.2	0.5	0.474

- Error in real angle standard deviation likely caused by relative accuracy of angle observations

The Learned Action Model

Velocity	Avg. Error	Compared to Range
Real Angular	0.135 rad/s	3.2%
Sim. Forwards	18.34 mm/s	2.2%
Sim. Sideways	23.06 mm/s	3.2%
Sim. Angular	0.086 rad/s	2.7%

- By comparison, attempted angular velocities have average error of 0.333 rad/s or 7.9%

Outline

The Autonomous Car

- **Self-driving car** provides many challenges for autonomous model learning

- Actions lead to accelerations, angular velocity:
  - Throttle, brake, and steering position
- Sensors provide information about pose and map:
  - Three-dimensional LIDAR
- Again start without accurate estimate of either model

Action Model

- Example model of **acceleration**, a:

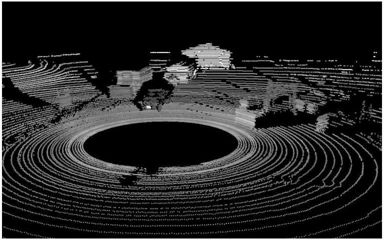
$$a_t = C_1 + C_2 \mathrm{throttle}_t + C_3 \mathrm{vel}_t + C_4 \mathrm{vel}_t \mathrm{brake}_t$$

- And **angular velocity**, ω:

$$\omega_t = C_1 \mathrm{vel}_t + C_2 \mathrm{vel}_t \mathrm{steer}_t$$

Three-Dimensional LIDAR

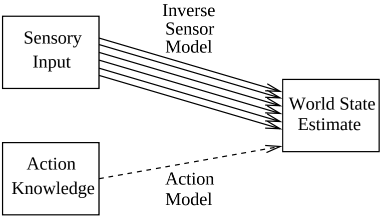
- The **Velodyne LIDAR** sensor:



- 64 lasers return distance readings
- Each laser is at a different vertical angle and different **horizontal offset**
- Unit spins around vertical axis at 10Hz

Autonomous Car Challenges

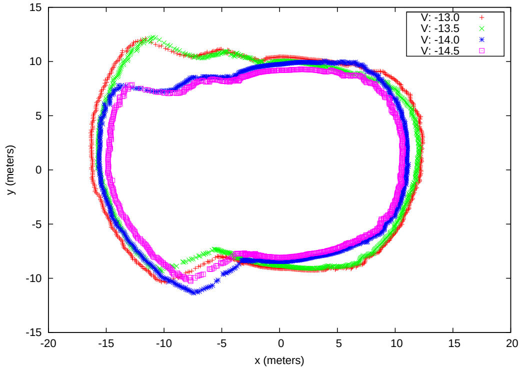
- Structure of environment is unknown
  - Component of world state
- High bandwidth sensor: **perceptual redundancy**



- **Plan:** Learn the sensor model first
- **Assumption:** Nearby angles have similar distances

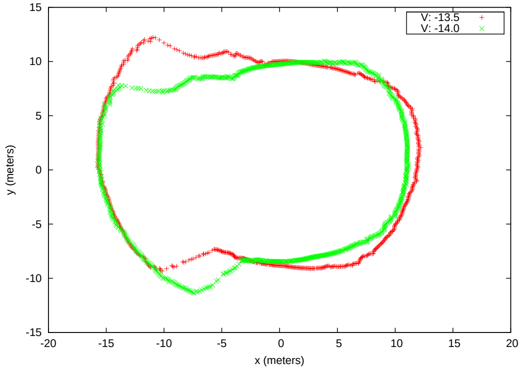
Learning the Sensor Model

- Top view of **uncalibrated** laser projections



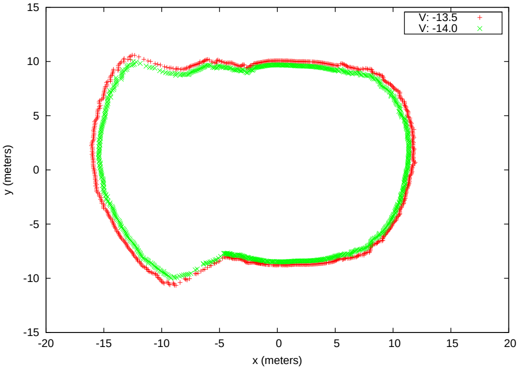
Learning the Sensor Model

- Consider pairs of vertically adjacent lasers



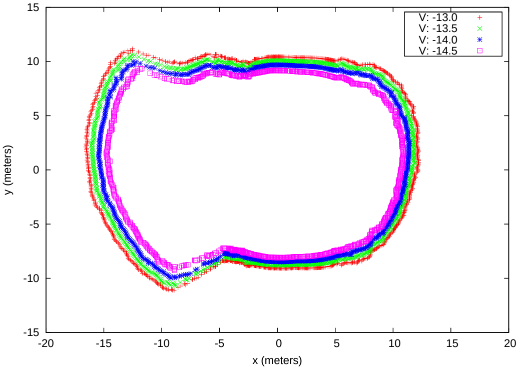
Learning the Sensor Model

- Normalized cross-correlation identifies the angle difference



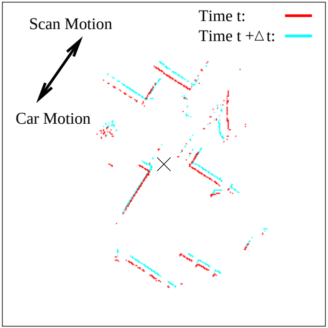
Learning the Sensor Model

- Process yields a relative horizontal angle for each laser



Identifying Car Motion

- Matching scans at consecutive times yields the car's motion
  - Transformation is determined by **Iterative Closest Point**



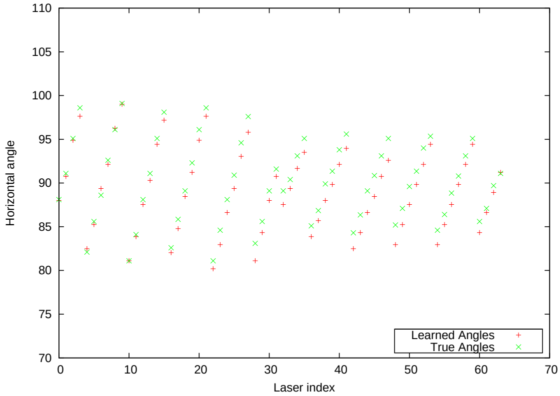
Learning the Action Model

- Given car motion estimates from ICP:
  - Determine overall orientation of Velodyne
    - **Define “forwards”** to be the car’s median direction
  - Combine with action command to **train action model**
- Learned action model yields more accurate car motion estimates
  - New motion estimates are used as starting points for ICP in iterative procedure

Experimental Setup

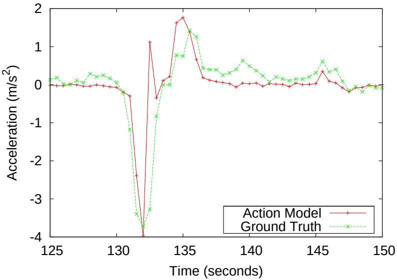
- Car collected data while driving autonomously for 200 seconds
- Sensor model evaluated by comparison to **ground truth:**
  - Horizontal angles calibrated manually by Velodyne
- For ground truth motions, Applanix sensor was used:
  - Combined GPS and inertial motion sensor
  - Ground truth accelerations and angular velocities compared to action model output

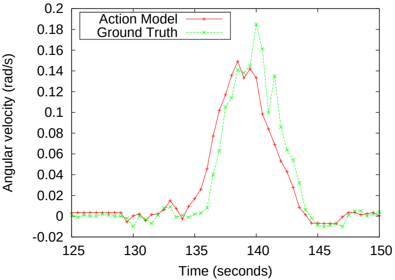
Learned Sensor Model



- Average horizontal angle error is 0.54°, 3.0% of range

Learned Action Model





- Average acceleration error is $0.39\mathrm{m/s}^2$, 6.8% of range
- Average angular velocity error is $0.74^\circ/\mathrm{s}$, 6.4% of range

Outline

Related Work

- Developmental Robotics:
  - [Pierce and Kuipers, ’97; Weng et al., ’01; Oudeyer et al., ’04; Olsson et al., ’06]
- Learning a sensor model:
  - [Tsai, ’86; Moravec and Blackwell, ’93; Hahnel et al., ’04]
- Learning an action model:
  - [Roy and Thrun, ’99; Martinelli et al., ’03; Duffert and Hoffmann, ’05]
- Dual estimation for Kalman filters:
  - [Ghahramani and Roweis, ’99; Briegel and Tresp, ’99; de Freitas et al., ’99]

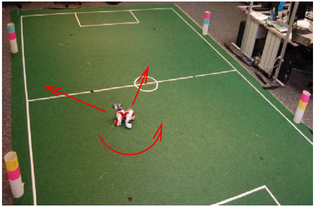
Summary

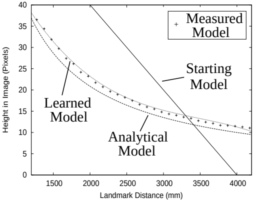
- Developed novel methodology that enables a mobile robot to **autonomously learn action and sensor models**
- Method validated on:
  - **Sony Aibo** in one-dimensional scenario
  - Aibo and simulation in two dimensions
  - **Autonomous car**

Future Work

- Adapt method to other robots, **more detailed models**
  - Explore possibilities for **learning the features**
- Learn about shapes, affordances of environmental **objects**
- Incorporate **curiosity** mechanisms

Questions?

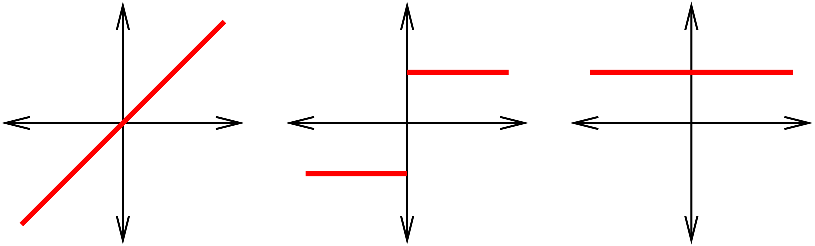




- **Thanks:** UT Austin Villa and Austin Robot Technology

Aibo in One Dimension: Additional Results

- Tried three different functions for the **initial action model** estimate, A_0



$A_0(c) = c$ $\quad$ $A_0(c) = \mathrm{Sgn}(c)$ $\quad$ $A_0(c) = 1$

Aibo in One Dimension: Additional Results

- Tried three different functions for the **initial action model** estimate, A_0
- Ran 15 trials on each starting point
- Recorded number of successes, average errors
- Even with **no information** ($A_0(c) = 1$), success in $10/15$ trials

A_0	**Sensor Model (mm)**	**Action Model (mm/s)**	**Success Rate**
$A_0(c) = c$	70.4 ± 13.9	29.6 ± 12.4	$15/15$
$A_0(c) = \mathrm{Sgn}(c)$	85.3 ± 24.5	31.3 ± 9.2	$15/15$
$A_0(c) = 1$	88.6 ± 11.5	27.3 ± 6.2	$10/15$

Additional Results: EM in one dimension

- Goal: Apply **EM**-based algorithm to Aibo in **one-dimensional domain**
  - Action model: Table-based function from actions to forwards velocities
  - Sensor model: Polynomial from landmark distance to image height
- **Results:**
  - Average sensor model error: 0.83 pixels
  - Average action model error: 22.1 mm/s

Learning the Action Model

- In the M-step, for each action, A, must maximize:

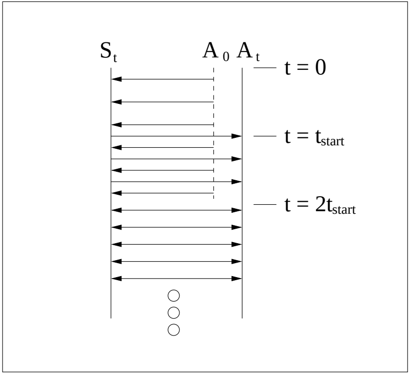
$$\sum_{t:c(t)=A} \int_{s_{t-1},s_t} \hat{\mathrm{p}}(s_{t-1}, s_t) \log \mathrm{p}(s_t | s_{t-1}, a)\, ds_{t-1} ds_t$$

- Action model is determined by μ_A, the **mean pose displacement** caused by action A over one time-step.
- Derivation yields an expression for μ_A^*, the maximizing displacement:

$$\frac{1}{|\{t : c(t) = A\}|} \sum_{t:c(t)=A} \int_{s_{t-1},s_t} \hat{\mathrm{p}}(s_{t-1}, s_t) \underbrace{d(s_{t-1}, s_t)}_{\text{displacement}} ds_{t-1} ds_t$$

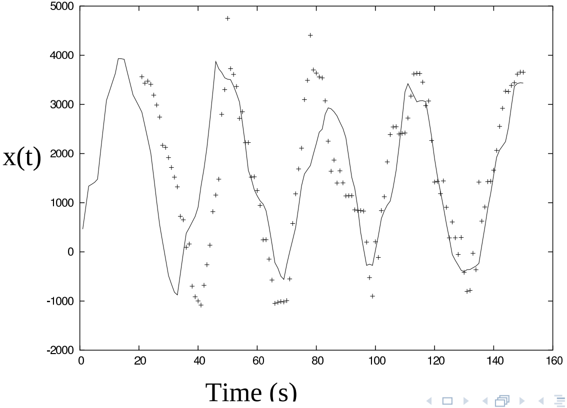
Aibo in 1D: Learning Both Models

- Ramping up process



Aibo in 1D: Estimates Converge

- Over time, $x_s(t)$ and $x_a(t)$ come into stronger agreement



Aibo in 1D: Learning Curves

- Average fitness of model over all 15 runs over time