Inferring User Intent for Learning by Observation

Kevin R. Dixon

krd@cs.cmu.edu

Department of Electrical & Computer Engineering

Carnegie Mellon University

Electrical & Computer ENGINEERING

Task Automation

- Many users repeat their regular tasks manually
  - Reading news
  - Watering plants
  - Sorting emails

- In the robot domain
  - Many industrial tasks are performed manually
  - Not automated at all

Task Automation

- Many users repeat their regular tasks manually
  - Reading news
  - Watering plants
  - Sorting emails
- In the robot domain
  - Many industrial tasks are performed manually
  - Not automated at all
- Majority of systems require that users convey knowledge with procedural-programming techniques
  - Effectively precludes automation in most cases

Goal of This Work

- Create a system that automates motor-skill tasks by learning from observations of user demonstrations

Motor-Skill Tasks

- An observable *state* sequence is sufficient to complete the task
- State sequence maps onto a set of motor commands
- Majority of industrial automation involves motor-skill tasks

Outline

- Background
- Modeling the User
- Predictive Robot Programming
- Hypothesizing about User Actions
- Learning By Observation
- Conclusions and Open Issues

Background

- There have been many approaches to improving task automation
- Iconic programming (Gertz et al., 1995)
- Multi-modal input (Iba et al., 2002)
- Cognitive modeling (Forsythe & Xavier, 2002)
- Task automation based on a user demonstration is called
  - Learning By Observation (LBO)
  - Programming by Demonstration
  - Teaching from Example

Learning by Observation

- Learning to drive a car (Pomerleau, 1991)
- User-interface agents (Cypher, 1993)
- Manipulator-robot programming (Asada & Asari, 1988)
- Manipulation tasks (Friedrich et al., 1996)
- Assembly tasks (Chen & Zelinsky, 2003)

Learning by Observation

- Learning to drive a car (Pomerleau, 1991)
- User-interface agents (Cypher, 1993)
- Manipulator-robot programming (Asada & Asari, 1988)
- Manipulation tasks (Friedrich et al., 1996)
- Assembly tasks (Chen & Zelinsky, 2003)

- However, each of these systems *imitates* the user

Problem with Imitation

- Initially, the self-driving system could not drive for long distances (Pomerleau, 1991).
- This is because drivers never show the system how to correct during normal operation (Pomerleau, 1996)

Problem with Imitation

- Initially, the self-driving system could not drive for long distances (Pomerleau, 1991).
- This is because drivers never show the system how to correct during normal operation (Pomerleau, 1996)
- The solution was to infer user intent

User Intent

- Consider user demonstrations as a sequence of goal-directed actions
- We interpret demonstrations as inferring user intent

User Intent

- Consider user demonstrations as a sequence of goal-directed actions
- We interpret demonstrations as inferring user intent
- Informally, user intent is the sequence of actions invariant from
  - Changes in the environment
  - Human imprecision
  - Sensor noise

User Intent

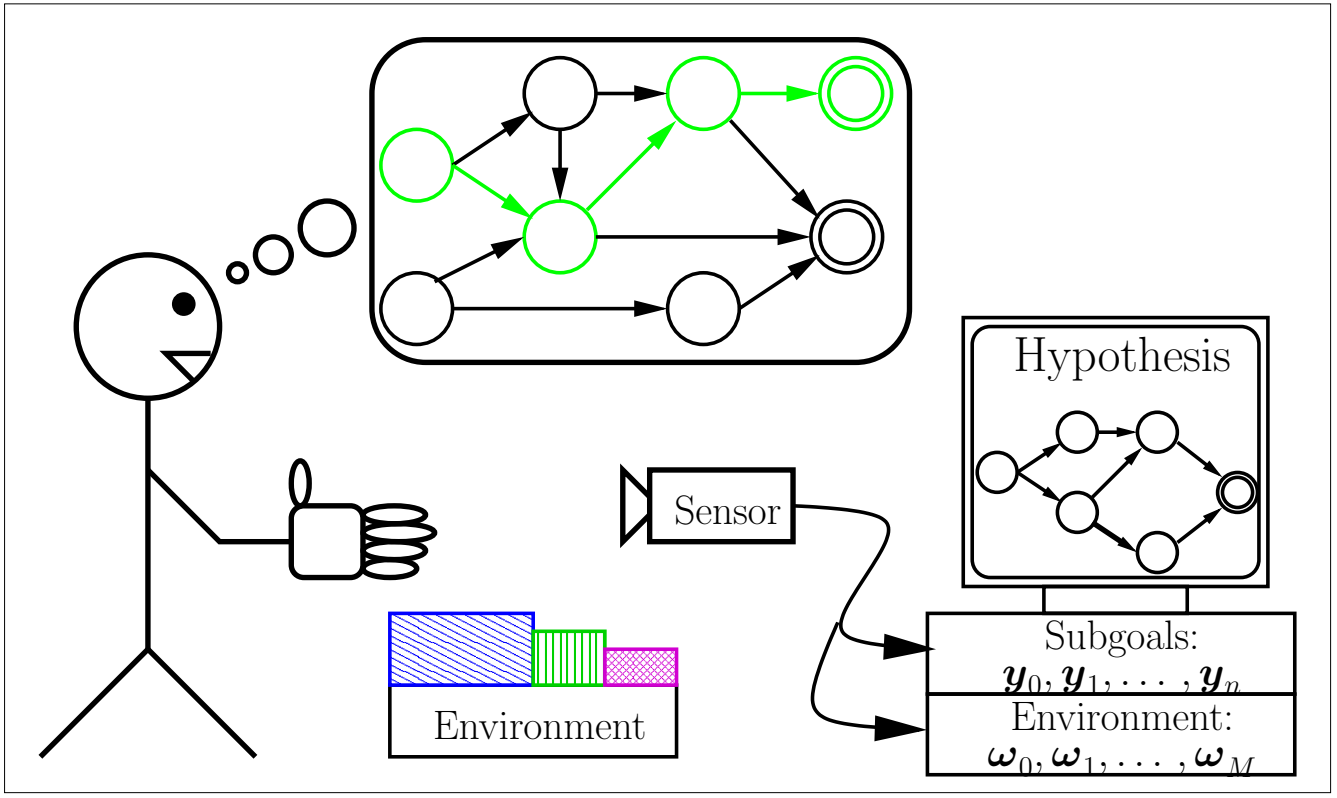
- Consider user demonstrations as a sequence of goal-directed actions
- We interpret demonstrations as inferring user intent
- Informally, user intent is the sequence of actions invariant from
  - Changes in the environment
  - Human imprecision
  - Sensor noise
- Even more informally, “what the user would have done”

User Intent

- Inferring user intent is relatively new
- Hierarchical search for invariants (Alissandrakis et al., 2002; Billard et al., 2003)
- Finite-state machines (Skubic & Volz, 2000; Nicolescu & Matarić, 2001)

Our Approach

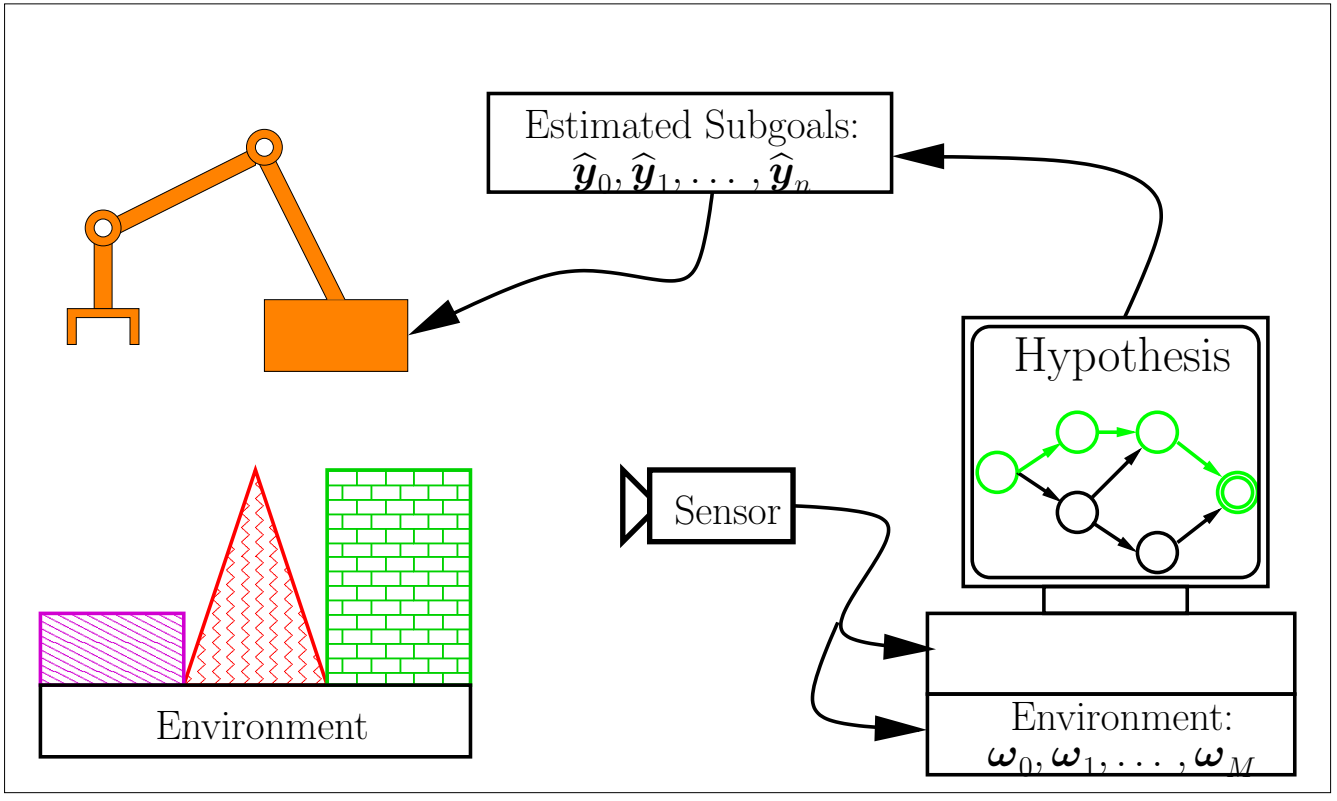
- Build a statistical model based on user observations



- Extract a set of subgoals as a function of the environment

Our Approach

- Hypothesize about “what the user would have done”



- System provides the sequence of subgoals specifying *how* to achieve the goal

Outline

- Background
- Modeling the User
- Predictive Robot Programming
- Hypothesizing about User Actions
- Learning By Observation
- Conclusions and Open Issues

Modeling the User

- Due to the low precision of humans, observations can be considered noise corrupted
- There are hidden, or latent, causes for user behavior
  - Many possible ways to complete a task
  - Difficult to instrument fully realistic environments

Modeling the User

- Due to the low precision of humans, observations can be considered noise corrupted
- There are hidden, or latent, causes for user behavior
  - Many possible ways to complete a task
  - Difficult to instrument fully realistic environments
- One flexible model of the user is a Continuous-Density Hidden Markov Model (CDHMM)

Estimating a CDHMM

- CDHMMs have continuous observations
- Many researchers have used HMMs to describe human actions (Hannaford & Lee, 1991)
  - Used fixed-topology models (Hovland et al., 1996)

Estimating a CDHMM

- CDHMMs have continuous observations
- Many researchers have used HMMs to describe human actions (Hannaford & Lee, 1991)
  - Used fixed-topology models (Hovland et al., 1996)
- We do not know what tasks the user will perform *a priori*
- Estimate the structure of the target CDHMM from user observations

Structure Estimation in CDHMMs

- Fixed-topology HMMs have well-known algorithms *e.g.* Baum-Welch, Viterbi (Rabiner, 1989)
- Estimating HMM structure is hard (Gillman & Sipser, 1994; Abe & Warmuth, 1992)
- Most work focuses on
  - Heuristic methods (Stolcke & Omohundro, 1994)
  - Special subclasses (Ron et al., 1998)
  - Asymptotic behavior (Ephraim & Merhav, 2002)
- Have only been used in theory or hand-crafted problems
  - Except in speech recognition (Singh et al., 2002)

Learning Algorithm Overview

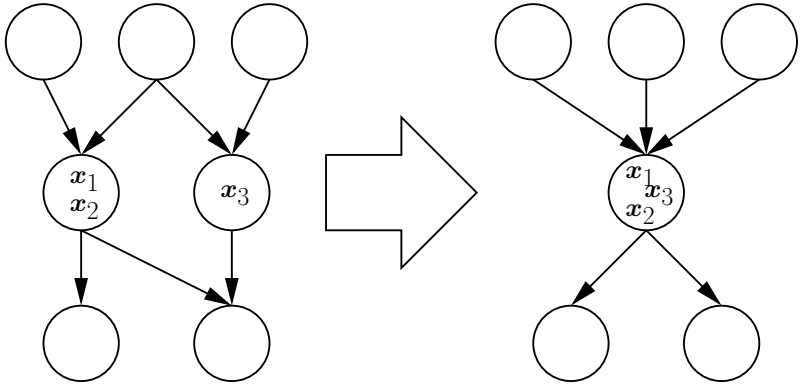
- Input:
  - Set of demonstrations
- Output:
  - CDHMM describing the demonstrations

Learning Algorithm Overview

- Input:
  - Set of demonstrations
- Output:
  - CDHMM describing the demonstrations
- Requirements:
  - Continuous-density observations
  - CDHMM should be simple
  - Low computational complexity
  - Correctness

Our Approach

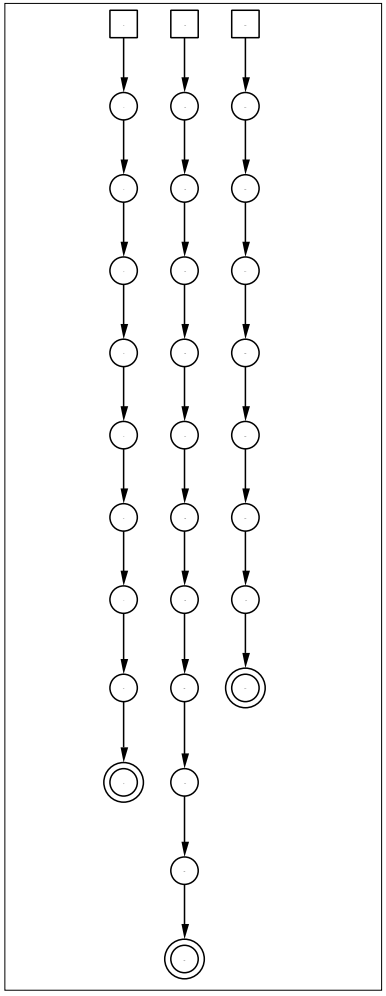
- Consider each task as a random walk through a target CDHMM
- Assign each observation to a node in a graph



- Repeatedly merge similar nodes
- Use fixed-topology estimation on resulting structure

Similarity Merging In Action

Original



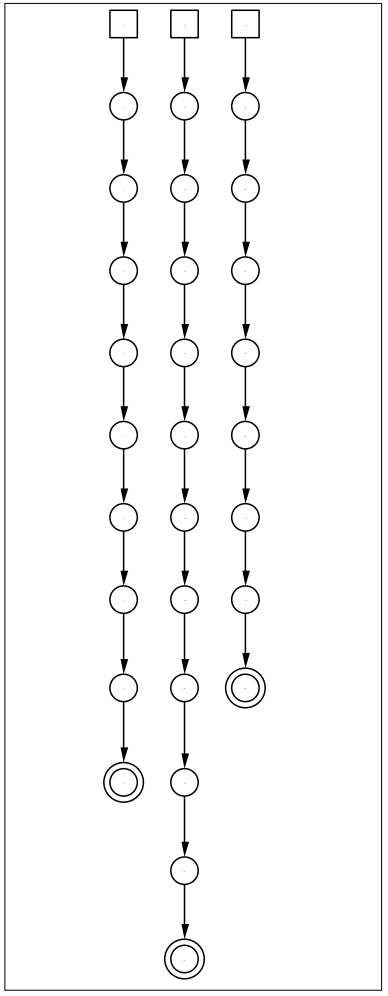
Loose

Strict

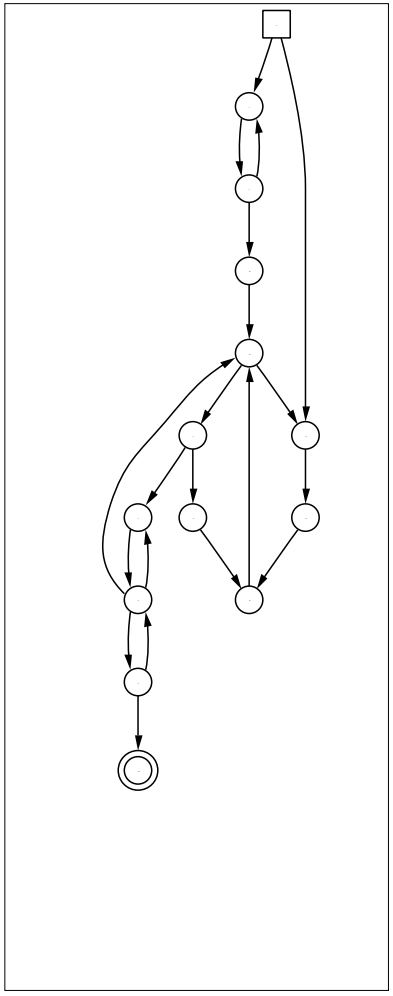
- Merging using “loose” and “strict” definitions of similarity

Similarity Merging In Action

Original



Loose

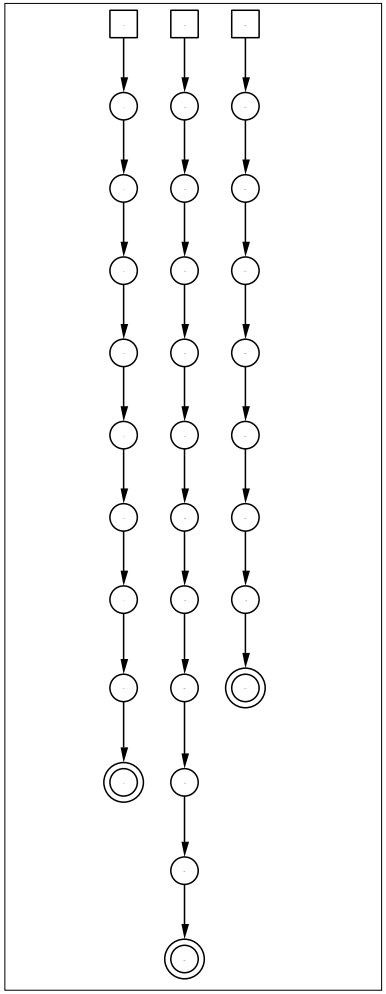


Strict

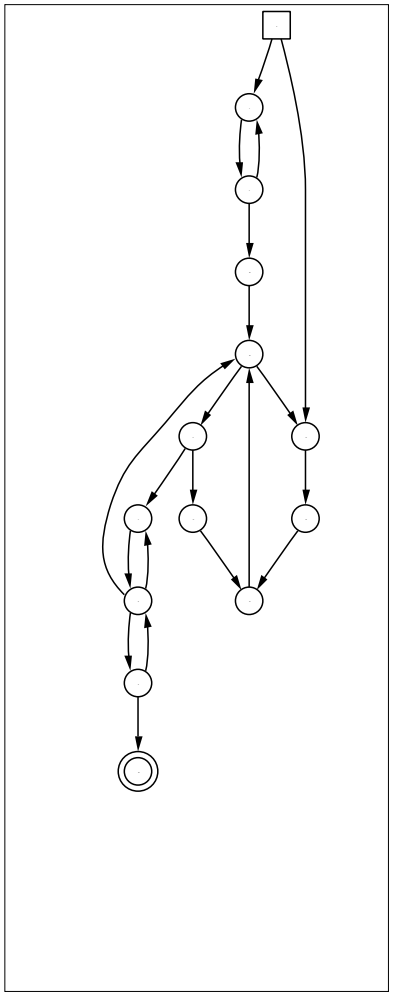
- Merging using “loose” and “strict” definitions of similarity

Similarity Merging In Action

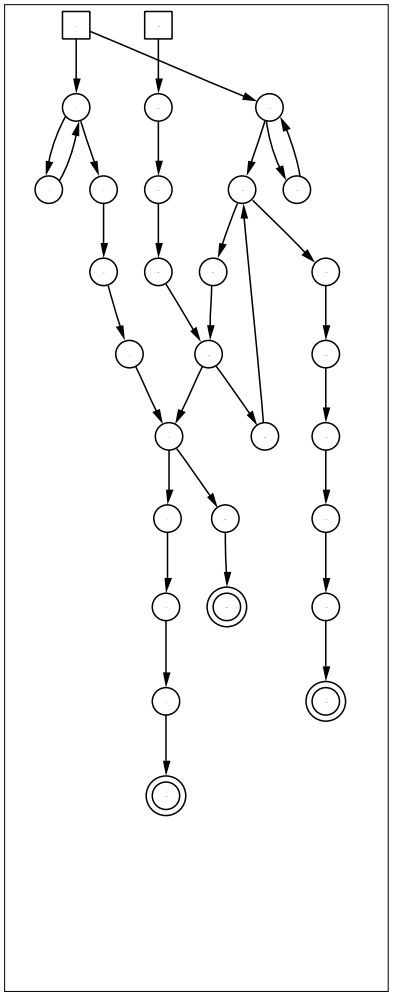
Original



Loose



Strict



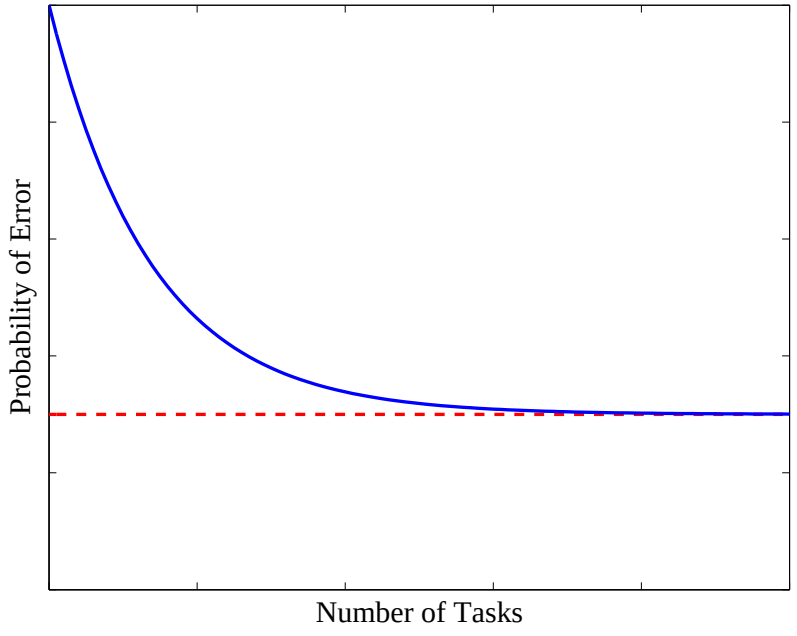
- Merging using “loose” and “strict” definitions of similarity

Properties of the Algorithm

- The algorithm only produces graphs where
  - All *similar* nodes are merged
  - All *dissimilar* nodes are unmerged
    - Implies locally minimal number of nodes, but not necessarily globally minimal
- The worst-case computational complexity is quadratic in the number of observations

Correctness

- The probability of error decreases exponentially as more tasks are added



- However, the theorem deals with estimating individual states, not a sequence of observations

Outline

- Background
- Modeling the User
- Predictive Robot Programming
- Hypothesizing about User Actions
- Learning By Observation
- Conclusions and Open Issues

Theory Meets Humanity

- Properties of the learning algorithm assume *ideal* conditions
- It remains to be seen how the algorithm works on *real-world* data
- Test in relative isolation with an application of Predictive Robot Programming

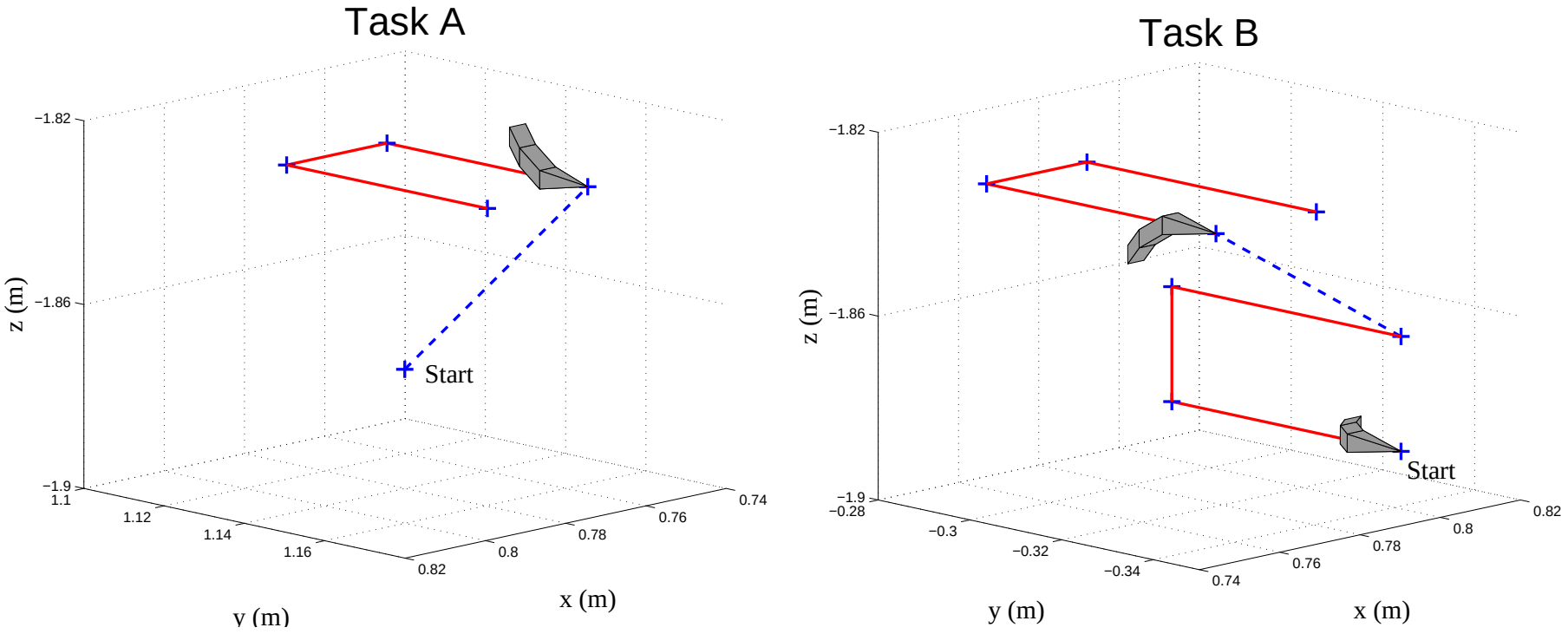
Predictive Robot Programming

- As capabilities increase, robots are performing more sophisticated tasks
- Simple tasks take days to program, complex tasks take weeks or months
- Significant programming time means that production may have to be halted temporarily
- Decreasing programming time will increase the appeal of robotic automation

Predictive Robot Programming

- Most tasks can be decomposed into simpler subtasks
- Subtasks may be repeated many times through the program
  - However, most robot programmers recreate the subtasks from scratch each time

Visualizing Similarity

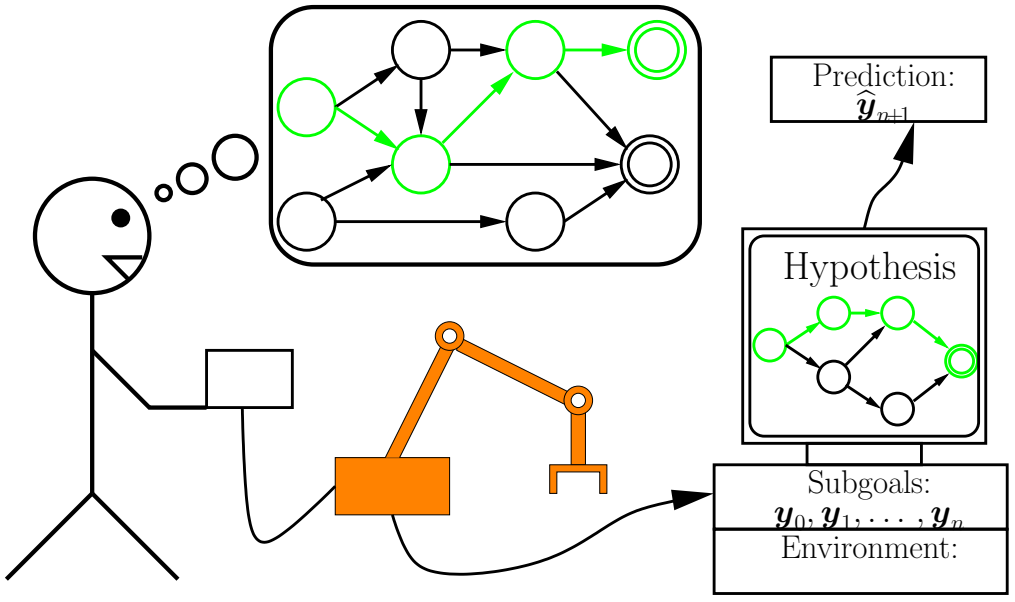


- Waypoints describe the location and orientation of the end effector
- Waypoints from two different subroutines
  - They are different, but contain a common pattern that is translated and rotated

Predictive Robot Programming

- Key idea behind Predictive Robot Programming:
  - Learn from previous user actions
  - Reduce programming time by identifying and completing subtasks automatically
- As an analogy, word-completion programs

Predictive Robot Programming

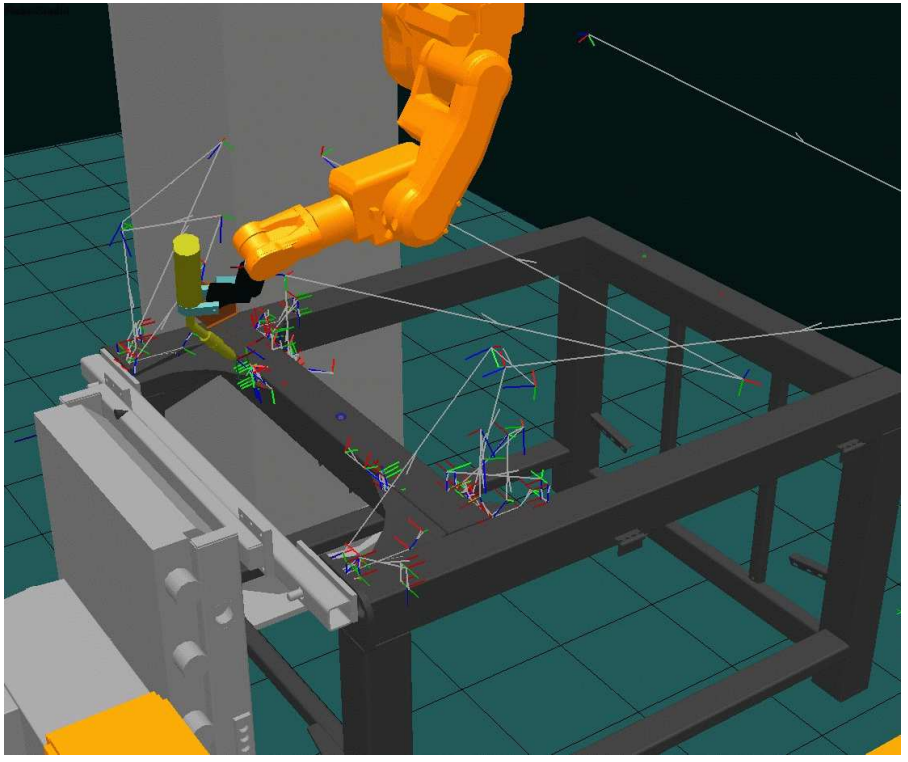


- User supplies subgoals
- Ignore environment
- System only predicts *next* subgoal (maximum likelihood)

Predictive Robot Programming

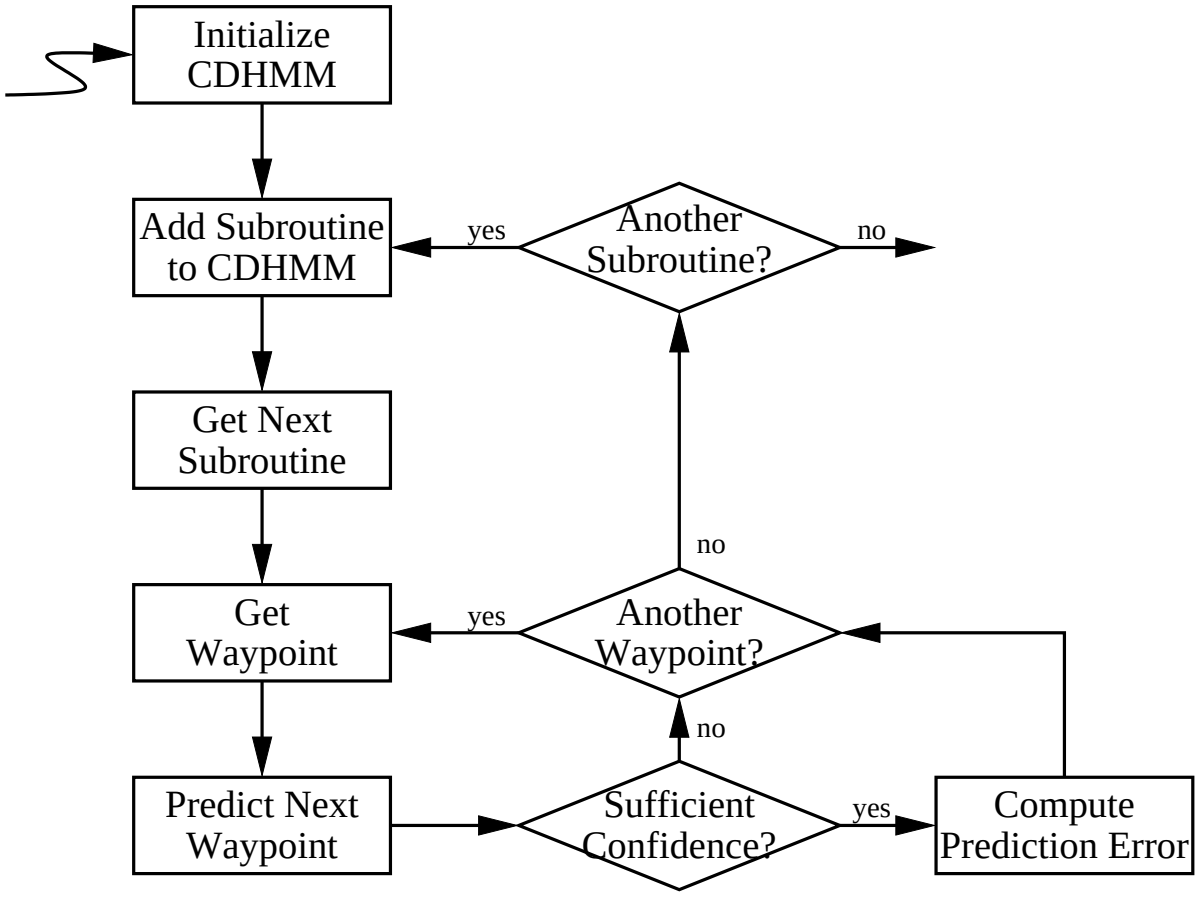
- We present two sets of results
  - Offline programming:
    - Showing prediction accuracy on real-world data
  - Online programming:
    - Showing decrease in programming time in laboratory

Offline-Programming Context



- 5 arc-welding programs, producing different products
- 252–1899 waypoints, 16–196 subroutines
- Took over 70 days to create by a professional robot programmer

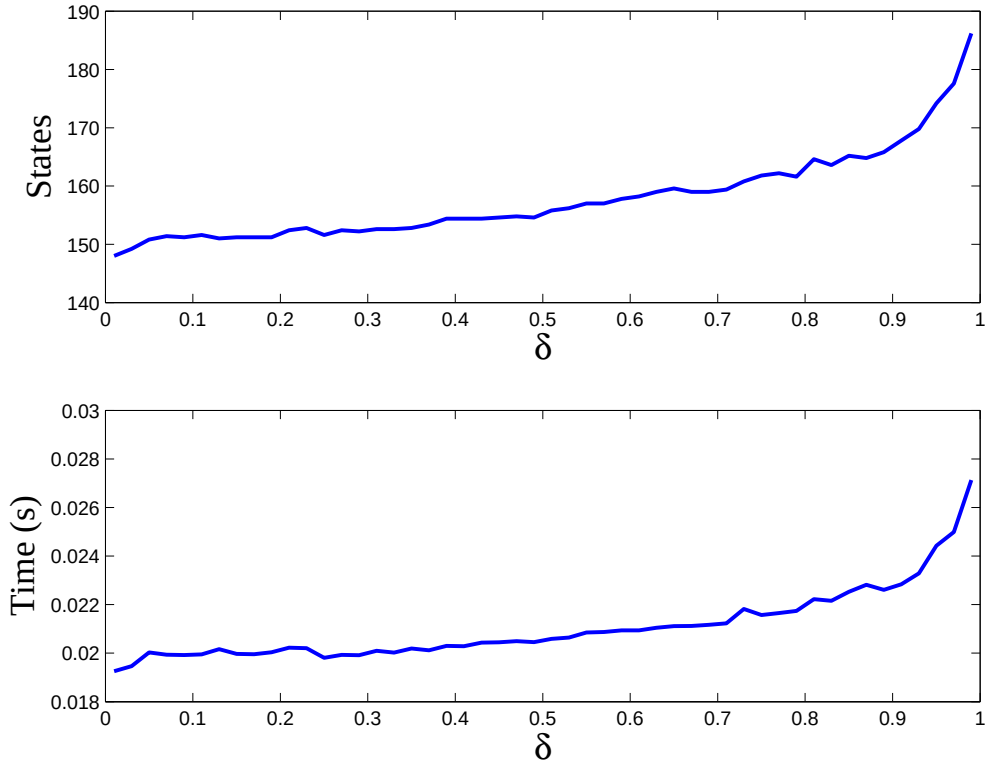
Offline-Programming Methodology



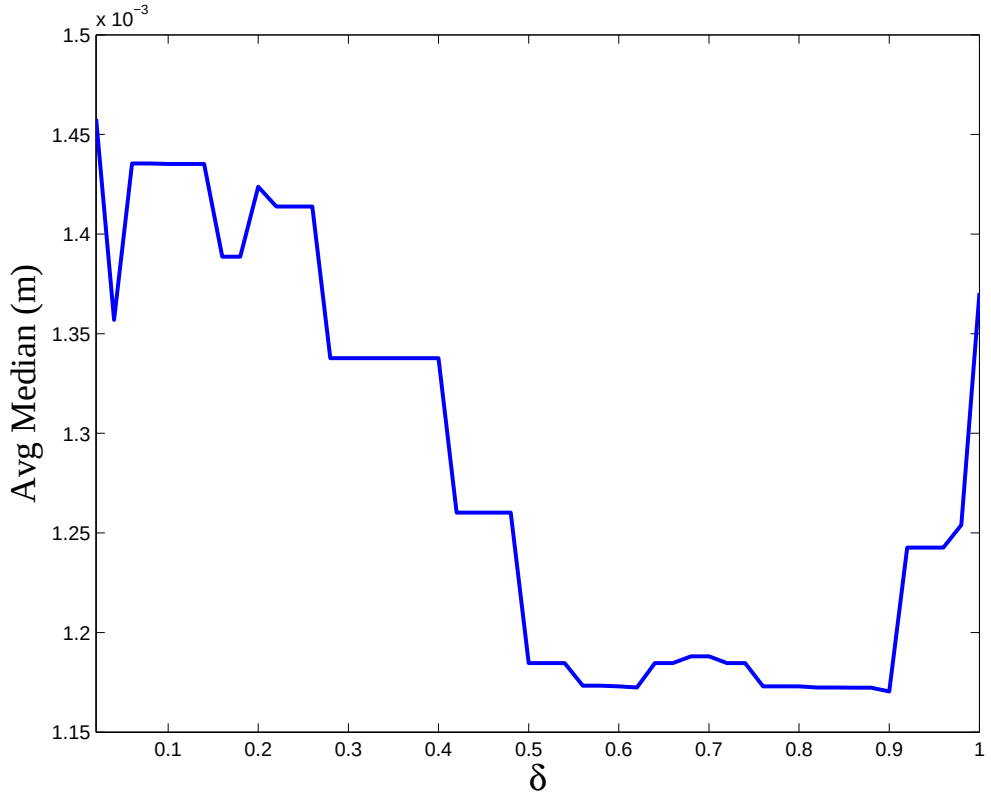
- Repeat this process for each robot program

Model Complexity

- Quantify similarity by $\delta \in (0, 1]$
  - $\delta \rightarrow 0$ induces simple CDHMMs
  - $\delta \rightarrow 1$ induces complex CDHMMs



Modeling the User



- When δ is too small, CDHMM is too simple
- When δ is too large, CDHMM overfits
- A value of $\delta \in (0.5, 0.9)$ induces the "right" complexity

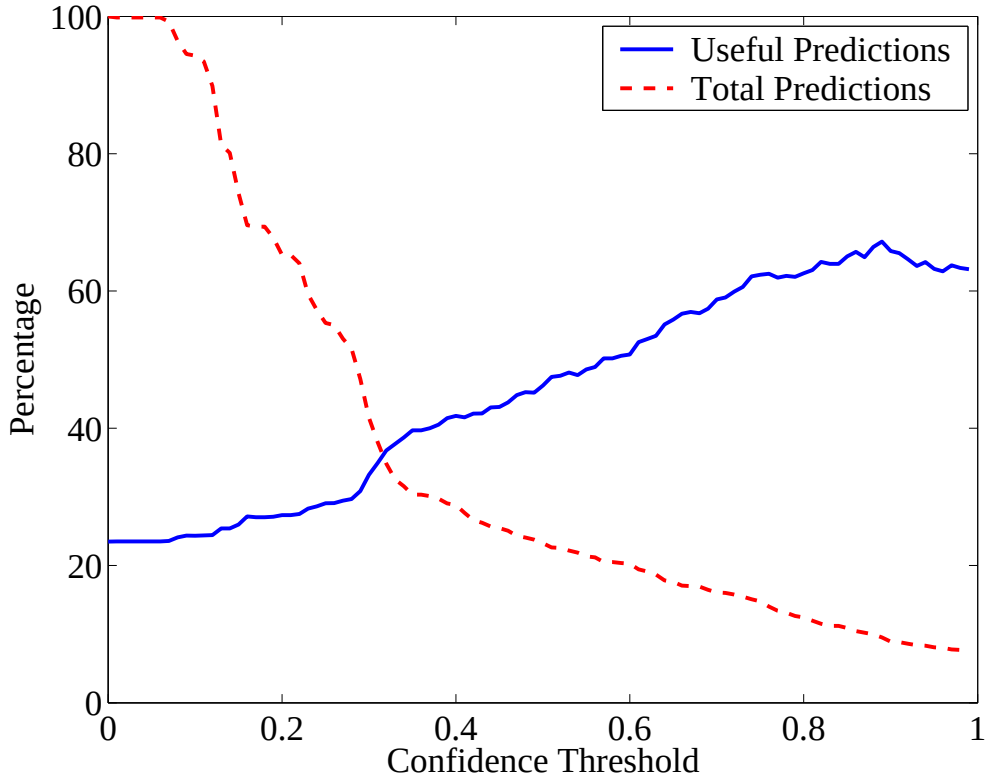
Prediction Confidence

- Regardless of criterion, prediction will always exist
- Should only suggest most accurate predictions
  - Need a *causal* statistic

Prediction Confidence

- Regardless of criterion, prediction will always exist
- Should only suggest most accurate predictions
  - Need a *causal* statistic
- Compute prediction confidence, $\phi_n \in [0, 1]$, based on CDHMM entropy from observing current task
  - High entropy results in low confidence
  - Low entropy results in high confidence
- Filter predictions based on confidence threshold

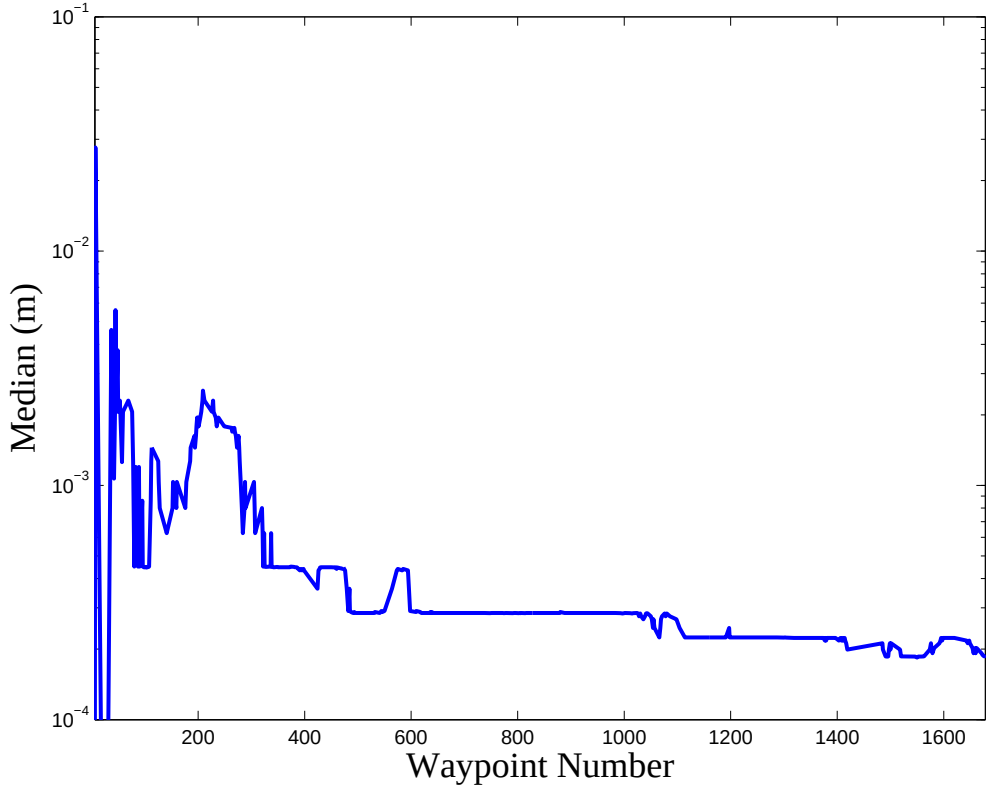
Prediction Confidence



- A useful prediction is within 1millimeter of the target
- Confidence correlates with prediction error as -0.89

  $(p \ll 0.01)$

Temporal Performance



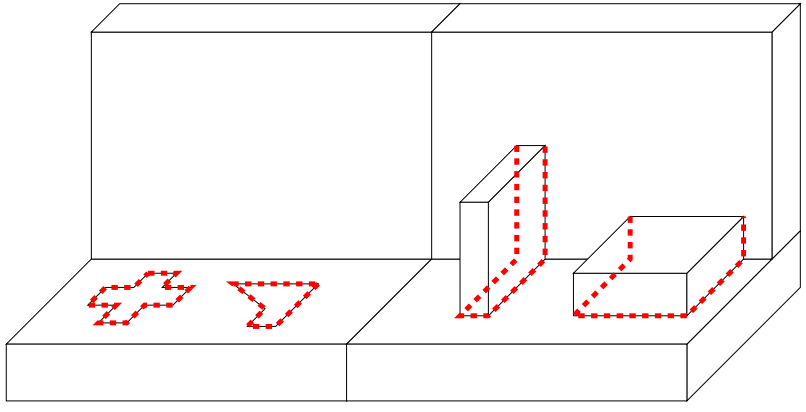
- Error generally decreases as more waypoints are added
  - median: 190microns, 0.38% Cartesian error

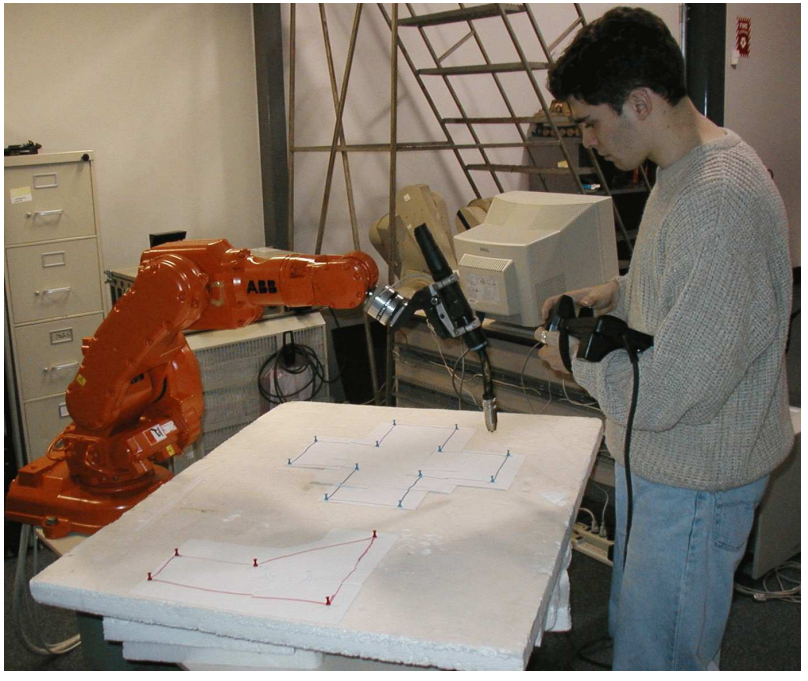
Offline-Programming Results

- Majority of predictions are *useful*
- Median errors:
  - Cartesian error 30–200 microns (0.03%–0.2%)
- Running time per waypoint
  - Construct CDHMM and predict: 30ms

Online-Programming Setup

- The ultimate criterion is reduction in programming time





- We collected 44 robot programs from 3 users in a laboratory setting

Online-Programming Results

Prediction Criterion	mean (sec)	std (sec)	change	Wilcoxon Conf
Baseline	292.2	78.61	N/A	N/A
$\phi_n \geq 0.8$	193.2	32.07	-33.88%	99.95%
$\phi_n \geq 0.5$	178.0	33.39	-39.08%	99.99%

- Programming time used to complete the tasks with
  - no predictions
  - high-confidence predictions
  - low-confidence predictions
- Statistically significant drop when using predictions
  - No statistical significance between prediction criteria

Limitations

- Indicating location of prediction
  - Collisions are a problem
- Monolithic CDHMM does not represent entire tasks well

Outline

- Background
- Modeling the User
- Predictive Robot Programming
- Hypothesizing about User Actions
- Learning By Observation
- Conclusions and Open Issues

Shelter from the Storm

- The algorithm appears to model user actions well
  - These were relatively sheltered conditions
- Inject more realistic factors

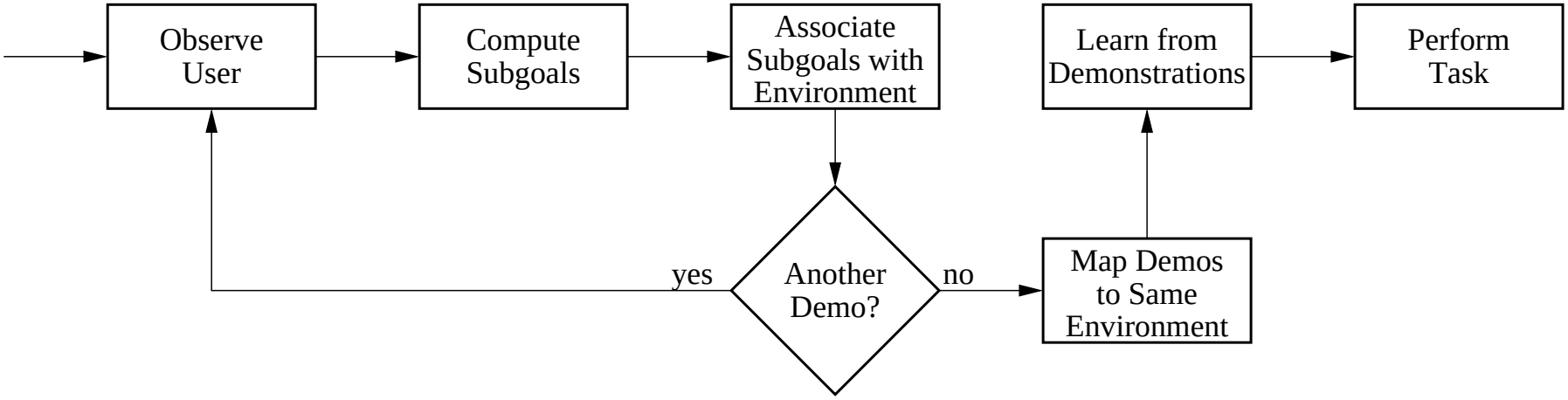
Learning By Observation

- Allow users to program mobile robots by demonstrating trajectory
- Consider a vacuum-cleaning robot
  - Undesirable to require retraining each time furniture is moved
- Create a system that automates motor-skill tasks, regardless of environment, occlusion, and noise

Learning By Observation

- Allow users to program mobile robots by demonstrating trajectory
- Consider a vacuum-cleaning robot
  - Undesirable to require retraining each time furniture is moved
- Create a system that automates motor-skill tasks, regardless of environment, occlusion, and noise
  - Within reason, of course

Methodology

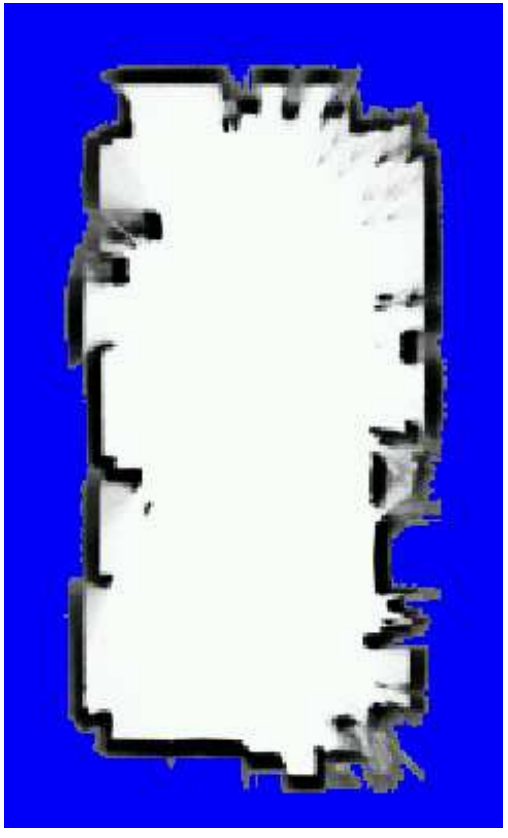


- In Predictive Robot Programming:
  - “Learn from Demos”
  - “Perform Task”
- Now we have added
  - Sensor issues
  - Extraction of subgoals
  - Environment considerations

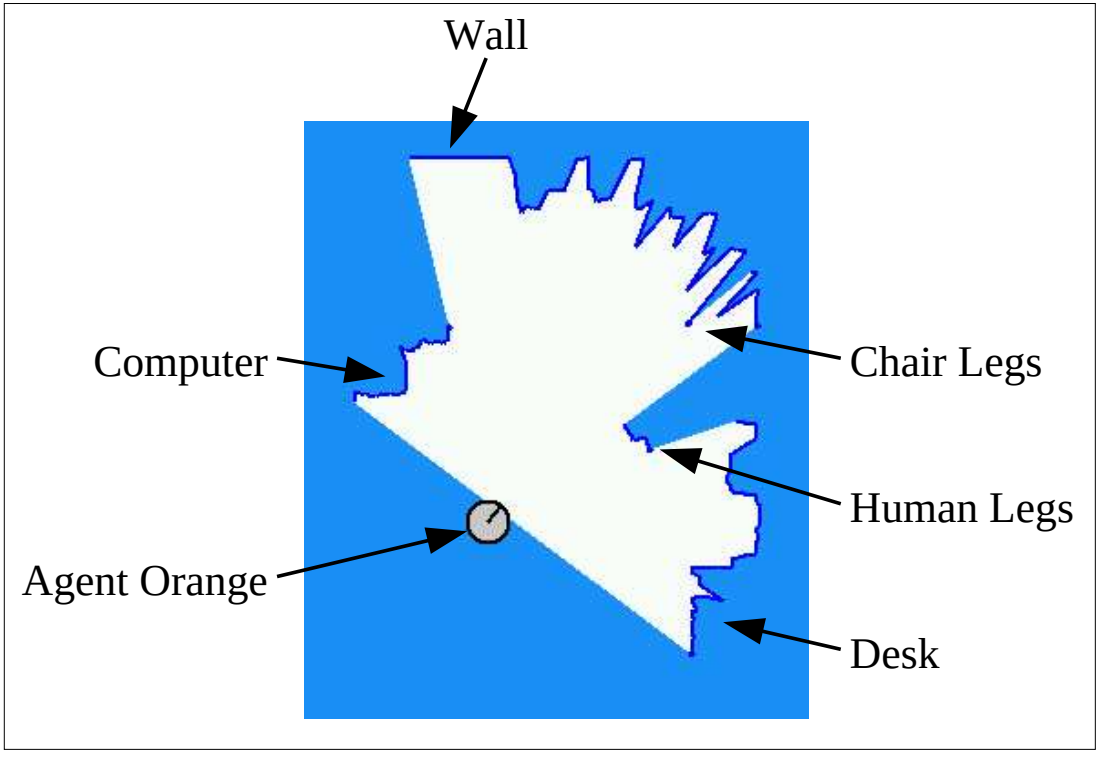
Agent Orange

- ActivMedia Pioneer DX II
- SICK scanning laser range finder
- Carmen toolkit (Montemerlo et al., 2003)

Observing the User



Map of laboratory



Sample laser scan

- Object occlusion handled by Kalman filters

Hypothesizing about User Actions

- Requirements:
  - Emulate “what the user would have done” in novel conditions
  - Ability to be mapped to different environments
  - Incorporate multiple examples
  - Facilitate learning
- A method for representing trajectories

Representing Trajectories

- There has been much work in representing trajectories
  - Cubic splines (Craig, 1989)
  - Bézier curves (Hwang et al., 2003)
  - Predicate calculus (Nicolescu & Matarić, 2001)
  - Nonlinear differential equations (Schaal et al., 2003)

Representing Trajectories

- There has been much work in representing trajectories
  - Cubic splines (Craig, 1989)
  - Bézier curves (Hwang et al., 2003)
  - Predicate calculus (Nicolescu & Matarić, 2001)
  - Nonlinear differential equations (Schaal et al., 2003)

- We want to store user trajectories in a generative manner

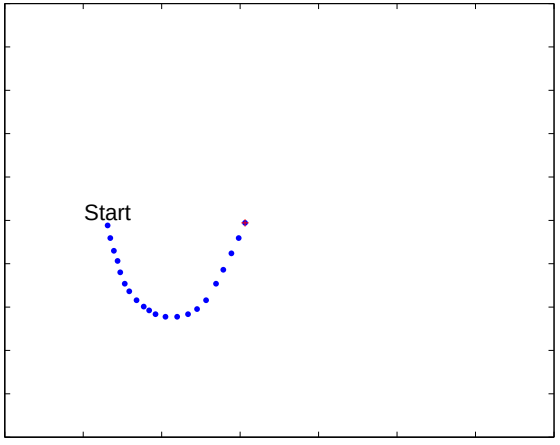
Representing Trajectories

- There has been much work in representing trajectories
  - Cubic splines (Craig, 1989)
  - Bézier curves (Hwang et al., 2003)
  - Predicate calculus (Nicolescu & Matarić, 2001)
  - Nonlinear differential equations (Schaal et al., 2003)
- We want to store user trajectories in a generative manner
- Our approach: Sequenced Linear Dynamical Systems (LDS)

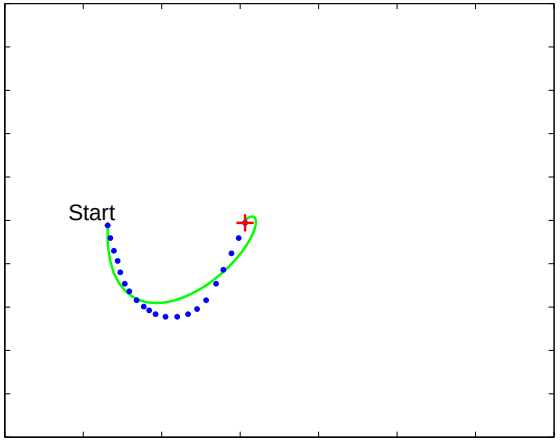
Sequenced LDS

- Segment trajectory at important points, the subgoals
- Represent each segment as a single Linear Dynamical System
- Reconstruct trajectory using the LDS estimates in sequence

LDS Example

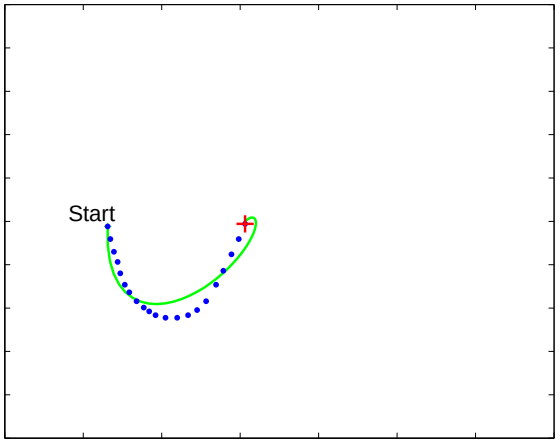


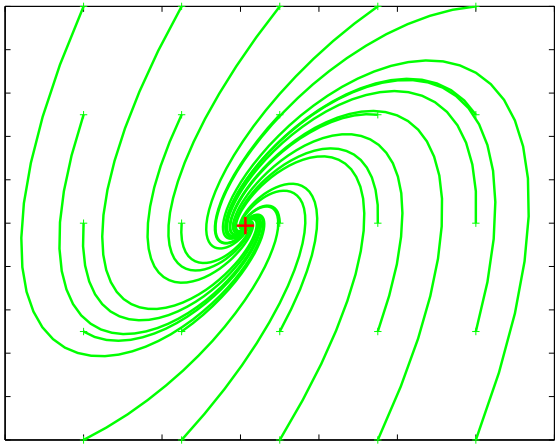
LDS Example



- Fit the trajectory with the least-squares LDS
- Reproduce the trajectory with the estimated LDS

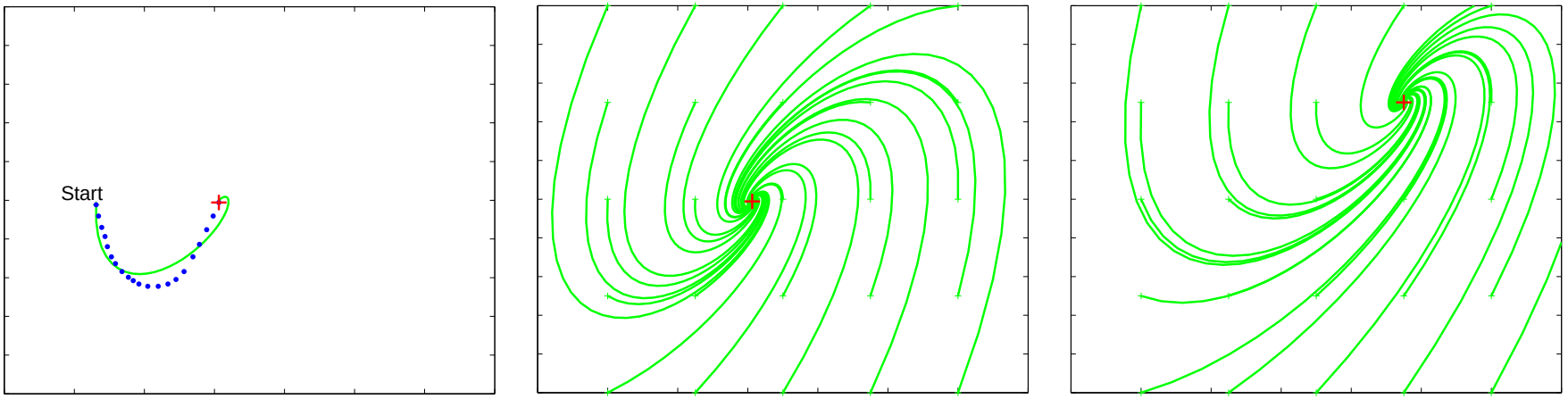
LDS Example





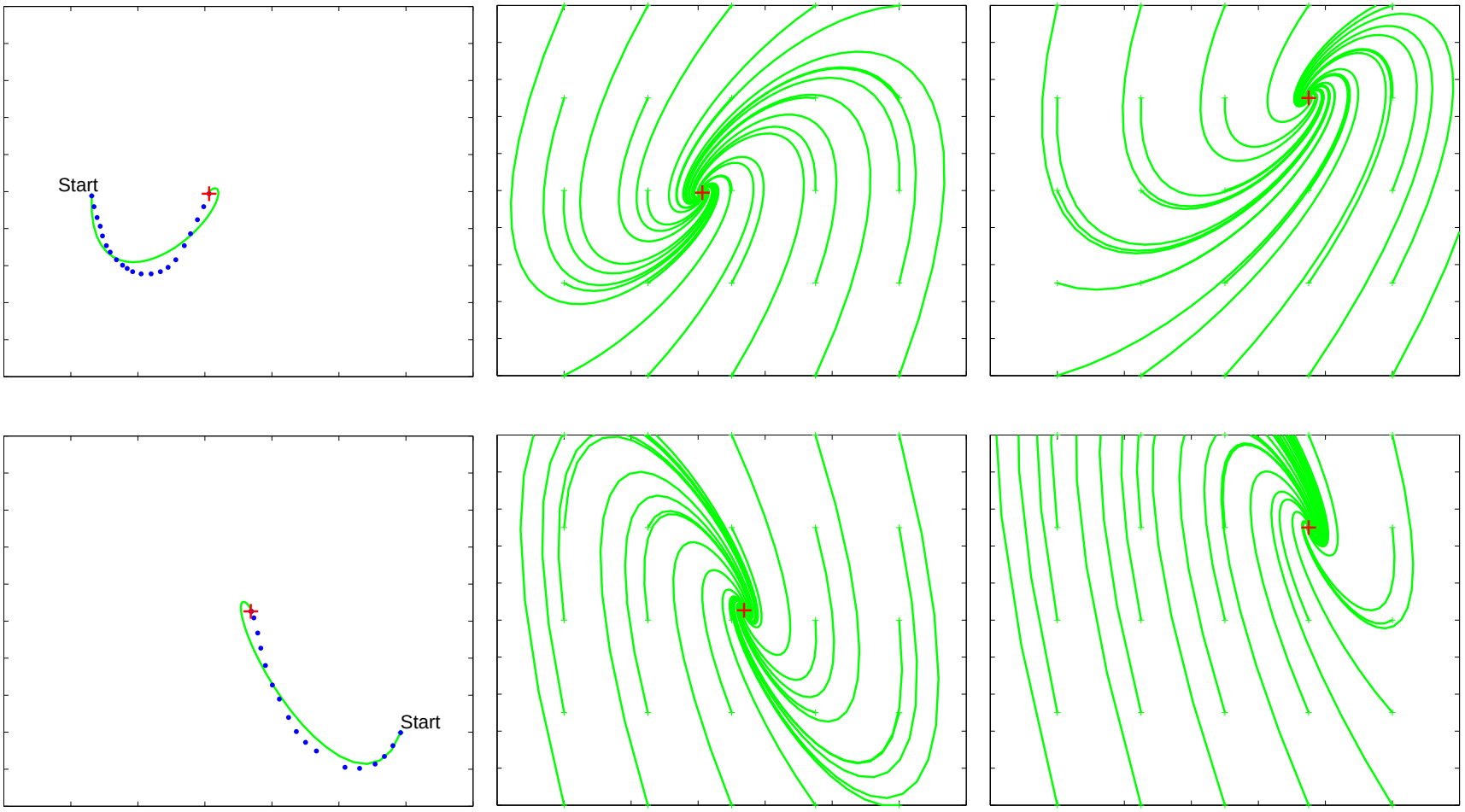
- Fit the trajectory with the least-squares LDS
- Reproduce the trajectory with the estimated LDS
- Response of LDS to various initial conditions represents a hypothesis of user intent

LDS Example



- Fit the trajectory with the least-squares LDS
- Reproduce the trajectory with the estimated LDS
- Response of LDS to various initial conditions represents a hypothesis of user intent
- Response of LDS to different subgoals represents a hypothesis of user intent

LDS Example



The Building Blocks

- Suppose we have a trajectory, $\boldsymbol{X} = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$
- Assume data are generated according to

$$\boldsymbol{x}_{n+1} = \boldsymbol{R}\ (\boldsymbol{x}_n - \boldsymbol{x}_N\ ) + \boldsymbol{x}_n$$

The Building Blocks

- Suppose we have a trajectory, $\boldsymbol{X} = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$
- Assume data are generated according to

$$\boldsymbol{x}_{n+1} = \boldsymbol{R}(\boldsymbol{x}_n - \boldsymbol{x}_N) + \boldsymbol{x}_n$$

- Captures direction, curvature, speed

The Building Blocks

- Suppose we have a trajectory, $\boldsymbol{X} = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$
- Assume data are generated according to

$$\boldsymbol{x}_{n+1} = \boldsymbol{R}\ (\boldsymbol{x}_n - \boldsymbol{x}_N) + \boldsymbol{x}_n$$

- Captures direction, curvature, speed
- Specifies trajectory terminus

LDS Estimation

- Assume subgoal is last point in trajectory segment

LDS Estimation

- Assume subgoal is last point in trajectory segment
- The least-squares solution to $\boldsymbol{R}$ is

$$\begin{aligned}\widehat{\boldsymbol{R}} &= \left([\boldsymbol{x}_1 \cdots \boldsymbol{x}_N] - [\boldsymbol{x}_0 \cdots \boldsymbol{x}_{N-1}]\right)\left([\boldsymbol{x}_0 \cdots \boldsymbol{x}_{N-1}] - [\boldsymbol{x}_N \cdots \boldsymbol{x}_N]\right)^{\mathsf{R}} \\ &\doteq \left(\boldsymbol{X}_{1:N} - \boldsymbol{X}_{0:N-1}\right)\left(\boldsymbol{X}_{0:N-1} - \boldsymbol{\Gamma}_N\right)^{\mathsf{R}}\end{aligned}$$

LDS Estimation

- Assume subgoal is last point in trajectory segment
- The least-squares solution to $\boldsymbol{R}$ is

$$\begin{aligned}\widehat{\boldsymbol{R}} &= \left([\boldsymbol{x}_1 \cdots \boldsymbol{x}_N] - [\boldsymbol{x}_0 \cdots \boldsymbol{x}_{N-1}]\right)\left([\boldsymbol{x}_0 \cdots \boldsymbol{x}_{N-1}] - [\boldsymbol{x}_N \cdots \boldsymbol{x}_N]\right)^{\mathsf{R}} \\ &\doteq \left(\boldsymbol{X}_{1:N} - \boldsymbol{X}_{0:N-1}\right)\left(\boldsymbol{X}_{0:N-1} - \boldsymbol{\Gamma}_N\right)^{\mathsf{R}}\end{aligned}$$

- Yes, it is a closed-form solution

Reconstructing Trajectories

- Use the induced control law with $\widehat{\boldsymbol{x}}_0 = \boldsymbol{x}_0$ and

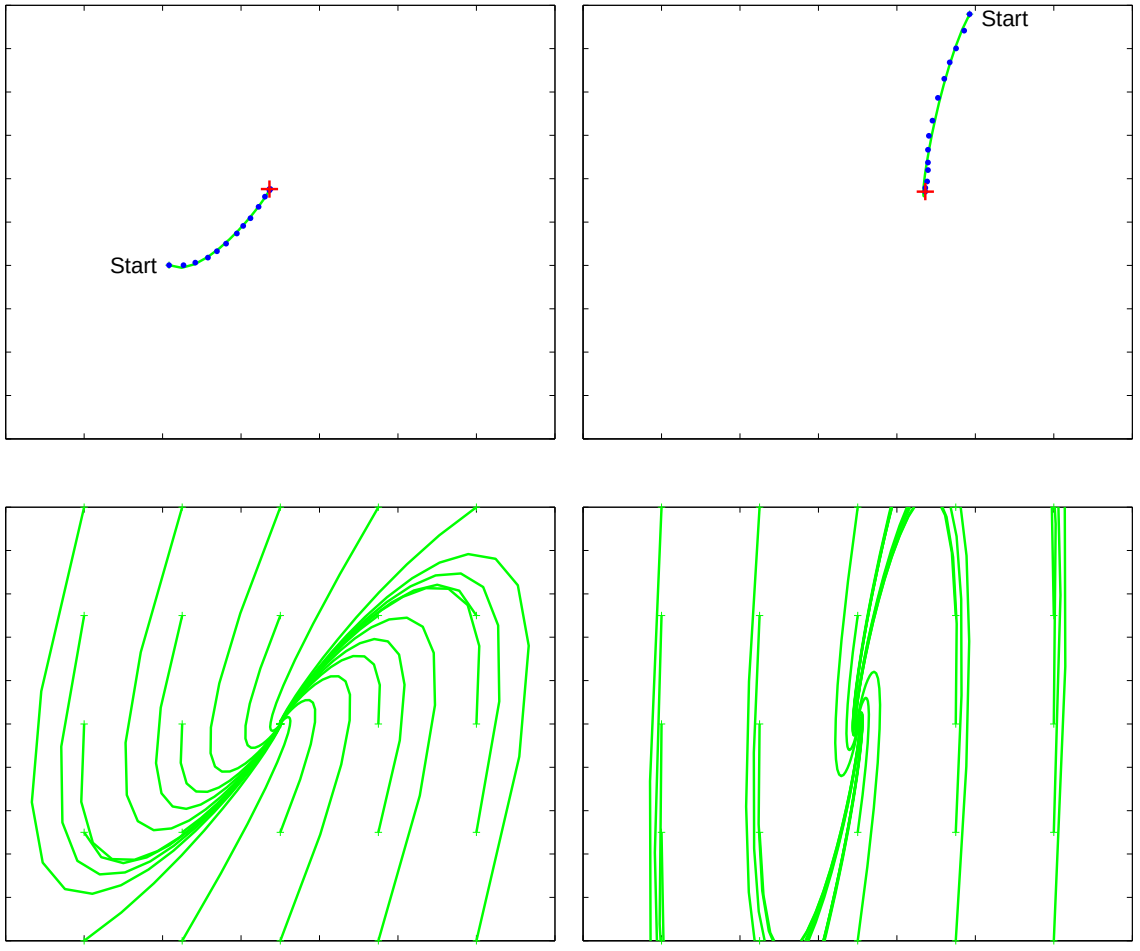
$$\widehat{\boldsymbol{x}}_{n+1} = \widehat{\boldsymbol{R}}\left(\widehat{\boldsymbol{x}}_n - \boldsymbol{x}_N\right) + \widehat{\boldsymbol{x}}_n$$

- We guarantee stability under reasonable conditions
  - Bounded trajectories
  - Terminate at the desired subgoal

Improving the Generalization

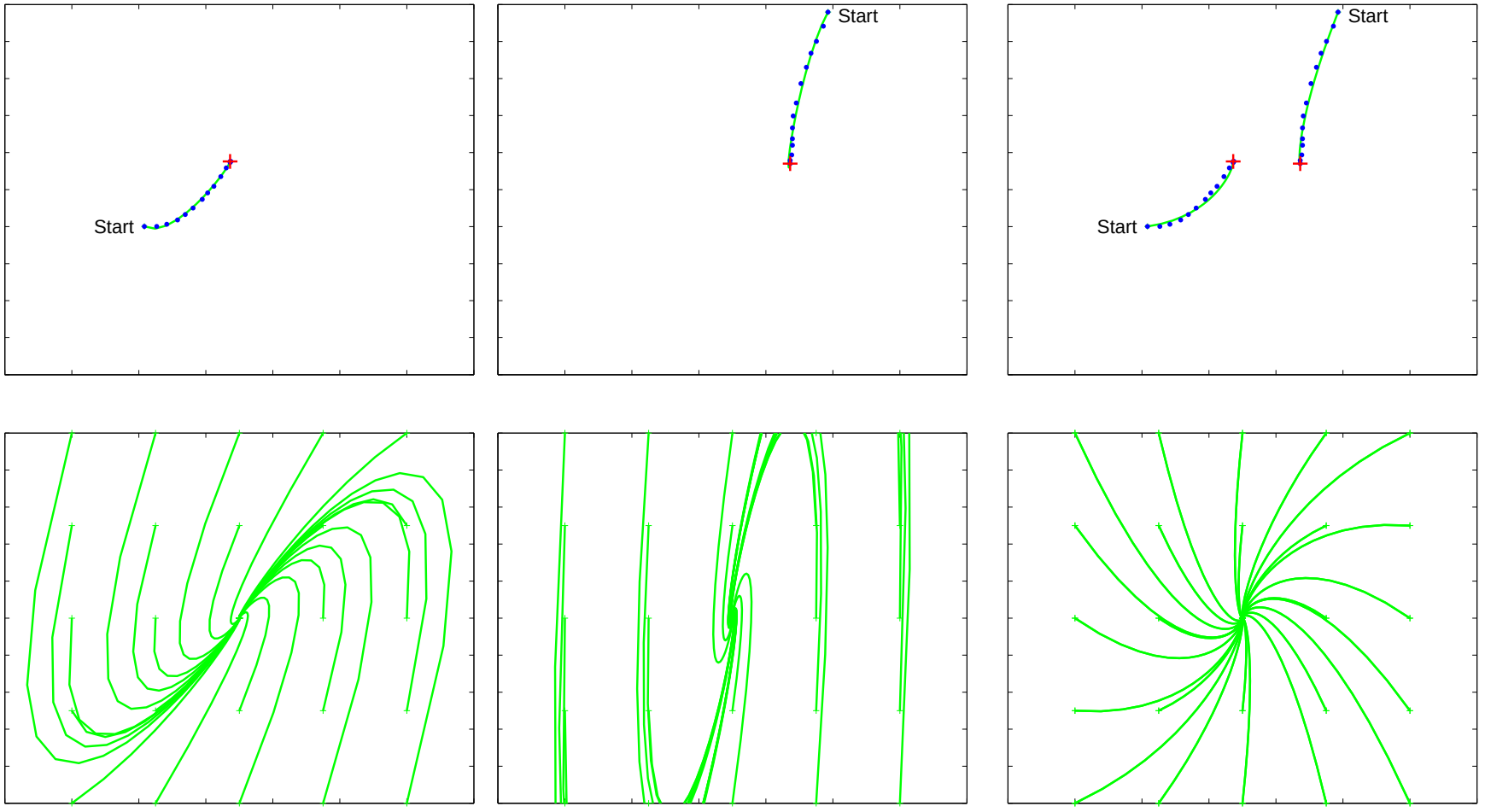
- Learning from more trajectories improves generalization

Improving the Generalization



- Learning from more trajectories improves generalization
- Two trajectories with “slight counter-clockwise curvature”

Improving the Generalization



- Learning from more trajectories improves generalization
- Two trajectories with “slight counter-clockwise curvature”

A Single LDS Is Not Enough

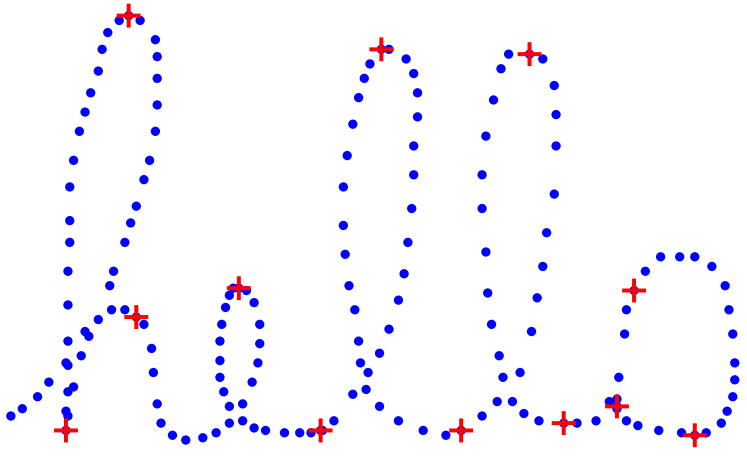
- A single LDS is an extremely compact representation
- But its simplicity is insufficient for LBO tasks

A Single LDS Is Not Enough

- A single LDS is an extremely compact representation
- But its simplicity is insufficient for LBO tasks
- Segment complicated trajectories based on predictability

A Single LDS Is Not Enough

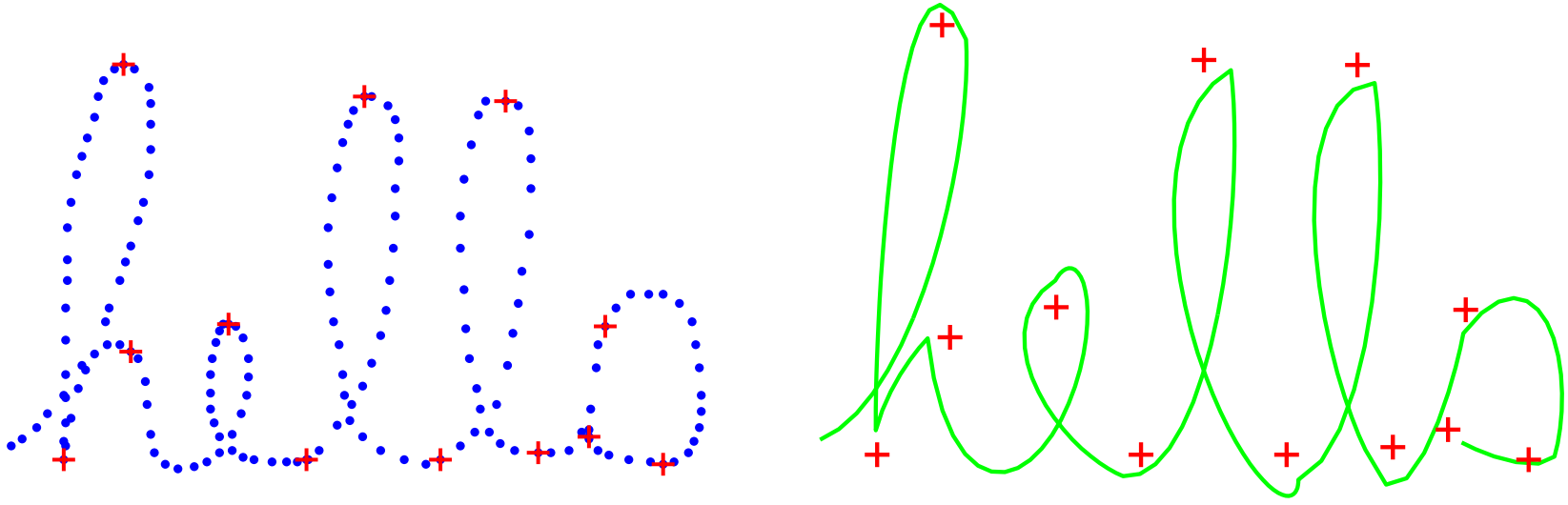
- Use LDS to predict next observation and segment trajectory at poorly predicted points



- These points mark subgoals in the trajectory

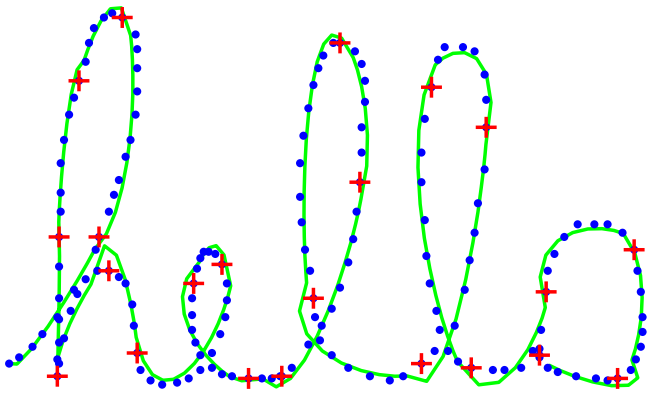
A Single LDS Is Not Enough

- Use LDS to predict next observation and segment trajectory at poorly predicted points

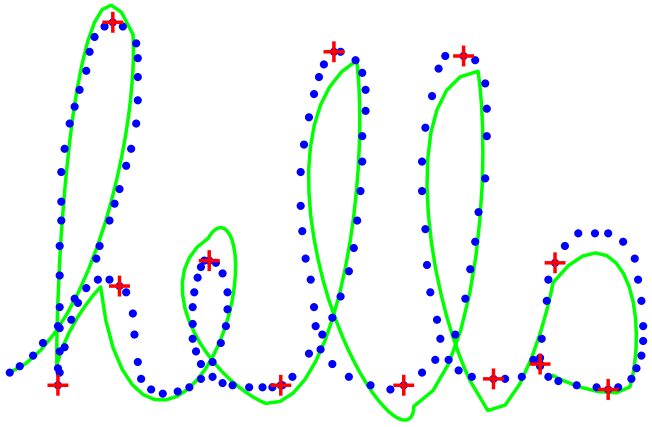


- These points mark subgoals in the trajectory

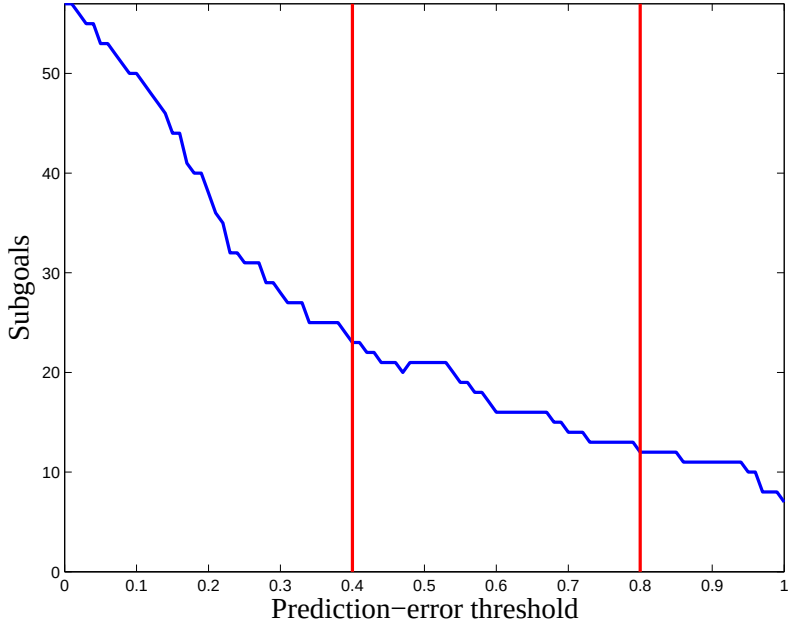
Trading Simplicity for Accuracy

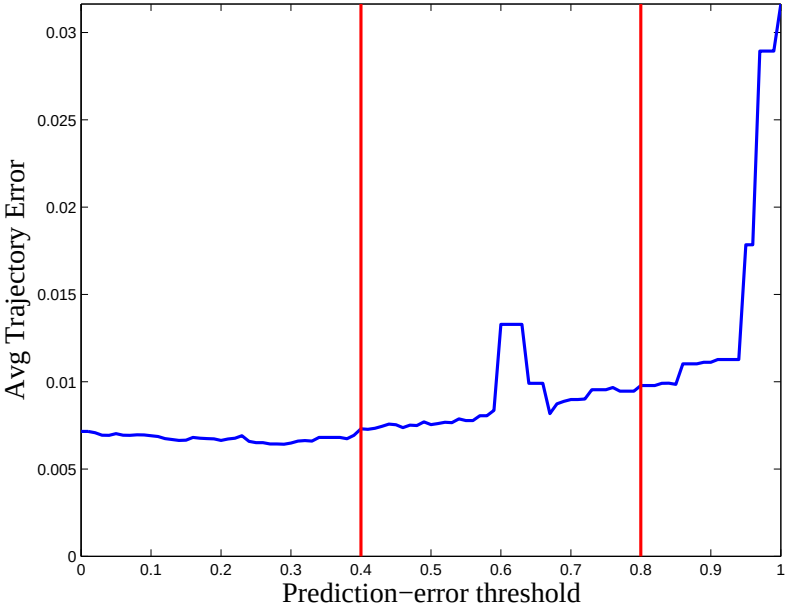


Prediction-error threshold = 0.4



Prediction-error threshold = 0.8





Electrical & Computer ENGINEERING

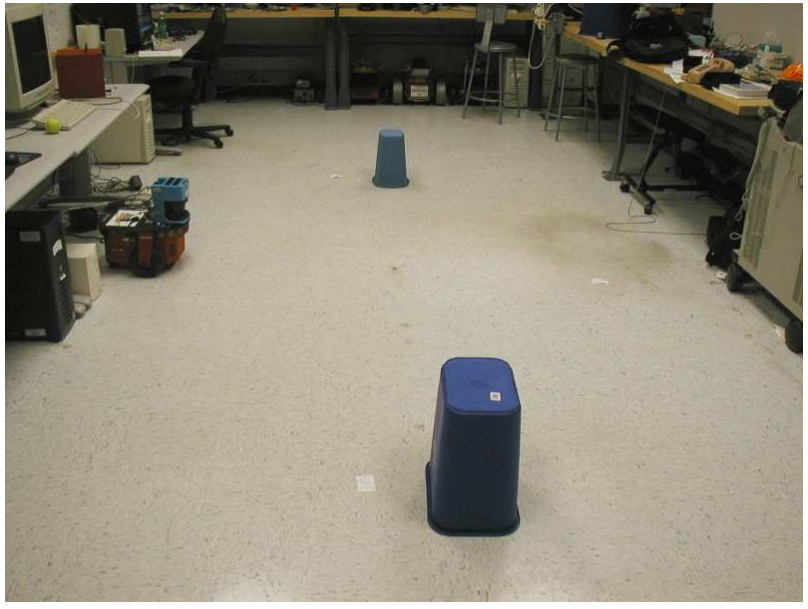
Outline

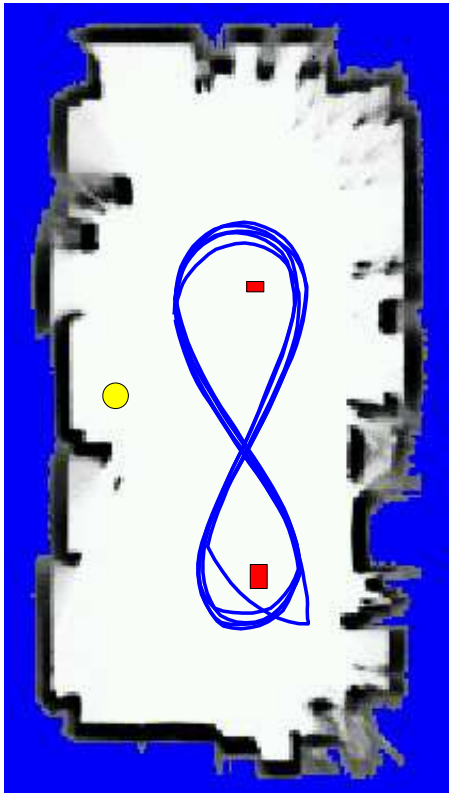
- Background
- Modeling the User
- Predictive Robot Programming
- Hypothesizing about User Actions
- Learning By Observation
- Conclusions and Open Issues

The Real World

- We now have a model of user actions
- Hypotheses of user actions to different conditions
- We apply these abilities to learning motor skills in mobile robots

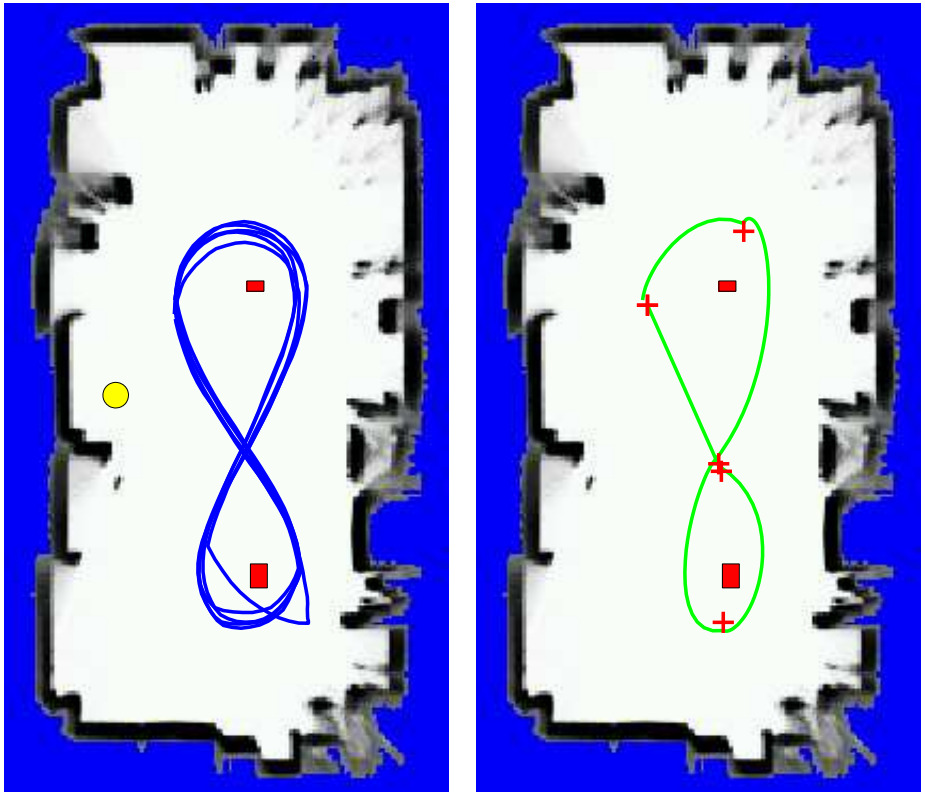
Observing the User





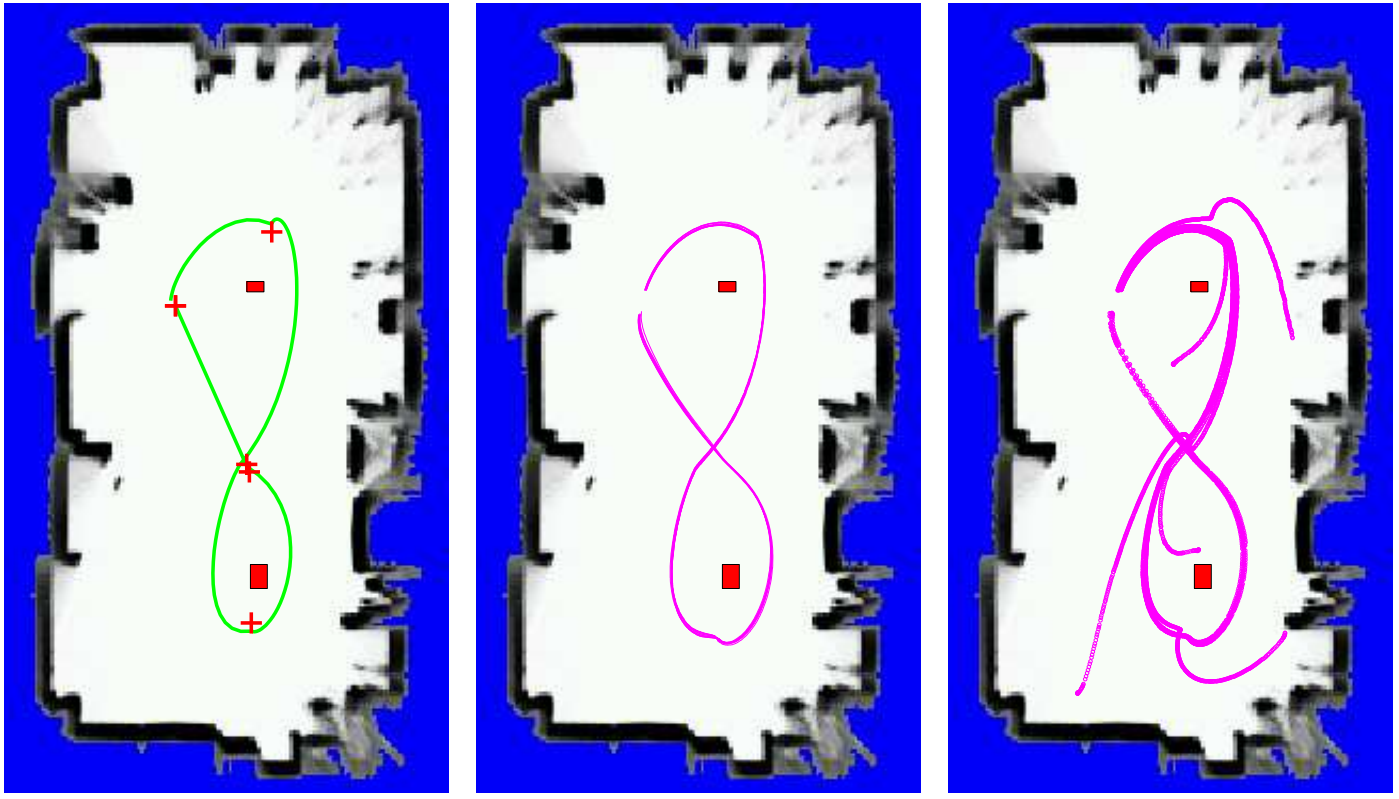
- Place slalom cones in the lab and track the user

Learning from the User



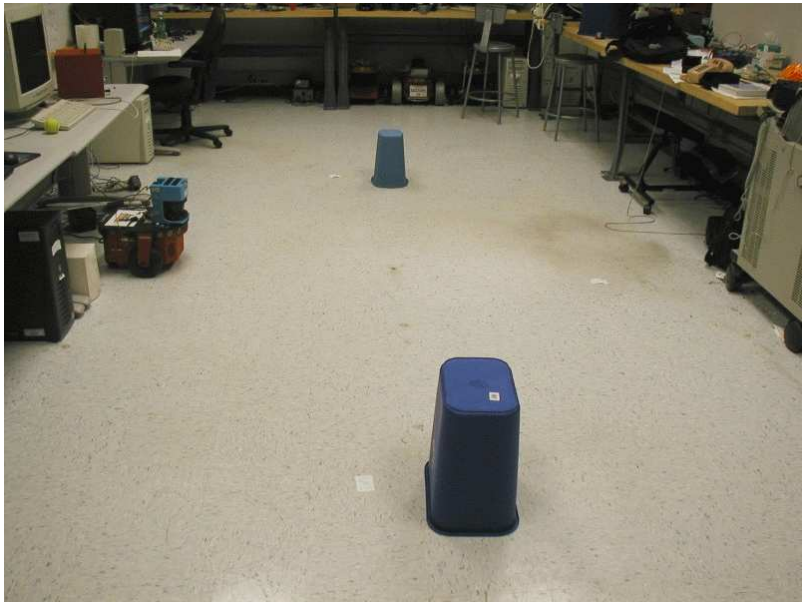
- Estimating the user trajectory
- Extracted 6 subgoals
- Average error of estimate is 20 millimeters

Learning from the User



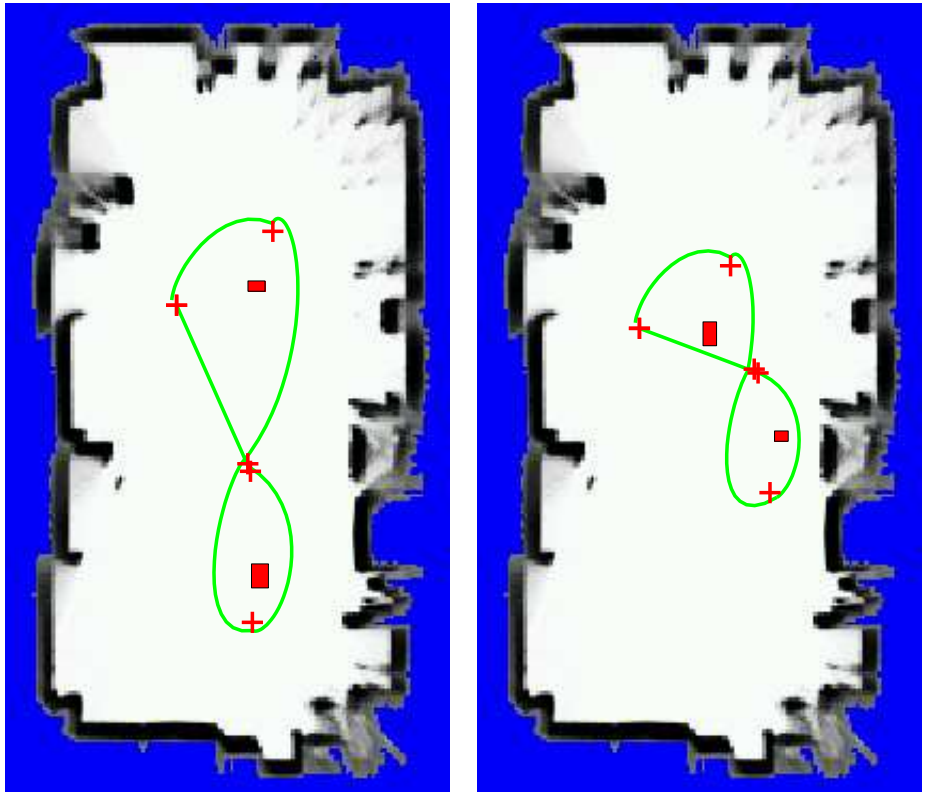
- 10 runs of the robot
- 5 runs where the robot was “kidnapped”

Environment Changes



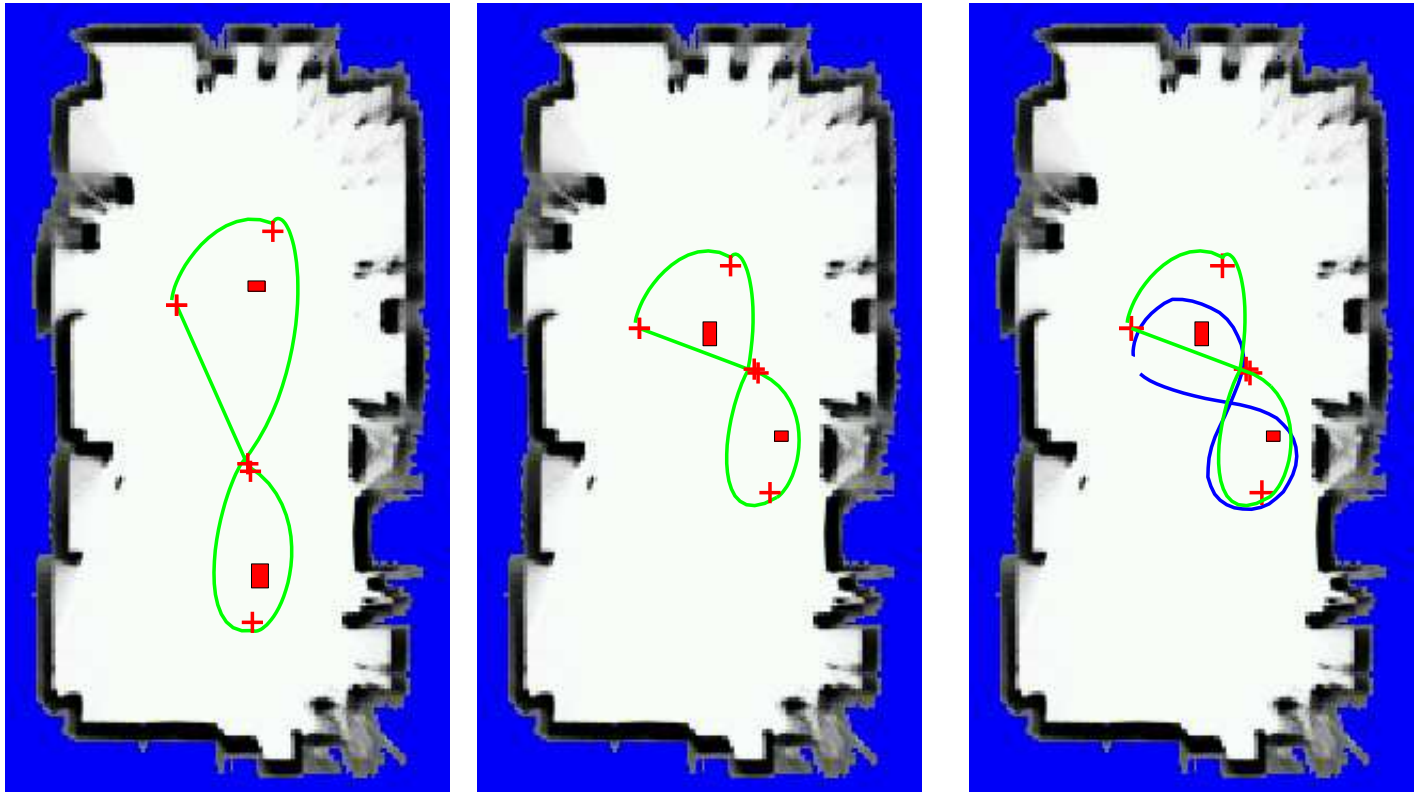
- "What would the user have done?"
- Automatically associate subgoals with objects in the environment
- LDS estimates automatically adjust their responses

Environment Changes



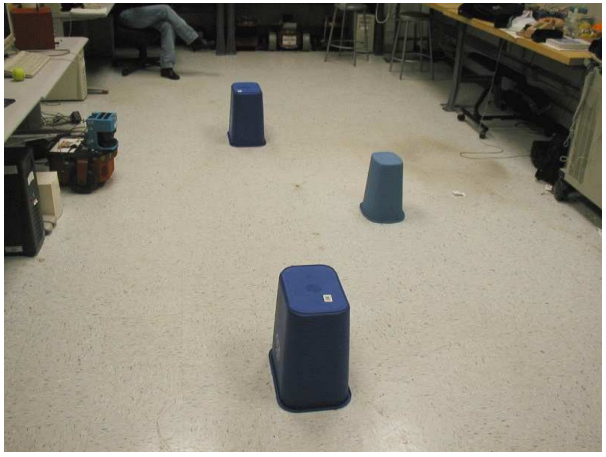
- “What would the user have done?”

Environment Changes



- "What would the user have done?"
- We asked the user to perform the same task
  - Average error of estimate is 200 millimeters

Learning in Different Environments

- Can demonstrations from these two environments help in performing in another?

Learning in Different Environments

- Learning from demonstrations in different environments

Learning in Different Environments

- Learning from demonstrations in different environments
  - First, map subgoals to the same environment

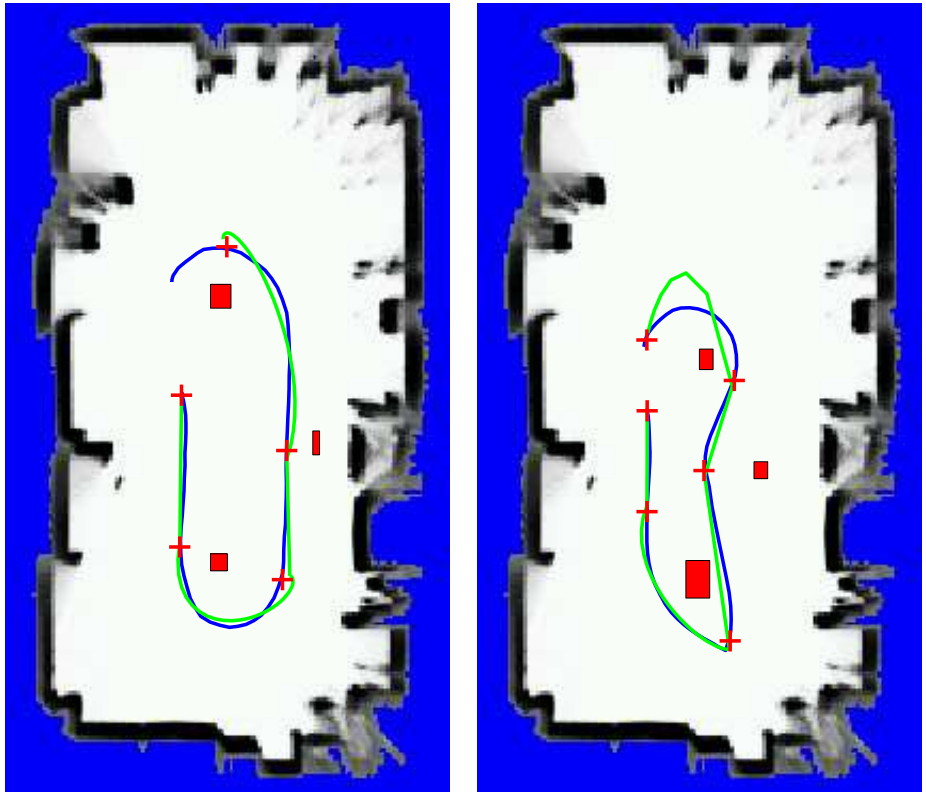
Learning in Different Environments

- Learning from demonstrations in different environments
  - First, map subgoals to the same environment
  - Construct CDHMM describing subgoals

Learning in Different Environments

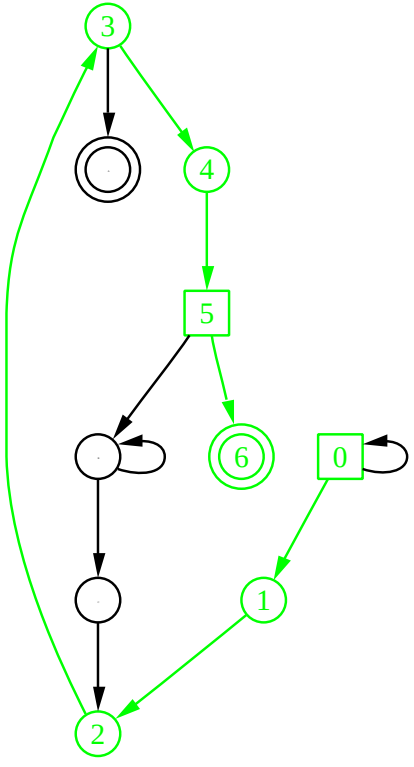
- Learning from demonstrations in different environments
  - First, map subgoals to the same environment
  - Construct CDHMM describing subgoals
  - Determine most-likely sequence of subgoals needed to complete task

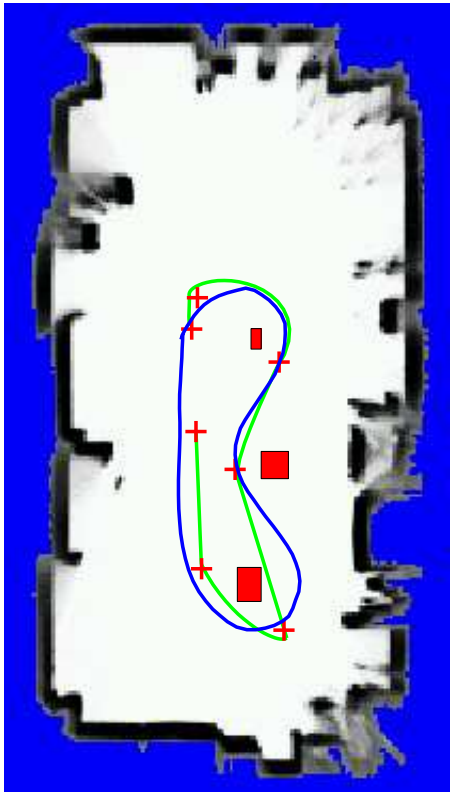
Learning in Different Environments



- Two demonstrations and corresponding subgoals

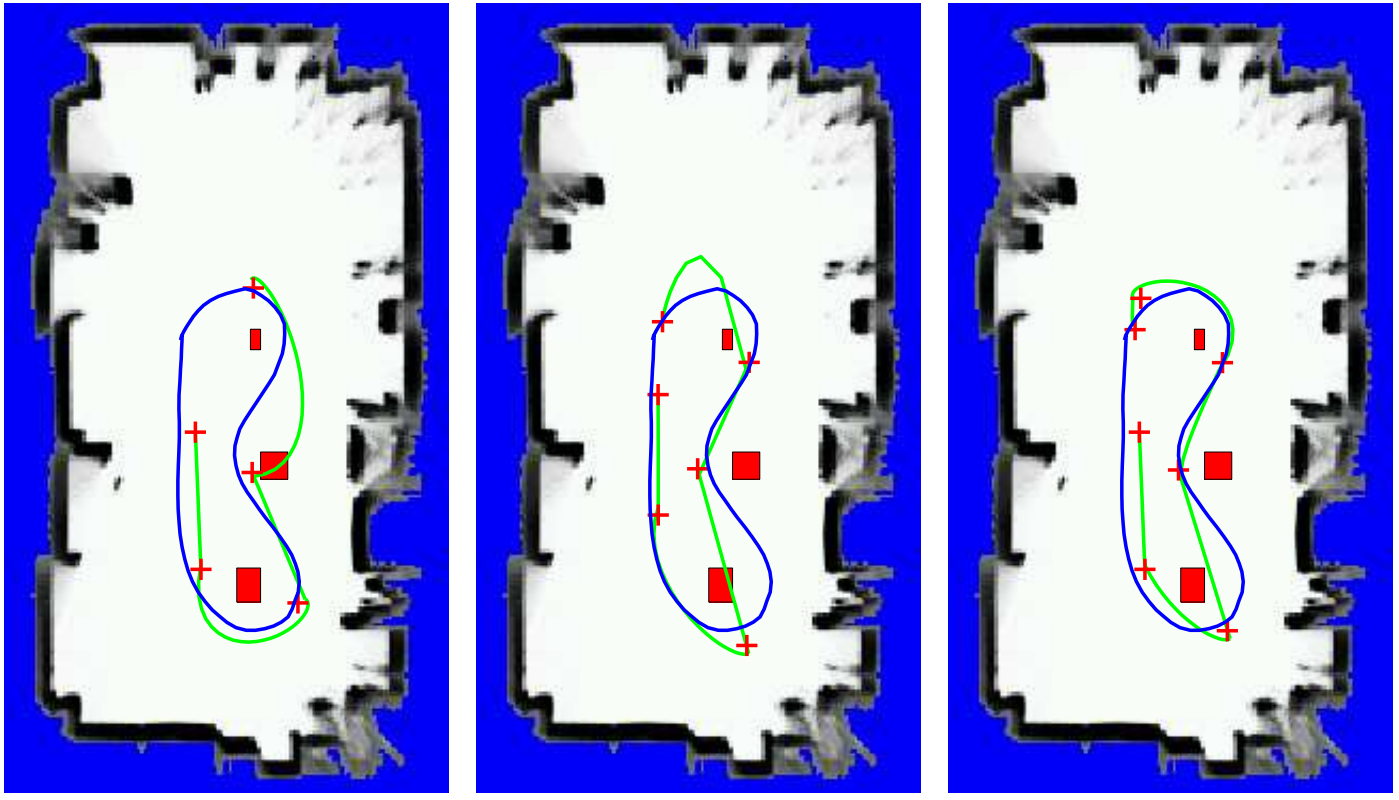
Learning in Different Environments





- CDHMM describing the subgoals with most-likely sequence shown in green

Learning in Different Environments



- Individually, the average error is 364 and 236 millimeters
- Together, the average error is 189 millimeters

LBO Discussion

- Computational approach appears viable
- Permits learning from
  - Multiple demonstrations
  - Demonstrations in different environments
- Environment mapping is weakest aspect
  - Hungarian Method requires bijective mapping

Outline

- Background
- Modeling the User
- Predictive Robot Programming
- Hypothesizing about User Actions
- Learning By Observation
- Conclusions and Open Issues

Conclusions and Contributions

- Developed algorithm that estimates the structure of CDHMMs efficiently
- Created PRP system that predicts waypoints in manipulator programs, based on previous observations, with high accuracy
- Developed LDS trajectory representation that forms hypotheses of user actions to different conditions
- Created mobile-robot LBO system that learns motor skills by observing user demonstrations

Open Issues / Ongoing Work

- Extending computational approach to include higher-level knowledge
  - Computational
  - Symbolic
  - Hybrid
- Incorporating semi-supervised (partially labeled) learning techniques to improve LBO and PRP
- Heuristic environment mapping methods

Thank You

Extra material

- References
- Material for clarification
- Theorems
- Proof sketches
- ...

References

Abe, N., & Warmuth, M. K. (1992). On the computational complexity of approximating probability distributions by probabilistic automata. *Machine Learning*, *9*.

Alissandrakis, A., Nehaniv, C. L., & Dautenhahn, K. (2002). Imitation with ALICE: Learning to imitate corresponding actions across dissimilar embodiments. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, *32*.

Asada, H., & Asari, Y. (1988). The direct teaching of tool manipulation skills via the impedance identifi cation of human motions. *Proceedings of the IEEE International Conference on Robotics and Automation*.

Billard, A., Epars, Y., Cheng, G., & Schaal, S. (2003). Discovering imitation strategies through categorization of multi-dimensional data. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Chen, J., & Zelinsky, A. (2003). Programming by demonstration: Coping with suboptimal teaching actions. *International Journal of Robotics Research*, *22*.

Craig, J. J. (1989). *Introduction to robotics: Mechanics and control*. Addison Wesley. Second edition.

Cypher, A. (Ed.). (1993). *Watch what I do: Programming by demonstration*. Cambridge, MA: MIT Press.

Ephraim, Y., & Merhav, N. (2002). Hidden Markov processes. *IEEE Transactions on Information Theory*, *48*.

Forsythe, C., & Xavier, P. G. (2002). Human emulation: Progress toward realistic synthetic human agents. *Proceedings of the 11th Conference on Computer-Generated Forces and Behavior Representation*.

Friedrich, H., Münch, S., Dillmann, R., Bocionek, S., & Sassin, M. (1996). Robot programming by demonstration (RPD): Supporting the induction by human interaction. *Machine Learning*, *23*, 163–189.

Gertz, M. W., Maxion, R. A., & Khosla, P. K. (1995). Visual programming and hypermedia implementation within a distributed laboratory environment. *Journal of Intelligent Automation and Soft Computing*.

Gillman, D., & Sipser, M. (1994). Inference and minimization of hidden Markov chains. *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory (COLT)*.

Hannaford, B., & Lee, P. (1991). Hidden Markov model analysis of force/torque information in telemanipulation. *International Journal of Robotics Research*, *10*.

Hovland, G., Sikka, P., & McCarragher, B. (1996). Skill acquisition from human demonstration using a hidden Markov model. *Proceedings of the IEEE International Conference on Robotics and Automation*.

Hwang, J.-H., Arkin, R. C., & Kwon, D.-S. (2003). Mobile robots at your fi ngertip: Bezier curve on-line trajectory generation for supervisory control. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Iba, S., Paredis, C. J., & Khosla, P. K. (2002). Interactive multi-modal

robot programming. *Proceedings of the IEEE International Conference on Robotics and Automation*.

Montemerlo, M., Roy, N., & Thrun, S. (2003). Perspectives on standardization in mobile robot programming : The Carnegie Mellon navigation (CARMEN) toolkit. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Nicolescu, M. N., & Matarić, M. J. (2001). Learning and interacting in human-robot domains. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, *31*.

Papadimitriou, C. H., & Steiglitz, K. (1998). *Combinatorial optimization: Algorithms and complexity*. Mineola, New York: Dover Publications. Second edition.

Pomerleau, D. (1991). Effi cient training of artifi cial neural networks for autonomous navigation. *Neural Copmutation*, *3*, 88–97.

Pomerleau, D. (1996). Neural network vision for robot driving. *Early Visual Learning*. Oxford University Press.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*, 257–286.

Ron, D., Singer, Y., & Tishby, N. (1998). On the learnability and usage of acyclic probabilistic fi nite automata. *Journal of Computer and System Sciences*, *56*.

Schaal, S., Ijspeert, A., & Billard, A. (2003). Computational approaches to motor learning by imitation. *Philosophical Transaction of the Royal Society of London: Series B, Bilogical Sciences*, *358*, 537–547.

Singh, R., Raj, B., & Stern, R. M. (2002). Automatic generation of subword units for speech recognition systems. *IEEE Transactions on Speech and Audio Processing*, *10*, 89–99.

Skubic, M., & Volz, R. A. (2000). Acquiring robust, force-based assembly skills from human demonstration. *IEEE Transactions on Robotics and Automation*, *16*.

Stolcke, A., & Omohundro, S. (1994). Inducing probabilistic grammars by Bayesian model merging. *International Conference on Grammatical Inference*.

Algorithm Learn-Structure

Algorithm Learn-Structure

$\boldsymbol{X} = \{\boldsymbol{X}^0, \boldsymbol{X}^1, \ldots, \boldsymbol{X}^M\}$ is the multiset of all observation sequences.
$\epsilon \geq 0$ is the similarity threshold.

```
1:  V := ∅, E := ∅
2:  G_X := (V, V^0, E, X, V, f, g)
3:  for all X^i ∈ {X^0, X^1, ..., X^M}
4:    for all x_n ∈ {x_0^i, x_1^i, ..., x_{N_i}^i}
5:      ε_min := min_{v_i ∈ V} μ_C(V_{v_i} ∪ {x_n}, ⟨V_{v_i} ∪ {x_n}⟩)
6:      if ε_min ≤ ε then
7:        v_new := arg min_{v_i ∈ V} μ_C(V_{v_i} ∪ {x_n}, ⟨V_{v_i} ∪ {x_n}⟩)
8:        V_{v_new} := V_{v_new} ∪ {x_n}
9:      end if
10:     else if ε_min > ε then
11:       create empty node v_new
12:       V := V ∪ {v_new}
13:       V_{v_new} := {x_n}
14:       g_{v_new} := 0
15:       if n > 0 then
16:         E := E ⋃ {e_{prev→new}}
17:         f_{e_{prev→new}} := 0
18:       end if
19:     end else if
20:     if n > 0 then
21:       f_{e_{prev→new}} := f_{e_{prev→new}} + 1
22:     end if
23:     else if n = 0 then
24:       g_{v_new} := g_{v_new} + 1
25:     end else if
26:     v_prev := v_new
27:   end for all
28: end for all
```

Electrical & Computer ENGINEERING

Similarity-Measure Merging

- Measure of similarity is defined to be

$$\mu_{\boldsymbol{C}}(\boldsymbol{\mathcal{V}}_{v_k}, \langle \boldsymbol{\mathcal{V}}_{v_k} \rangle) \triangleq \sum_{\boldsymbol{x} \in \boldsymbol{\mathcal{V}}_{v_k}} \|\boldsymbol{x} - \langle \boldsymbol{\mathcal{V}}_{v_k} \rangle\|_{\boldsymbol{C}}^2$$

- Two nodes are considered *similar* if

$$\mu_{\boldsymbol{C}}(\boldsymbol{\mathcal{V}}_{v_i} \cup \boldsymbol{\mathcal{V}}_{v_j}, \langle \boldsymbol{\mathcal{V}}_{v_i} \cup \boldsymbol{\mathcal{V}}_{v_j} \rangle) \leq \epsilon$$

Compactness

Definition 1. *An observation graph is* ϵ-compact *with respect to* μ *if, for any depth* n*, all nodes* $v_i \in V^n$ *and* $v_j \in V^n$

- $\mu_C(\boldsymbol{\mathcal{V}}_{v_i}, \langle \boldsymbol{\mathcal{V}}_{v_i} \rangle) \leq \epsilon$;
- $\epsilon < \mu_C(\boldsymbol{\mathcal{V}}_{v_i} \cup \boldsymbol{\mathcal{V}}_{v_j}, \langle \boldsymbol{\mathcal{V}}_{v_i} \cup \boldsymbol{\mathcal{V}}_{v_j} \rangle)$,

for some $\epsilon \geq 0$.

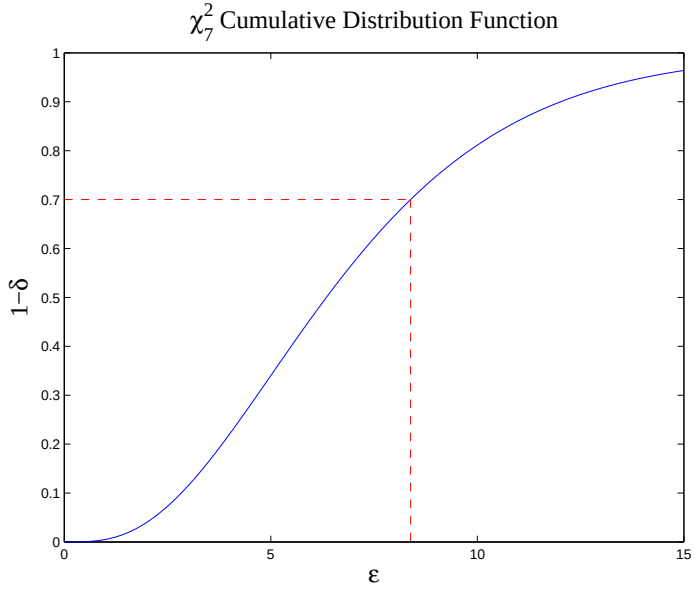
Theorem 1. `Learn-Structure` *only produces* ϵ*-compact observation graphs.*

- Proof uses $\exists \widetilde{\boldsymbol{\mathcal{A}}} \subseteq \boldsymbol{\mathcal{A}}$ s.t.
  $\tau < \mu_C(\widetilde{\boldsymbol{\mathcal{A}}}, \langle \widetilde{\boldsymbol{\mathcal{A}}} \rangle) \iff \tau < \mu_C(\boldsymbol{\mathcal{A}}, \langle \boldsymbol{\mathcal{A}} \rangle)$.

Model Complexity

- Similarity threshold ϵ does not have a real-world analogue
  - Define a "complexity" parameter, $\delta \in (0, 1]$

$$\Pr\left\{ \|\boldsymbol{x} - \langle \boldsymbol{\mathcal{V}}_{v_i} \rangle\|^2_{\boldsymbol{\Sigma}^{-1}} \leq \epsilon \right\} \quad > \quad 1 - \delta$$



Correctness

Theorem 2. *Let $\boldsymbol{\lambda}$ be a CDHMM converted by one-shot estimation from* `Learn-Structure` *with $\epsilon \geq 0$ and M* iid *tasks generated by some source CDHMM $\boldsymbol{\lambda}^*$. If the state $q_* \in \boldsymbol{\lambda}^*$ generates observations with finite first and second moments,*

$$\Pr\left\{\boldsymbol{\lambda} \xrightarrow{\gamma} q_* \in \boldsymbol{\lambda}^*\right\} \;>\; \left(1-(1-p_*)^M\right)\left(1-\frac{\operatorname{tr}[\boldsymbol{C}\operatorname{Var}(\boldsymbol{x}|q_*,\boldsymbol{\lambda}^*)]}{(\gamma-\sqrt{\epsilon})^2}\right),$$

where $\gamma > \sqrt{\epsilon}$, $p_* = \mathrm{P}(c_n = q_* | \boldsymbol{\lambda}^*) > 0$, and $\operatorname{Var}(\boldsymbol{x}|q_*)$ is the observation-generation variance from state q_*.

- Proof uses a multivariate Chebyshev's Inequality

$$\varepsilon^2 \Pr\left\{\|\boldsymbol{x}-\boldsymbol{y}\|_{\boldsymbol{C}} \geq \varepsilon\right\} \;\leq\; \operatorname{tr}[\boldsymbol{C}\operatorname{Var}(\boldsymbol{x})] + \|\boldsymbol{y}-\mathrm{E}\{\boldsymbol{x}\}\|_{\boldsymbol{C}}^2$$

Correctness

$$\Pr\left\{\boldsymbol{\lambda} \xrightarrow{\gamma} q_* \in \boldsymbol{\lambda}^*\right\} \quad > \quad \left(1-(1-p_*)^M\right)\left(1-\frac{\mathrm{tr}[\boldsymbol{C}\mathrm{Var}(\boldsymbol{x}|q_*,\boldsymbol{\lambda}^*)]}{(\gamma-\sqrt{\epsilon})^2}\right)$$

Correctness

$$\Pr\left\{\boldsymbol{\lambda} \xrightarrow{\gamma} q_* \in \boldsymbol{\lambda}^*\right\} > \left(1-(1-p_*)^M\right)\left(1-\frac{\mathrm{tr}[\boldsymbol{C}\mathrm{Var}(\boldsymbol{x}|q_*,\boldsymbol{\lambda}^*)]}{(\gamma-\sqrt{\epsilon})^2}\right)$$

- A state in the estimated CDHMM is "within" γ of a state in the target CDHMM

Correctness

$$\Pr\left\{\boldsymbol{\lambda} \xrightarrow{\gamma} q_* \in \boldsymbol{\lambda}^*\right\} > \left(1-(1-p_*)^M\right)\left(1-\frac{\mathrm{tr}[\boldsymbol{C}\mathrm{Var}(\boldsymbol{x}|q_*,\boldsymbol{\lambda}^*)]}{(\gamma-\sqrt{\epsilon})^2}\right)$$

- A state in the estimated CDHMM is "within" γ of a state in the target CDHMM
- Increases exponentially as more tasks are observed

Correctness

$$\Pr\left\{\boldsymbol{\lambda} \xrightarrow{\gamma} q_* \in \boldsymbol{\lambda}^*\right\} \quad > \quad \left(1-(1-p_*)^M\right)\left(1-\frac{\mathrm{tr}[\boldsymbol{C}\mathrm{Var}(\boldsymbol{x}|q_*,\boldsymbol{\lambda}^*)]}{(\gamma-\sqrt{\epsilon})^2}\right)$$

- A state in the estimated CDHMM is "within" γ of a state in the target CDHMM
- Increases exponentially as more tasks are observed
- Asymptotes to an "intrinsic error" based on the variability of the target CDHMM

Prediction Confidence

- Compute prediction confidence, $\phi_n \in [0, 1]$

$$\begin{aligned} \phi_n &\triangleq \frac{\mathcal{D}_{\mathrm{KL}}(c_n \parallel \frac{1}{|\mathcal{Q}|})}{\log_2 |\mathcal{Q}|} \\ &= 1 - \frac{H(c_n)}{\log_2 |\mathcal{Q}|}, \end{aligned}$$

- c_n is the current-state random variable conditioned on observing the current task
- $|\mathcal{Q}|$ is the number of states in the CDHMM

Prediction

- We use Maximum Likelihood estimators

$$\widehat{\boldsymbol{x}}_n^* = \arg\max_{\boldsymbol{x}_n} \sum_j b_j(n) \qquad \nu_n(j)$$

Prediction

- We use Maximum Likelihood estimators

$$\widehat{\boldsymbol{x}}_n^* = \arg\max_{\boldsymbol{x}_n} \sum_j b_j(n) \quad \nu_n(j)$$

- Likelihood state j generates $\boldsymbol{x}_n$

Prediction

- We use Maximum Likelihood estimators

$$\widehat{\boldsymbol{x}}_n^* = \arg\max_{\boldsymbol{x}_n} \sum_j b_j(n) \sum_i \nu_n(j)$$

- Likelihood state j generates $\boldsymbol{x}_n$
- Probability that state j generates *next* observation

Proof of Stability

Let the estimated LDS be

$$\begin{aligned}\widehat{\boldsymbol{x}}_{n+1} &= \widehat{\boldsymbol{R}}\left(\widehat{\boldsymbol{x}}_n - \boldsymbol{x}_N\right) + \widehat{\boldsymbol{x}}_n \\ &= \left(\widehat{\boldsymbol{R}} + \boldsymbol{I}\right)\widehat{\boldsymbol{x}}_n - \widehat{\boldsymbol{R}}\boldsymbol{x}_N\end{aligned}$$

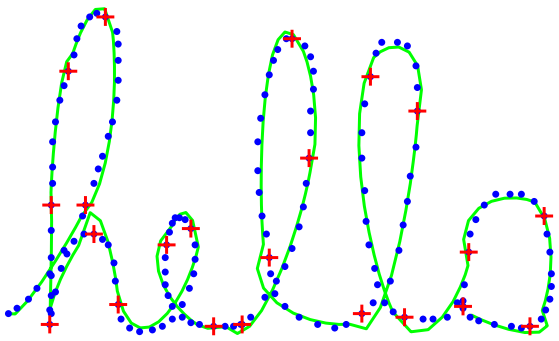
where

$$\begin{aligned}\widehat{\boldsymbol{R}} &= \left([\boldsymbol{x}_1 \cdots \boldsymbol{x}_N] - [\boldsymbol{x}_0 \cdots \boldsymbol{x}_{N-1}]\right)\left([\boldsymbol{x}_0 \cdots \boldsymbol{x}_{N-1}] - [\boldsymbol{x}_N \cdots \boldsymbol{x}_N]\right)^{\mathsf{R}} \\ &\doteq \left(\boldsymbol{X}_{1:N} - \boldsymbol{X}_{0:N-1}\right)\left(\boldsymbol{X}_{0:N-1} - \boldsymbol{\Gamma}_N\right)^{\mathsf{R}}\end{aligned}$$

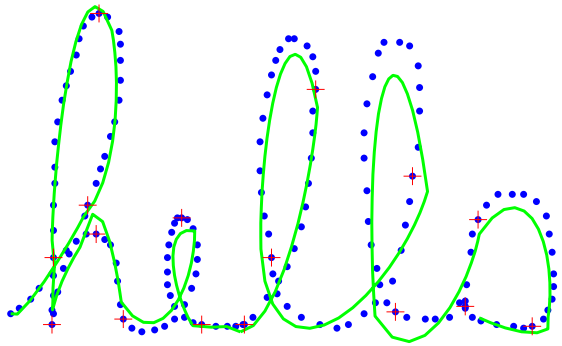
Theorem 3. *The estimated LDS is stable in the sense of Lyapunov about the equilibrium point $\boldsymbol{x}_N$ if the matrix $\left(\boldsymbol{X}^0_{N-1:-}\boldsymbol{\Gamma}_N\right)$ has full row rank.*

- Proof uses the discrete algebraic Lyapunov equation

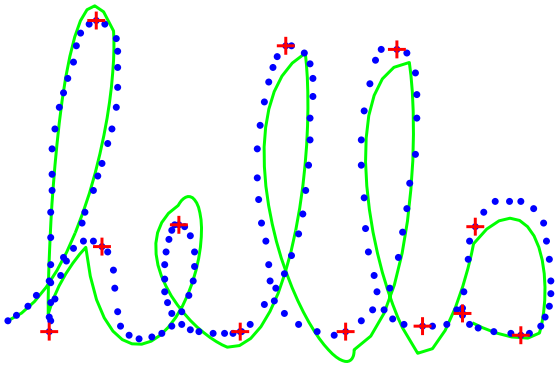
Trading Simplicity for Accuracy



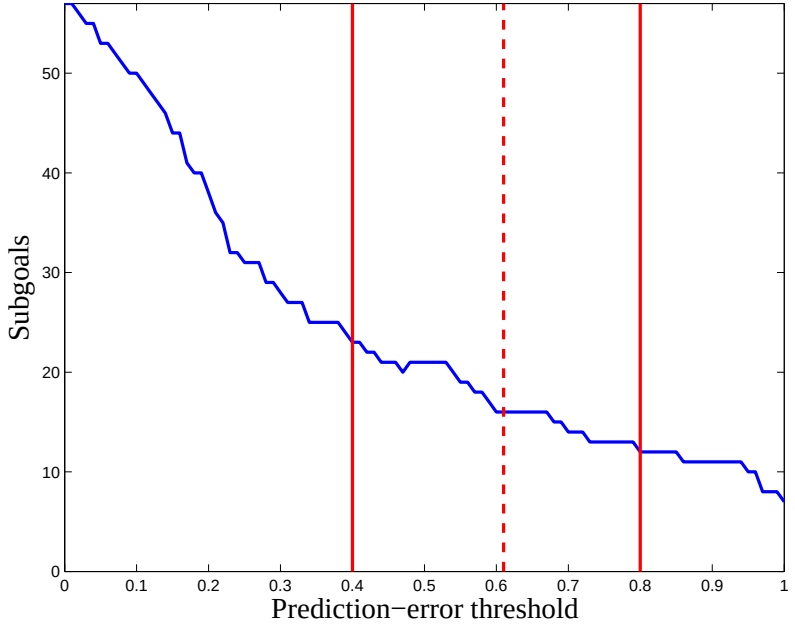
Error threshold = 0.4

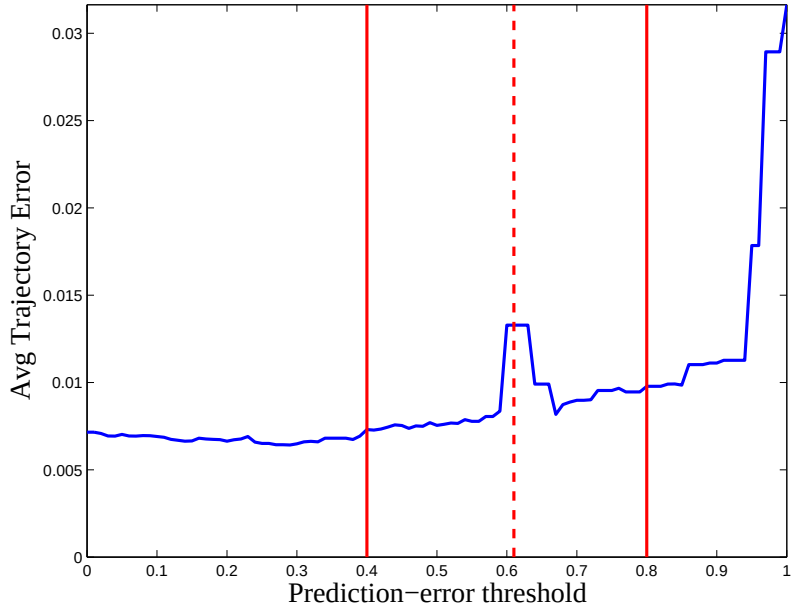


Error threshold = 0.61



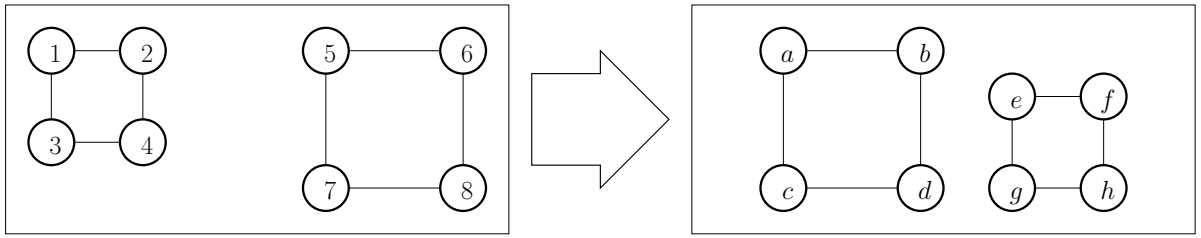
Error threshold = 0.8



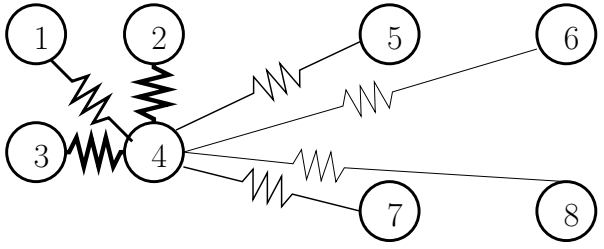


Electrical & Computer ENGINEERING

Environment Mappings



- Map each object on the left to one on the right



- Consider object connected to all others with “springs”
- Compute mapping that minimizes *total* change in force
- Weighted bipartite graph matching problem
  - Hungarian Method $\Rightarrow$ Linear Program (Papadimitriou & Steiglitz, 1998)