CSCI 561
Foundation for Artificial Intelligence

24: The Future of AI, ML, Robotics

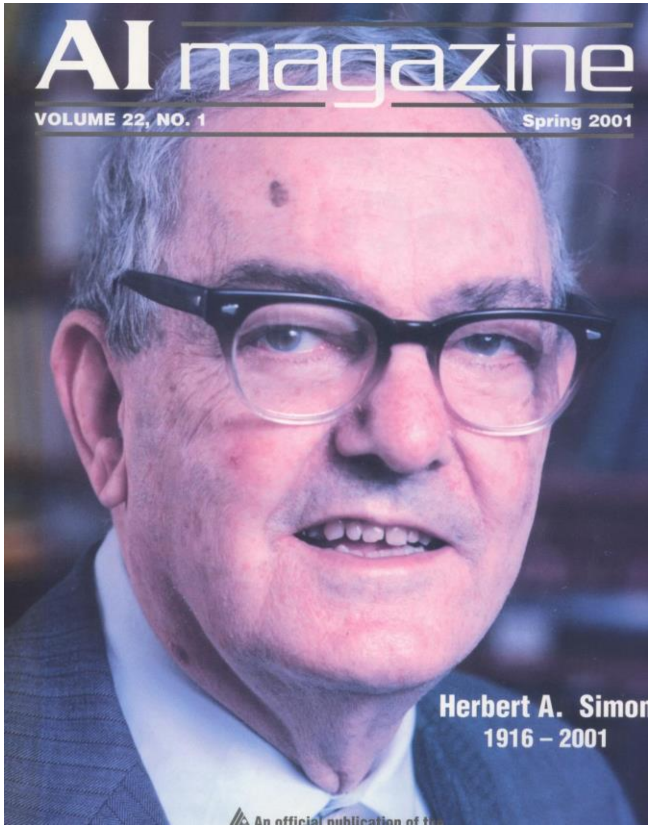
Professor Wei-Min Shen
University of Southern California

Outline

- Predicting or shaping the future?
- Robotics (see the handout slides)
- Autonomous Learning from the environment

Future of AI, ML, Robotics

- The Future of Artificial Intelligence
  - From DeepBlue to AlphaGo
  - From Roomba to UAVs and Swarm Robots
  - From autocoups learning and self-reconfigurable robots to soft-robots
  - !!!!!
- Quotes from the Pioneers of AI and ML
  - Professor Herbert A. Simon



Forecasting the Future or Shaping It?

October 19, 2000

Our task is not to *predict* the future; our task is to *design* a future for a sustainable and acceptable world, and then to devote our efforts to bringing that future about.

Professor Herbert A. Simon
Nobel Prize Laureate
A Founder of Artificial Intelligence

A Sustainable and Acceptable World!
--- Herbert A. Simon, 2000

1. We must find a way for living at peace with all of nature, not destroying the bases for the survival of all of us

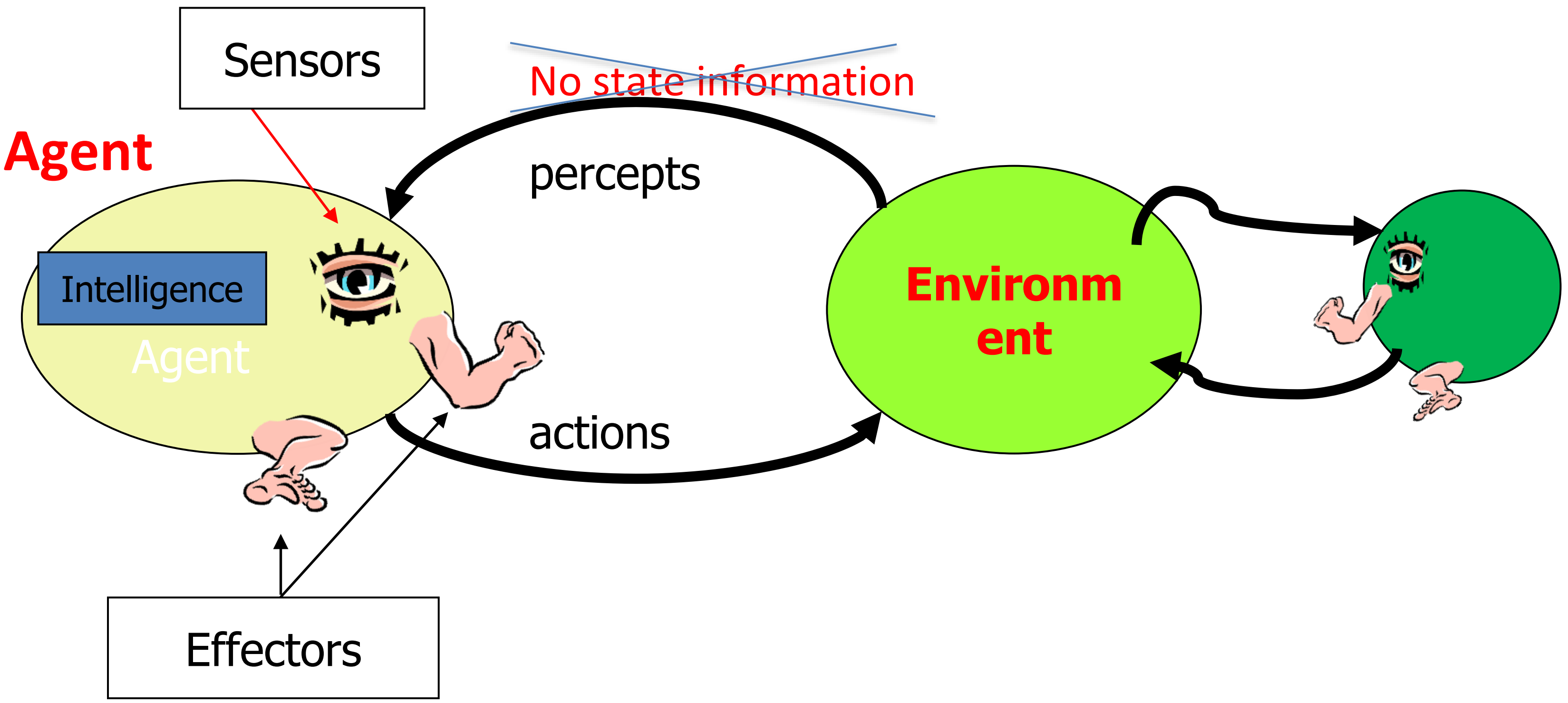
2. We must find a way for sharing broadly and fairly the outputs of our productive efforts of all kinds, no matter how ample or limited these outputs are in a sustainable world

3. We must, as a necessary condition for maintaining any standard of fairness, find some way of greatly mitigating, and if possible eliminating, the innumerable and passionate divisions of "we" from "they" that continue to make the human world a blood-stained collection of warring tribes, continually engaging in fluctuating patterns of mutual hostility and collective mayhem, of which traditional war is only one of the more obvious forms, perhaps not even the one most devastating or difficult to root out

Future of Artificial Intelligence

by Dr. Demis Hassabis (2017)

Agent, Environment, No States



Autonomous Incremental Structure Learning from the Environment

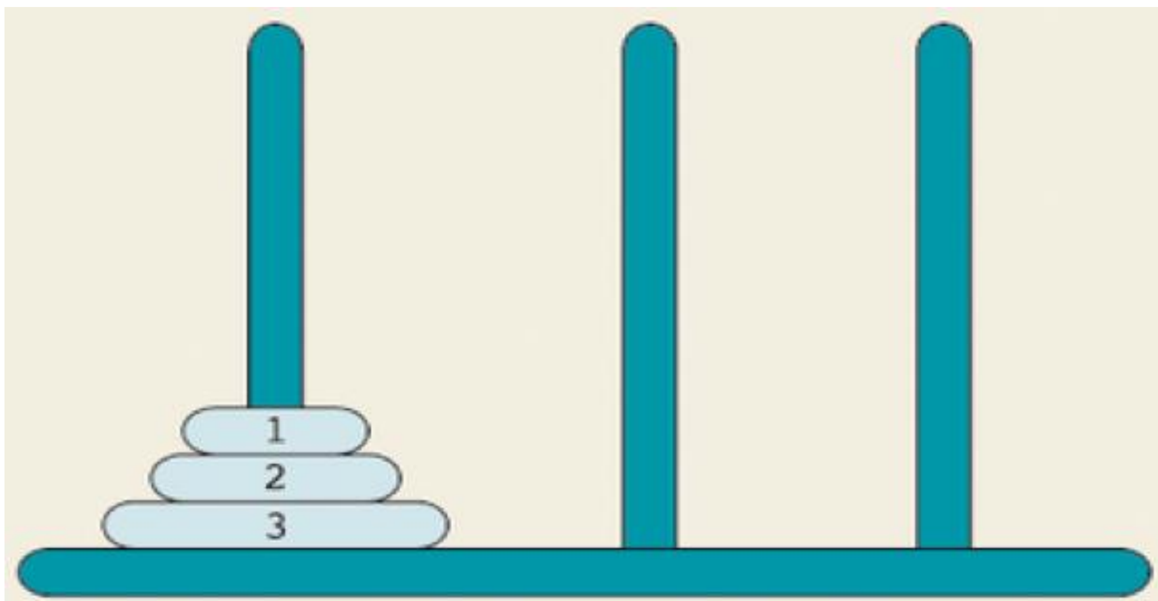
- Type I: Clustering the data
  - Automatically group the data into clusters
- Type II: Parameter Learning (states are known)
  - Learn transitions, sensor models, & current state
- Type III: Structural non-parametric Learning
  - States are not known and must be learned
  - States have internal structures (not just "symbols")

Formation of General Problems

- **$P(X_{1:t}|E_{1:t})$**
  – compare all state sequences I might go through ("explanation")
- **$P(X_t|E_{1:t})$**
  – which state I am in now ("state estimation" "localization")
- **$P(X_{t+k}|E_{1:t})$**
  – which state I will be in at time t+k ("prediction")
- **$P(X_k|E_{1:t})$**
  – which state I was in at time k (k<t) ("smoothing")
- **$P(x_{1:t}|E_{1:t})$**
  – what is the best state sequence that I went through ("viterbi")
- **$P(M_t|E_{1:t})$** // when states are known
  – How correct is my model at time t ("model learning")

- **$P(M_t|E_{1:t})$** // when states are not known
  – When states are not known, how can we construct and abstract a correct model from the experience in an environment?

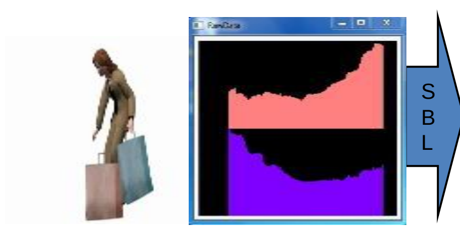
Example: Solving Tower of Hanoi without knowing the rules



See details in the ALFE book

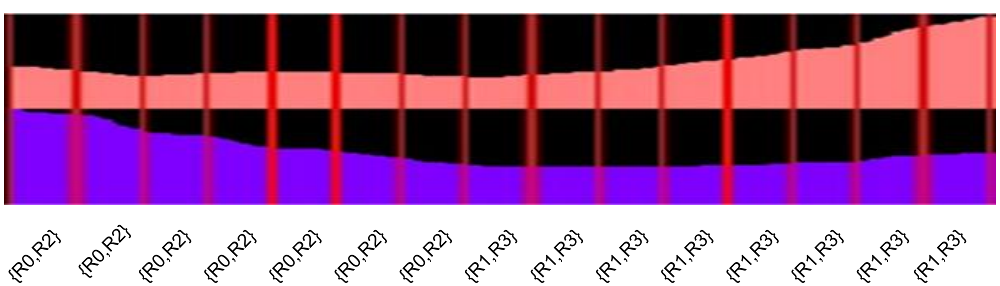
Recognizing Actions from Videos

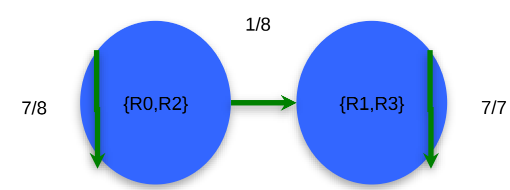
- Model Learning
  - Sample the data stream at fixed intervals, e.g. 10 frames
  - Learn a prediction model by observing data before and after each segment
  - Group prediction rules that fire together into a state
  - Extract the sequence of states (prediction sequence) for each example
  - Store state transition probabilities for positive examples in a Markov chain



RULES LEARNED FOR "*REPLACE*"

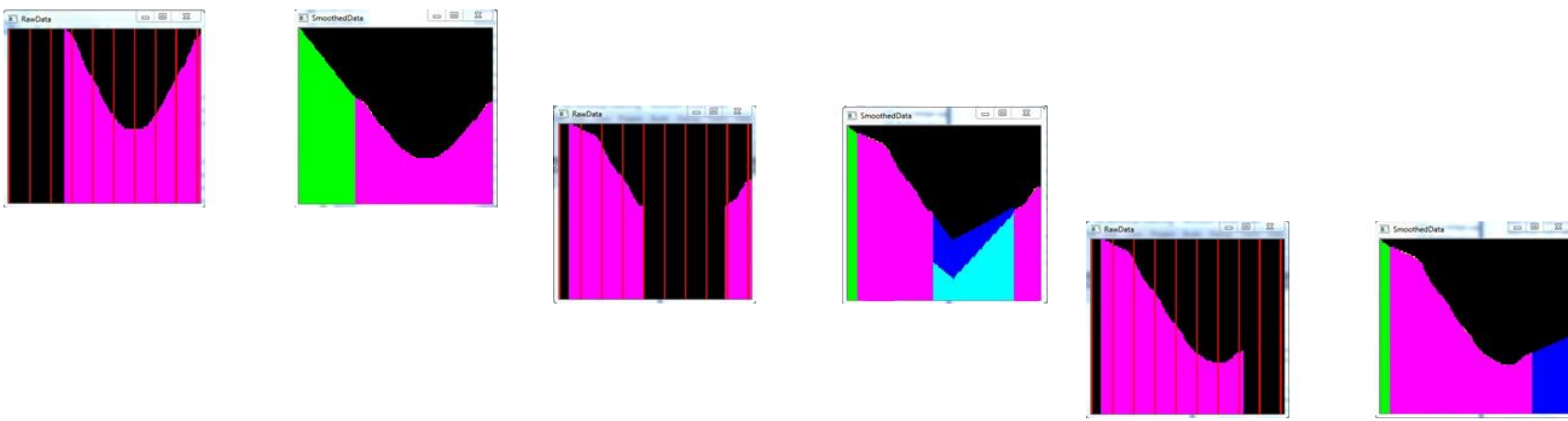
RULE 0 - 1 : 0.9 [(REL 0 % 0) ^ (REL 1 S < 60)] [ACTION = 1] [((REL 0 S = 50) OR (REL 1 S != 60) OR (REL 1 ~ 0))]
RULE 1 - 0 : 1 [(REL 0 % 0) ^ NOT { (REL 1 S < 60) }] [ACTION = 1] [((REL 0 S != 50) OR (REL 0 ~ 0)) AND ((REL 1 S ^ 5))]
RULE 2 - 3 : 0.9 [(REL 1 % 0) ^ (REL 1 S > 60)] [ACTION = 1] [((REL 1 S v 5) OR (REL 1 S != 60) OR (REL 1 ~ 0))]
RULE 3 - 1 : 1 [(REL 1 % 0) ^ NOT { (REL 1 S > 60) }] [ACTION = 1] [((REL 1 S ^= 5) OR (REL 1 ~ 0)) AND ((REL 1 S = 60))]





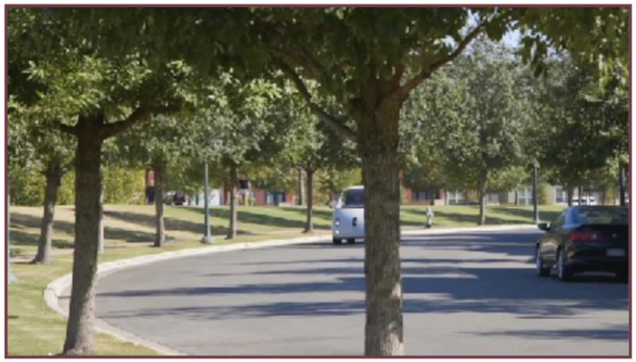
Example: Filling the Gaps in Data

- Perform BFS with the Markov Chain to fill gaps
  - Beginning – Post diction from first known state
  - Middle - Interpolation between known states
  - End - Prediction from the last known state
- Does not recover each trend, recovers state
- Slight penalties applied to decrease the likelihood action detection



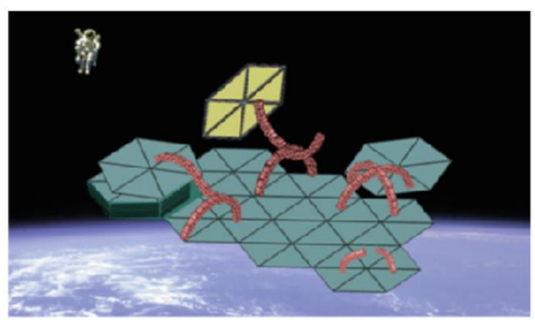
Adapting to New Environments

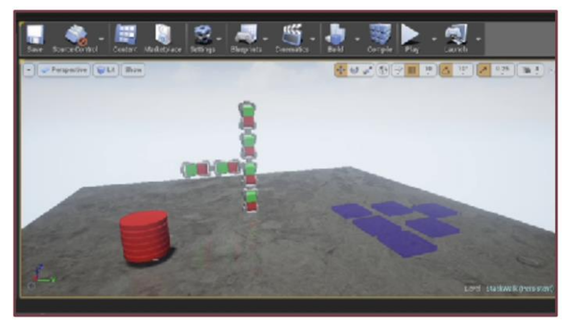
- There is an ever-increasing need for autonomous agents and robotic systems that are capable of adapting to and operating in a range of challenging environments.



- These environments vary substantially depending on the task, but virtually all environments of interest exhibit both **partial-observability** and **stochasticity**.

- Human-engineered features and prior knowledge remain some of the most important inputs to algorithms solving fundamental problems in robotics, machine learning, and AI.





- One of the most common forms of this prior knowledge is an **underlying state space representation** specified in terms of human-engineered features.

Other Famous Examples

- New Environment/Task ⇔ Learning ⇔ Knowledge/Skill
- Key Idea: To detect and learn from the experiences
  - Adaptive to new environments (may have no priori knowledge, "swim")
  - Learn to accomplish new tasks (goals may change dynamically)
  - Self-heal from unexpected failures or dynamics (e.g., inverted visions)

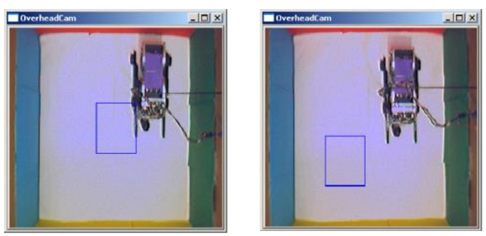


Technical Challenges (why hard?)

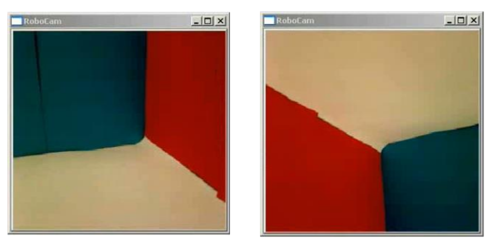
- Changes and mistakes are not easily detectable
  - Changes may not be predicted and occur simultaneously
  - Uncertainties in sensors and actions exist, not fault-free
  - Sufficient redundancy of components cannot be guaranteed
- Lack of complete and permanently correct knowledge
  - Sensor, action & environment models
  - Incomplete initial knowledge
  - Accidents happen
- Causes of interference may or may not be identifiable
  - Learn to reason with interference with/without adaptation
- May not have external supervision to detect changes
- Requires fast response times
  - Expensive solutions for memory/resource may not be feasible
- Must deal with both discrete and continuous data, uncertain and vast information space

Deal with Unpredicted Changes

- Uninformed changes to the robot at runtime
  - Sensors: addition, deletion, definition-change, interference
  - Actions: addition, deletion, definition-change, interference
  - Goals: addition, deletion
  - Dynamic configurations of the environment
    - Objects: addition, deletion, relocation
- “Definition-change” alters mapping of input to output
  - Rotation, Translation, Scale
- Interference includes noise & missing data or gaps



Goal change



Rotated camera

Gap

Problem Statements

Autonomous Learning from the Environment (**ALFE**): in which an autonomous agent is tasked with **actively** learning a **goal-independent** model of the **state** (and associated dynamics) of an unknown discrete, partially-observable and stochastic environment from a stream of experience (**actions** and **observations**).

- The agent knows nothing about the environment other than its **actions** and **observations**.
- The agent is responsible for **selecting its actions** and determining the **amount of experience** needed to build a well-performing model, making this problem highly unique in the literature.
- Learning is **non-episodic**, and the agent **cannot** reset itself to a known initial state.

Key Assumptions

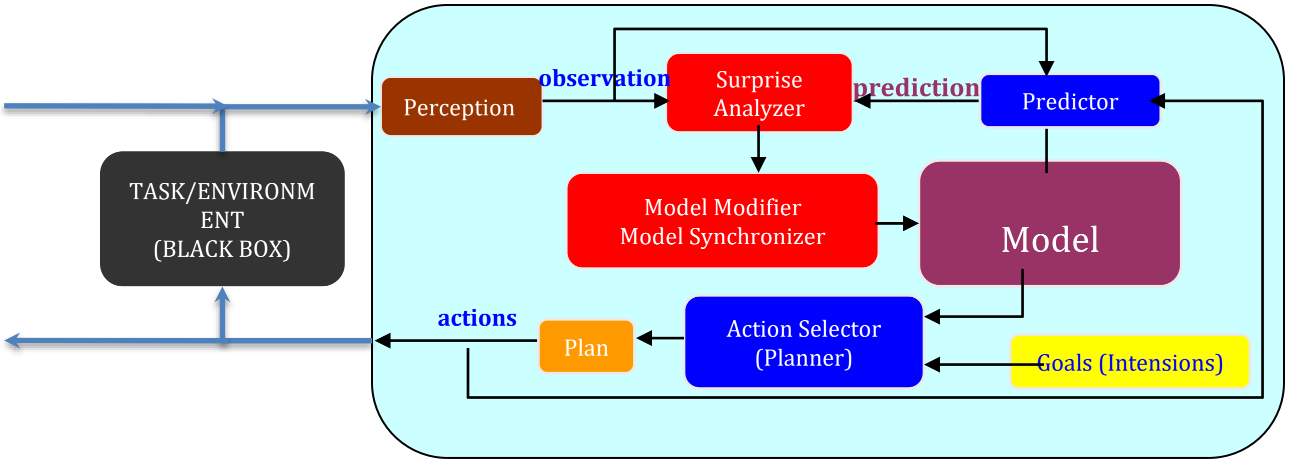
Autonomous Learning from the Environment (**ALFE**): in which an autonomous agent is tasked with **actively** learning a **goal-independent** model of the **state** (and associated **dynamics**) of an unknown discrete, partially-observable and stochastic environment from a stream of experience (**actions** and **observations**).

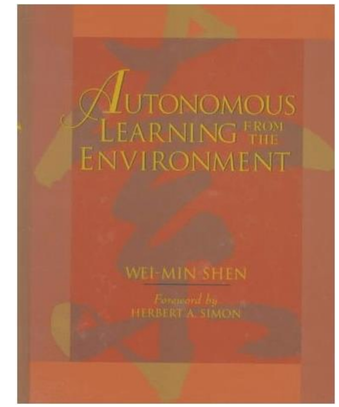
- **Assumption**: The environment is a discrete, rewardless, Partially-observable Markov Decision Process (**POMDP**): <$\cancel{Q}$, A, O, $\cancel{T}$, $\cancel{\Omega}$>.
  - ~~**Q** is a set of environment states.~~
  - **A** is a set of agent actions.
  - **O** is a set of agent observations.
  - ~~**T** is a set of transition probability distributions satisfying the Markov property.~~
  - ~~**Ω** is a set of observation probability distributions satisfying the Markov property.~~

Key Challenges

- Partial-observability
  - How much history does the agent need to consider?
- Stochasticity
  - What is noise and what is surprising?
- Minimal prior knowledge
  - How many parameters should the agent's model have?
- Action selection
  - How to choose actions with no explicit rewards?
- State space formalization
  - How can a model built in terms of action/observation sequences be used as a state space?
- Decision making
  - How can the agent use this model to make decisions that allow it to achieve goals?

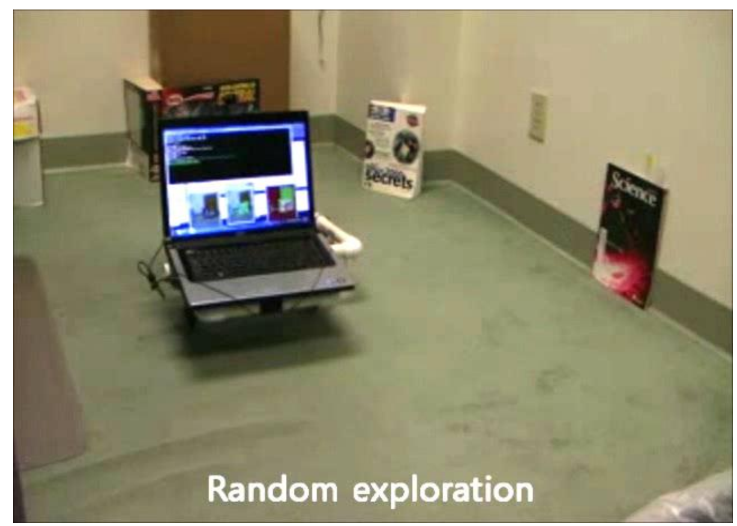
Surprise-Based Learning (CDL++)





- The Learner continuously makes predictions, detects surprise, analyzes surprises, extracts critical information from surprises, and improves and use its action models

Surprise ==> Model ==> Prediction



Surprises and Their Sources

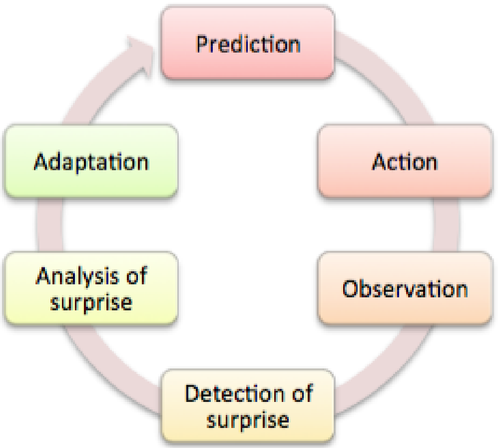
- What is a Surprise?
  - A significant discrepancy of mental predictions and sensor measurements
- Where are its sources?
  - Incomplete initial knowledge
    - Designers cannot capture the richness of the real world
    - e.g. Spirit rover got stuck in soft soil
  - Dynamic nature of the environment and tasks
    - Environment changes
    - System or robot's components change
    - Accidents happen
    - e.g. Spirit's right-front wheel stopped working
  - Lack of critical knowledge in the current KB
    - e.g. Toddlers gradually learn to grab stuff
    - e.g. Things wear out with time

Properties of Surprises

- Types of surprise
  - Unexpected failures
  - Unexpected successes
  - Null prediction surprise
    - When there is no a priori model
- Differentiate “new information” from noise
  - Focus attention on the relevant features
  - Seek differences that are statistically significant

The Cycle of SBL (CDL++)

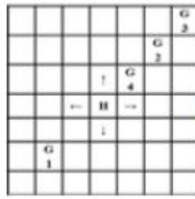
- Autonomously learn Predictive Rules from experience
  1. Select Actions for the current goals
  2. Make Predictions before actions
  3. Perform actions
  4. Observe events and entities via sensors
  5. Detect Surprises
  6. Analyze and explain Surprises
  7. Adapt or create prediction rules



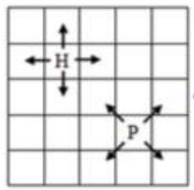
- **Detecting Surprises**
  - When a prediction does not match the actual outcome of an action
- **Explaining Surprises**
  - Finding the differences of a normal satiation and the unexpected situation
- **Revise Prediction Models**
  - Use the found differences to revise the relevant prediction rules
- **Adapt Decisions and Actions**
  - Use the adapted prediction rules to decide actions in future

Example Environments/Tasks

Simulated hunter-goal
Sensor: location
Entities: H, G
Attributes: x, y, u
Actions: N, S, E, W
Goal: H(x,y)=G(x,y) || H(u)=G(u)



Simulated hunter-prey
Sensor: location
Entities: H, P
Attributes: x, y, u
Actions: N, S, E, W
Goal: H(x,y)=P(x,y) || H(u)=P(u)



Real office room
Sensors: camera
Entities: SURF objects
Attribute: s, x, y
Actions: L, R
Goal: current=shown camera

Visint.org video dataset
Sensors: tracks
Entities: actors, objects, rel. distance
Attribute: s, x, y, velocity
Actions: 12 verbs
Goal: seen=learned verb

Contributions

- Structure Learning
- Learning from Uninterpreted Sensors & Actions
- Detecting and Adapting to Unpredicted Changes
- Detecting and Reasoning with Interference

Prediction Rules and Models

- A learned Predictive Model is represented by Prediction Rules
  - Rule = Conditions → $Action^+$ → Predictions
  - Conditions describe the state of observed entities and attributes prior to executing an action
    - Condition = (Entity, Attribute, Comparison Operator, Value)
    - e.g. c1 = (red, size, >, 10)
  - Predictions describe the forecasted state of entities and attributes as a result of executing an action (when conditions are satisfied)
    - Prediction = (Entity, Attribute, Expected Change, Value)
    - e.g. p1 = (blue, size, ↑, 5)
  - Rule example: [c1→ left → p1], or P(p1|c1,left)=0.62
- Rules can be sequenced
  - i.e. $[o_0, a_1, p_1, \ldots, a_n, p_n]$
  - Represent concepts such as plan, experiment, example and advice
  - Prediction sequences can capture temporal relations in observation sequences

Model Representation

- Prediction Rules
  - [C,A]→P, e.g., [Raining, Step-out] → Get-Wet
  - In the form of probability **P**(R|C,A), where R is the prediction
  - Different from C→A (production rules or action policy)
- Prediction Sequences
  - PS = (S_0,A_0,P_1,A_1,P_2,A_2, …, A_n, P_n)
- The multifunction of prediction sequences
  - <u>Plan:</u> if P_n is a goal
  - <u>Exploration:</u> if P_n guarantees a surprise
  - <u>Experiment:</u> if $\mathrm{P_n}$ may generate a surprise
  - <u>Example:</u> if the sequence will generate some surprises
  - <u>Advice:</u> if the sequence must be convert into prediction rules
- Layered networks for representing prediction rules from action-sensory data

Components of Prediction Rule

Goals

$G = \{(s_1e_1b_1{=}v_1) \wedge (s_1e_1b_2{=}*) \wedge (s_2e_1b_x{\neq}\emptyset) \wedge \ldots \wedge (s_ie_jb_k = v_l)\}$

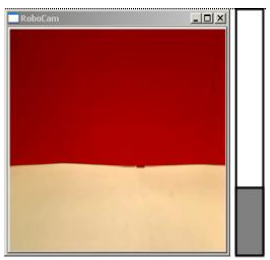
True when $O_t \vDash G$

e.g. desired observation shown

Models

Observations

$O_t = \{(s_1e_1b_1{=}v_1) \wedge \ldots \wedge (s_ne_ib_j{=}v_k)\}$



Sensors

$\{s_1, s_2, \ldots , s_n\}$

e.g. {proximity, camera, goal}

Actions

$\{a_1, a_2, \ldots , a_n\}$

e.g. {F,B,L,R}

Comparison Operators

$\{\%, \sim, \uparrow, \downarrow, \odot_1, \odot_2, \ldots , \odot_n\}$

e.g. {%,~,<,<=,=,!=,>=,>,↑,↓}

Entities

$\{e_1, \ldots , e_f\}$

e.g. {blob_white, blob_red etc.}

Attributes

$\{b_1, \ldots , b_g\}$

e.g. {size, center-x, center-y}

Values

continuous $[p, q]$ or *discrete* $\{v_1, v_2, \ldots , v_h\}$

Predictions

Prediction for t is a forecasted observation in CNF prior to t

$P_t = \{(s_1e_1b_1{\odot}v_1) \wedge \ \ldots \wedge \neg(s_2e_1b_1{\odot}v_k)\}$

True when $O_t \vDash P_t$

Surprises

A discrepancy between a prediction and its observation at t

True when $O_t \nvDash P_t$

Analyzing and Using Surprises

- Extract the critical information from a surprise
  - Create new model, if it is a null-prediction surprise
    - (~non-exists ^ exists) = action => (appeared ^ disappeared)
  - Improve the learned model
    - Compare the surprised situation with previous situations where the prediction was successful (i.e., “find the differences”)
    - Identify salient features that discriminate the situation
    - Which part of the memory to compare? (the effect of “first kiss”)
- Finding the true cause of a surprise could be difficult
  - E.g., Raining days? Mondays?
  - Too many differences
  - Too few differences (no differences)

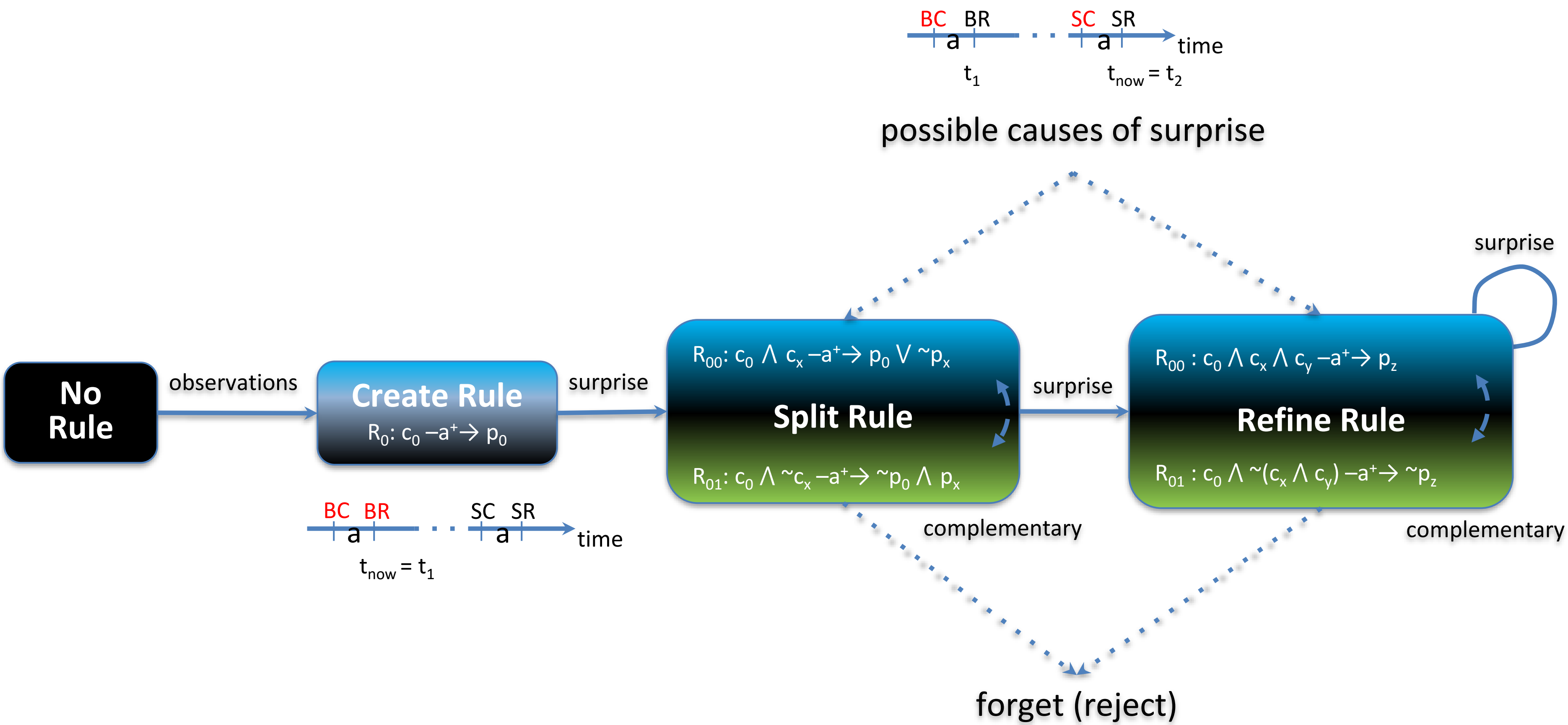
Desirable Features

- Autonomously extracts the states and corresponding state machines from the training dataset object tracks
- Number of states, observed features of states & transitions between states need not predetermined
- Top-down approach to meet low-level measurements and evidences
- Learn the structure of states for description and explanation of revisions
- “Surprise” connects to “anomaly” and “unexpected interference”

Some Initial Results

- Discrete or symbolic environments
  - Developmental learning to use novel tools
  - Scientific discovery of hidden features (genes)
  - Game playing
  - Learning from large knowledge bases
- Robotics environments
  - Sensor recovery in office environment
  - Recognize/predict visual acDons from video
  - Detect abnormality in robot/UAV swarms

Life Cycle of a Predictive Rule/Model



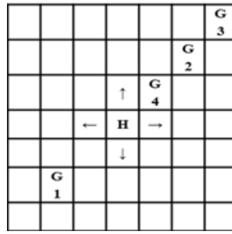
Dealing with Unpredicted Changes

- Goal change
  - Plan with the learned model to observable goals
  - Maintain a list of goal observations for hidden goals
- Dynamic configurations in the environment
  - Forget obsolete rules through rule rejection when contradictions or the success probability of a rule drops below a cutoff
- Action & sensor changes
  - Maintain a table with each row recording the relevance of a sensor, entity, attribute and action
  - Identify irrelevance by monitoring entities and attributes in the learned prediction rules
  - Force relevance when none of the active entities or attributes change

Some Evaluation Environments

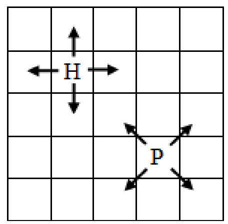
Simulated hunter-goal

Sensor: location
Entities: H, G
Attributes: x, y, u
Actions: N, S, E, W
Goal: H(x,y)=G(x,y) || H(u)=G(u)



Simulated hunter-prey

Sensor: location
Entities: H, P
Attributes: x, y, u
Actions: N, S, E, W
Goal: H(x,y)=P(x,y) || H(u)=P(u)



Real office room

Sensors: camera
Entities: SURF objects
Attribute: s, x, y
Actions: L, R
Goal: current=shown camera

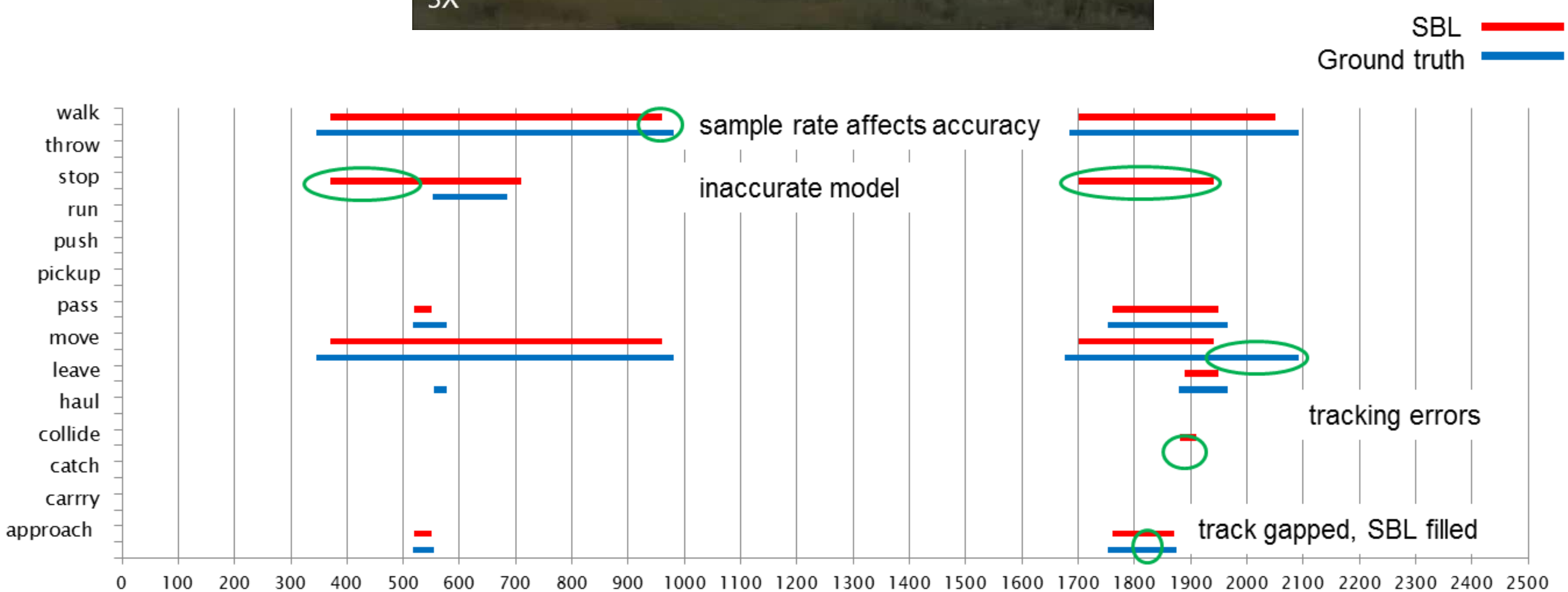
Visint.org video dataset

Sensors: tracks
Entities: actors, objects, rel. distance
Attribute: s, x, y, velocity
Actions: 12 verbs
Goal: seen=learned verb

Contributions

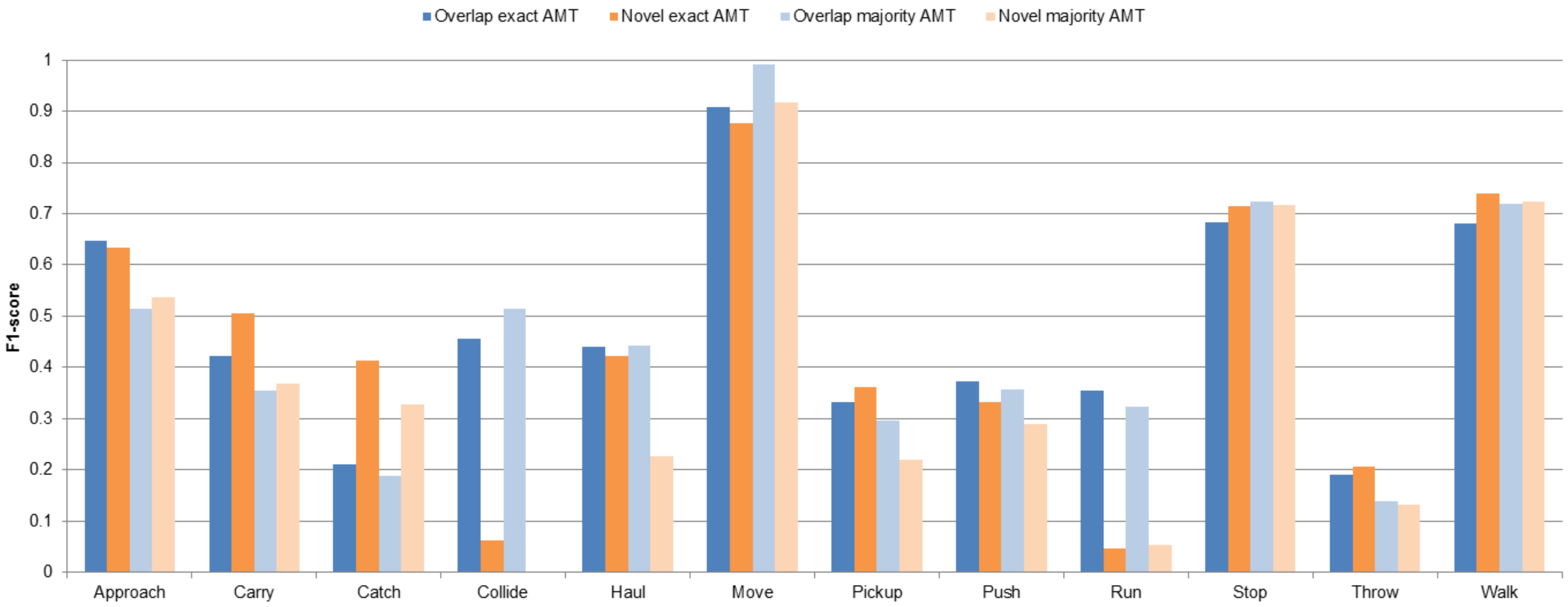
- Structure Learning
- Learning from Uninterpreted Sensors & Actions
- Detecting and Adapting to Unpredicted Changes
- Detecting and Reasoning with Interference

Recognition Example



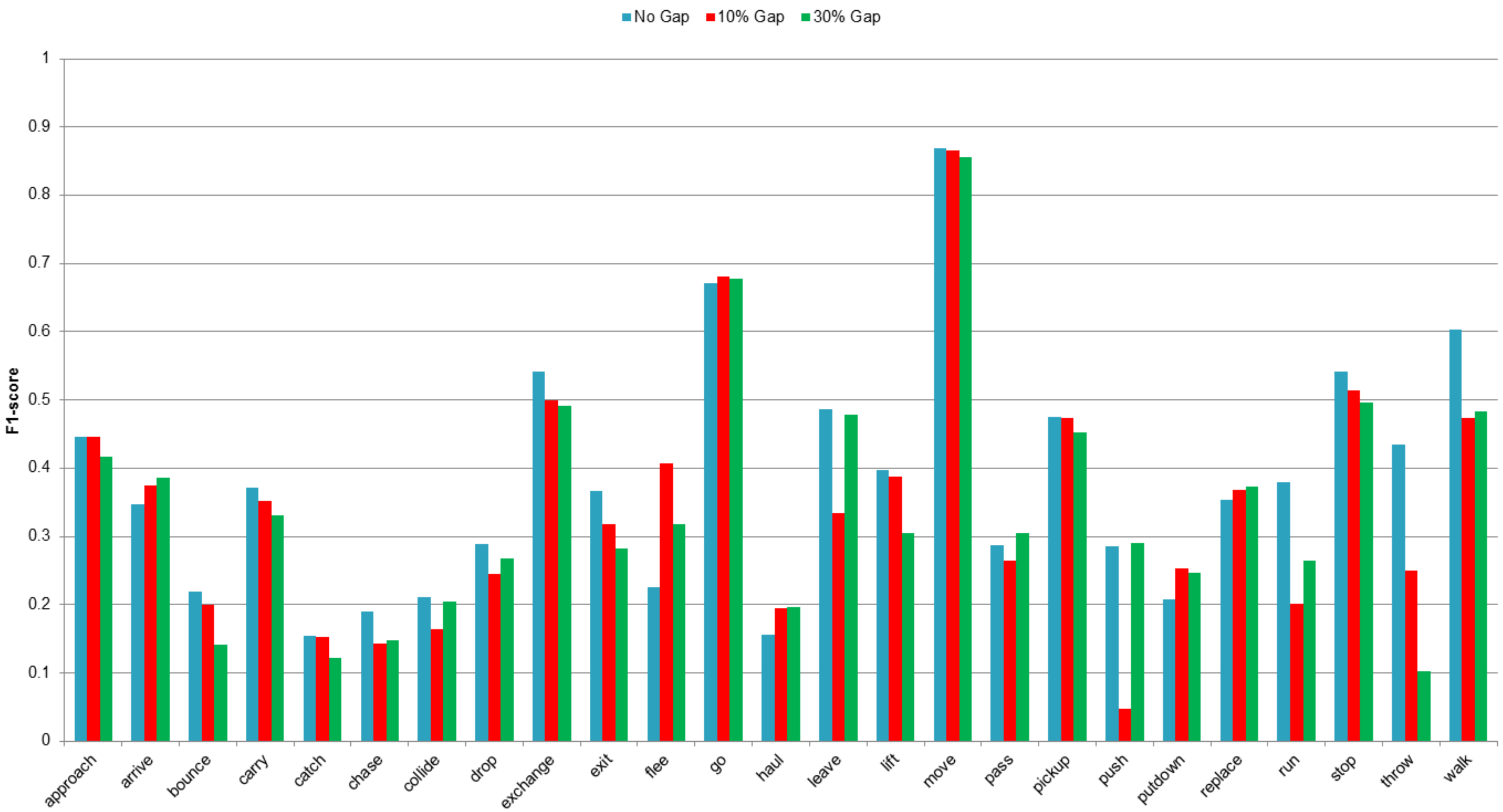
Recognition Results

- Visint.org Year 1 Dataset, USC tracks, 1200 videos
- F1 = (2*precision*recall)/(precision+recall)
- Outperformed hand-coded SAM by more than 10%
- Reasoning with detector accuracies below 20%



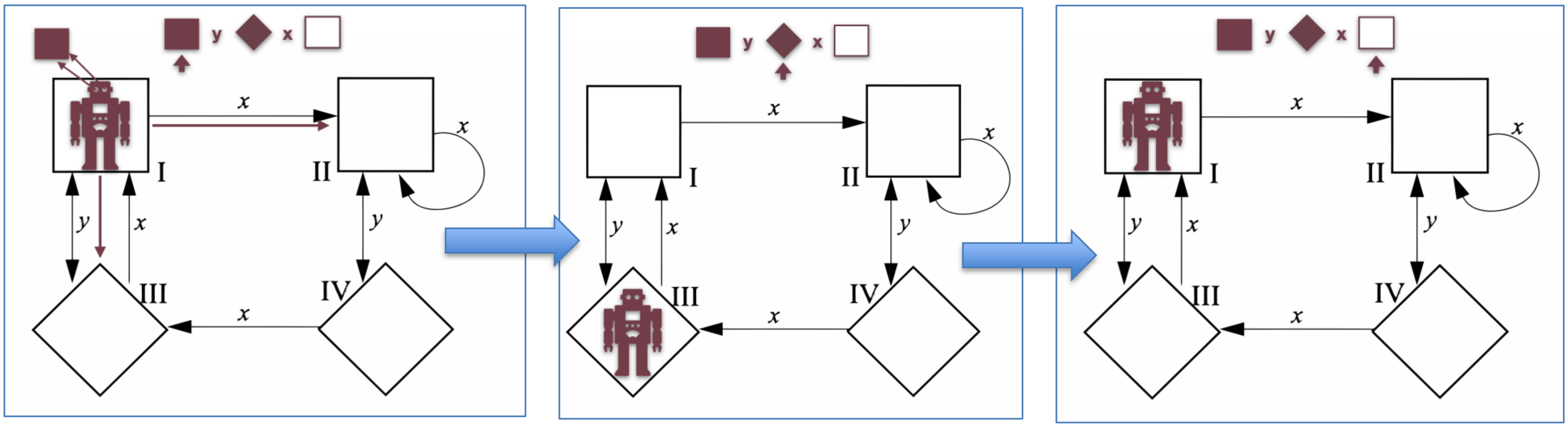
Gap Filling Results

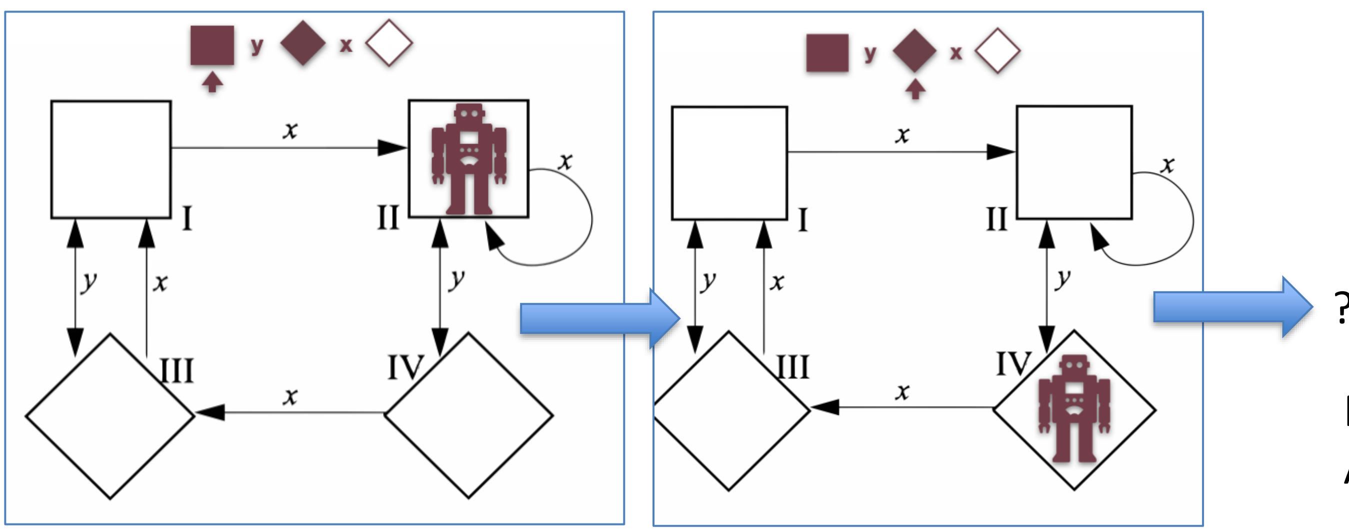
- Synthetic gaps introduced into 700 annotated videos



Stochastic Distinguishing Experiments

- TJ Collins and WM Shen, 2018





Predict [], but see <>
A surprise!

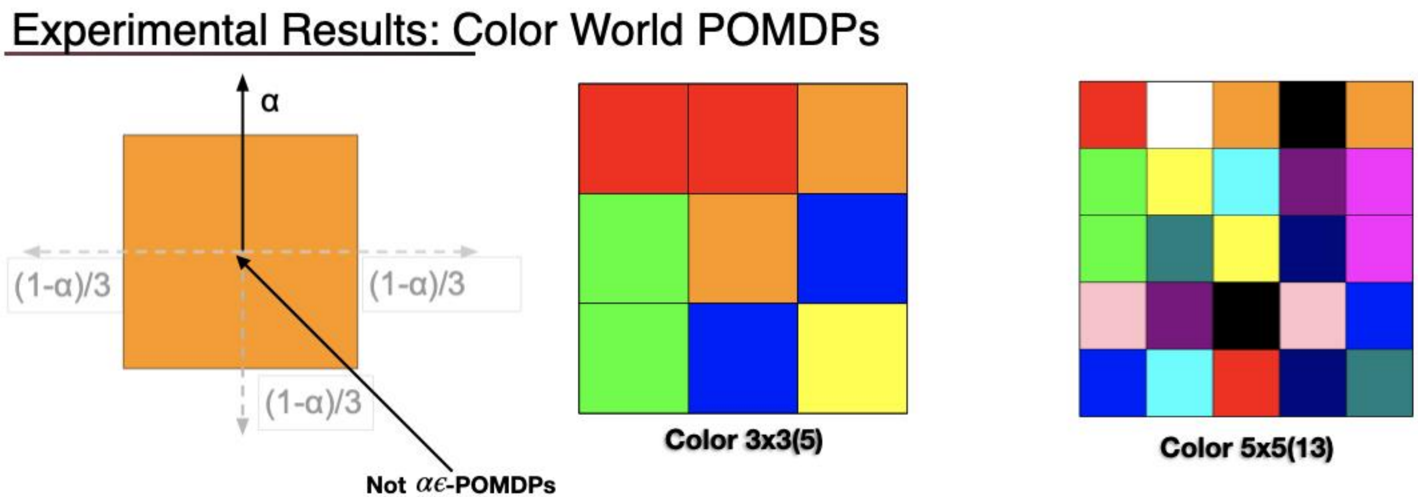
Prediction -- Formal Definition

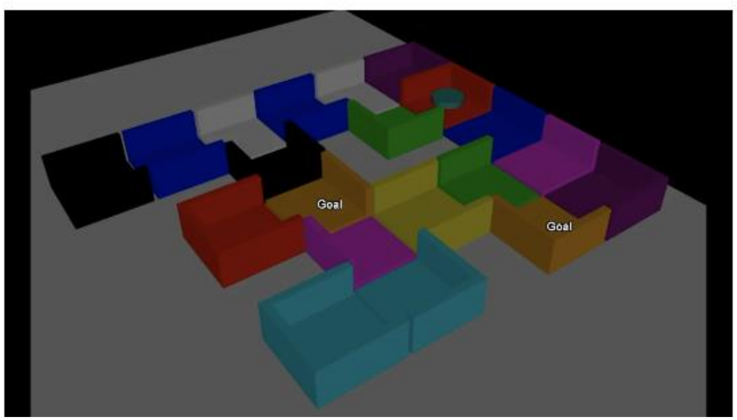
- A Predictive Stochastic Distinguishing Experiment (**PSDE**) is a **time-invariant conditional probability distribution** over the next observation given a **finite**, **variable-length** sequence of ordered actions and expected observations up to the present.
- In a valid **PSDE model**, the **conditions (experiments)** of a set of PSDEs form a mutually exclusive and exhaustive set of possible agent history suffixes.
- By maintaining an amount of history equal in length to the **longest** PSDE (which is provably finite), the agent can approximate the **history-dependent** probability over its next observation for **any** possible history **without** attempting to model latent environment structure.
- Since the agent can construct PSDEs of arbitrary length by considering additional time steps of history, this is a **nonparametric** model.

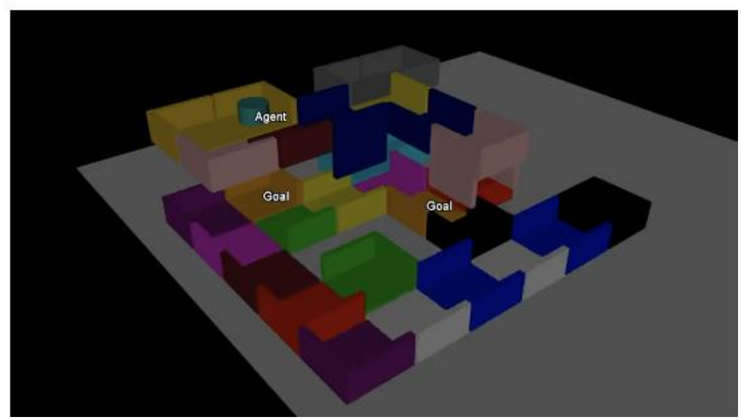
Length k PSDE $\longrightarrow$ $$P(O_{t+1} \mid \underbrace{o_{t-k+1}, a_{t-k+1}, \ldots, o_t, a_t}_{\text{Condition (Experiment)}})$$

Random variable representing the next observation

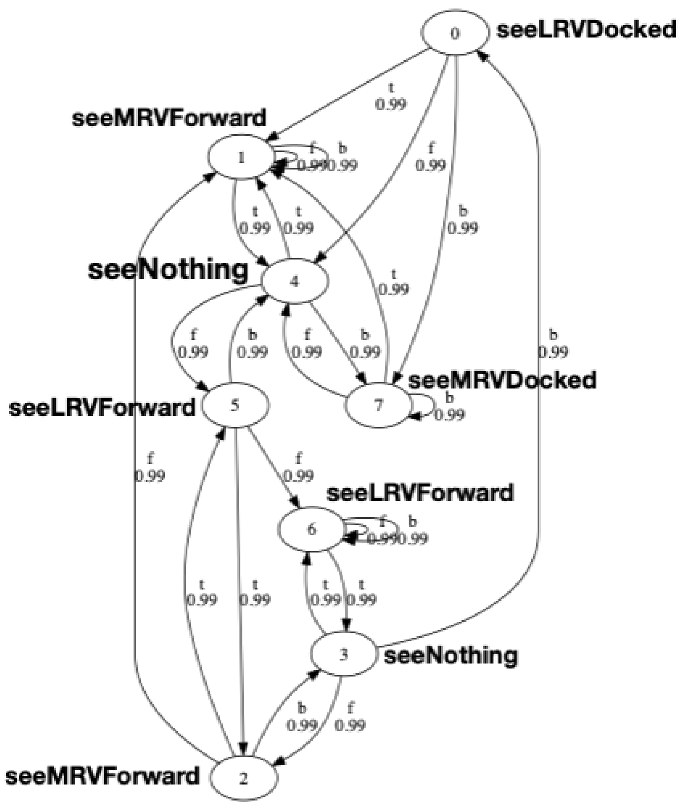
Surprise-Based Learning POMDP - the color world environment

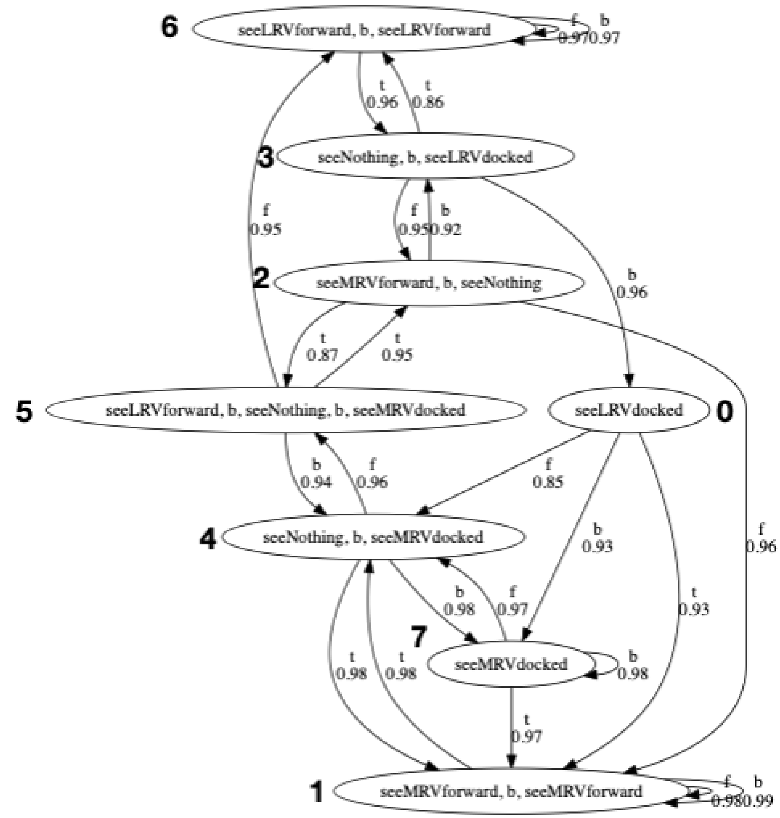






Surprise-Based Learning POMDP (shuttle environment)





Summary of SBL Capabilities (so far)

Structure Learning Surprise-Based Learning for Developmental Robotics, LAB-RS 2008	+Discover the number of states +Scalable/not over fitting in structured environments +Few training examples produce a good model +Learns generic models that can solve specific goals +Temporal modeling with predictive capability
Learning from uninterpreted sensors and actions The Surprise-Based Learning Algorithm, ISI Technical Report 2008	+Preprocessing not required +Can use preprocessed data +Discretizes continuous data +Identifies useful sensors and ignores irrelevant sensors
Detecting and adapting to unpredicted changes Autonomous Adaptation to Simultaneous Unexpected Changes in Modular Robots, IROS 2011 Workshop Surprise-based developmental learning and experimental results on robots, ICDL 2009	+Adapt to action changes +Adapt to sensor changes +Adapt to environmental changes +Adapt to goal changes +Adapt to simultaneous action, sensor, environment & goal changes +Repairs model faster than rebuilding it from scratch
Detecting and reasoning with interference Autonomous Surveillance Tolerant to Interference, TAROS 2012	+Detects noise +Detects and fills gaps +Quantify similarity between learned model and observed data +Reason with data of variable durations

Future Work for SBL

- Explicitly tolerate interference during model learning
  - Probabilistic rules can accommodate noise
    - Keep copies of the original rules during splitting and refinement
    - Planner must consider these probabilities to remove invalid plans
- Faster model convergence
  - Incorporate knowledge that some actions are reversible
  - Hypothesize and test
    - Create abstracted prediction rules during rule creation
    - Identify hidden states with Local Distinguishing Experiments [2018]
    - Utilize multiple SBL learners (distributed or nested hierarchy)
- Improve scalability in the number of sensors
  - Limit search space of surprise analysis through goal directed learning
- Reduce planning overheads
  - Generate policies with the prediction model for common goals

Surprise-Based Learning

- Herbert A. Simon (1916-2001)
  - Nobel Prize Winner (AI, Machine Discovery)
- Wei-Min Shen (1983 – now)
  - Autonomous Learning from the Environment
  - Discovery of hidden variables
- Nadeesha Ranasinghe (2005 – 2011)
  - Learn and predict unexpected changes
- Thomas J. Collins (2013 – 2018)
  - Discovery of hidden/latent structures