CS561 Fall 2018
Foundation of Artificial Intelligence

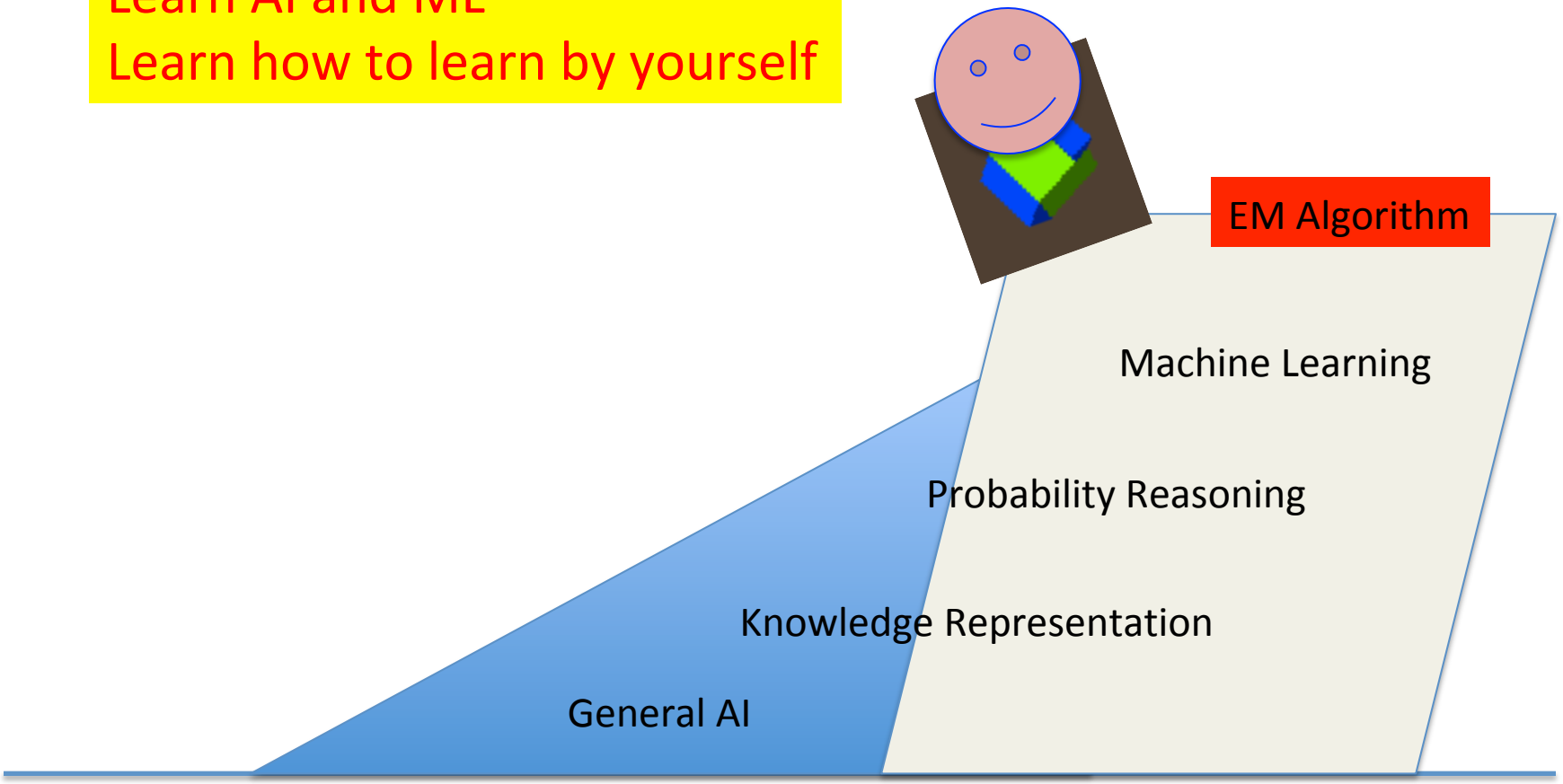
Intelligent Systems

Tuesday Lectures, Session 24-25
6:40-7:55 & 8:05-9:20

Professor Wei-Min Shen

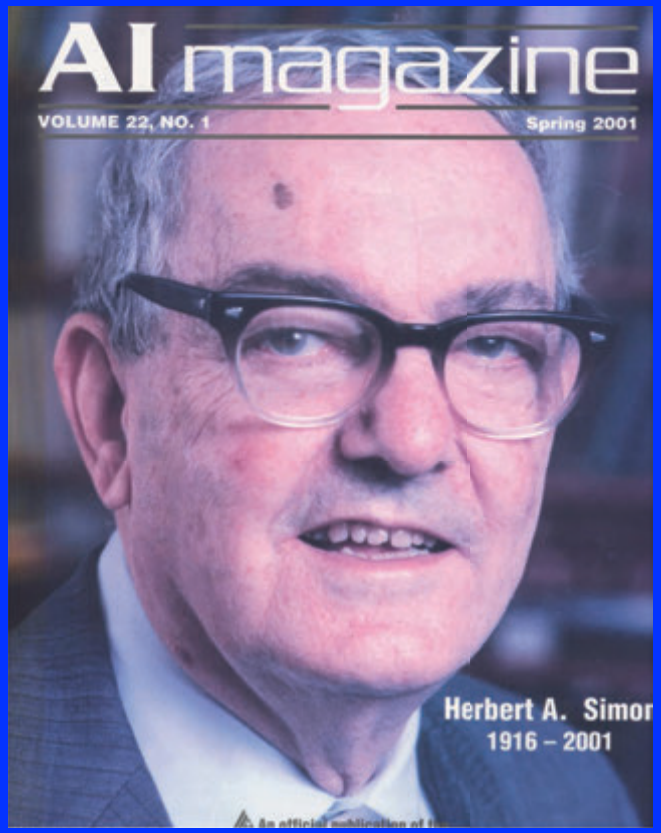
What You Have Learned This Semester

Learn AI and ML
Learn how to learn by yourself



Self-Reconfigurable Robots, Digital Hormones, and Surprise-Based Learning

Wei-Min ("wingman") Shen
Polymorphic Robotics Laboratory
Information Sciences Institute (ISI)
Computer Science Department (CS)
Center for Robotics and Embedded Systems (CRES)
University of Southern California
http://www.isi.edu/robots



Forecasting the Future or Shaping it?
October 19, 2000

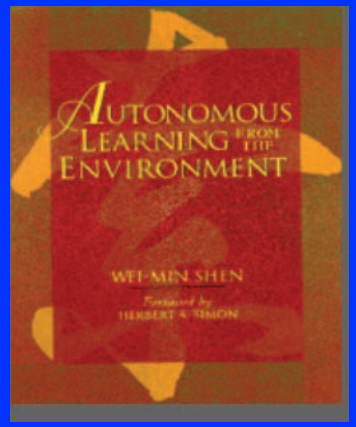
Our task is not to *predict* the future; our task is to *design* a future for a sustainable and acceptable world, and then to devote our efforts to bringing that future about.

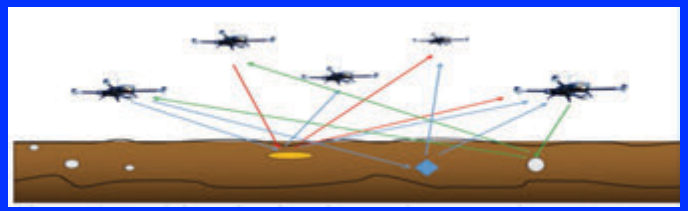
Professor Herbert A. Simon
Nobel Price Laureate
A Founder of Artificial Intelligence

Prof Wei-Min Shen http://www.isi.edu/robots

Self-Reconfigurable Modular Robots (System of Systems)

Surprise-Based Learning





Polymorphic Robotics Lab

http://www.isi.edu/robots

- Our Research
  - ***Self-Reconfigurable Systems***, including robots, AUVs, UAVs, agents, structures, networks, bio-models, …
- Funding Agencies
  - Government: DARPA, NASA, NSF, AFOSR, ARO, NavLab
  - Industry: Chevron, Motorola, LA, …
- Awards
  - Navigation Robots: AAAI Robotic Competition (1996)
  - World-championship Soccer-playing robots (1997)
  - Best Paper at SAB Conference (2002)
  - USC Research Awards (2003)
  - Robotic Contiguous Challenge Award, ICRA (2008)
  - Best Paper at 26th Army Science Conference (2008)

World Champion Soccer Robots (Nagoya, Japan, 1997)

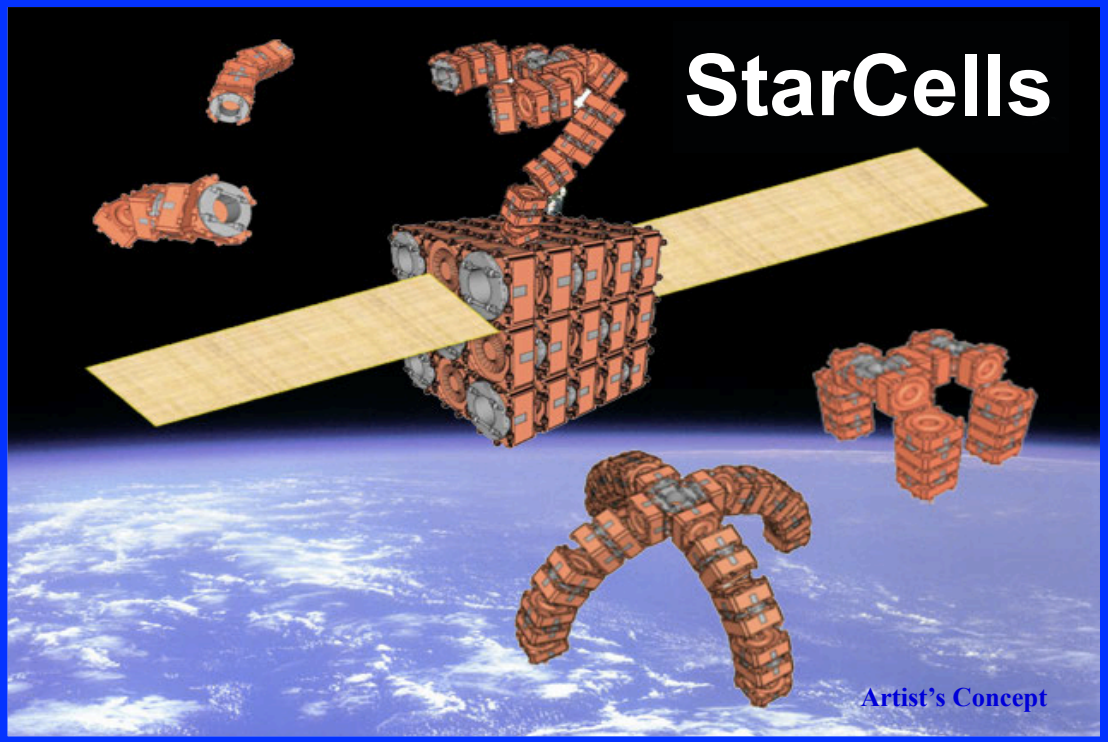
Self-Reconfigurable Robots

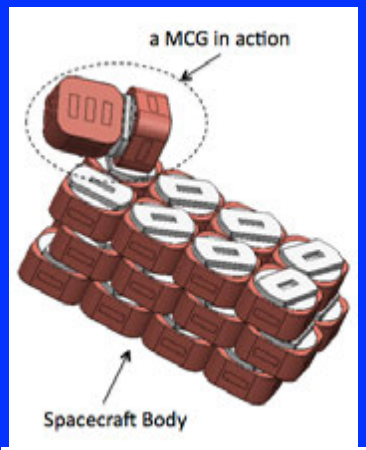
Introduction Movie

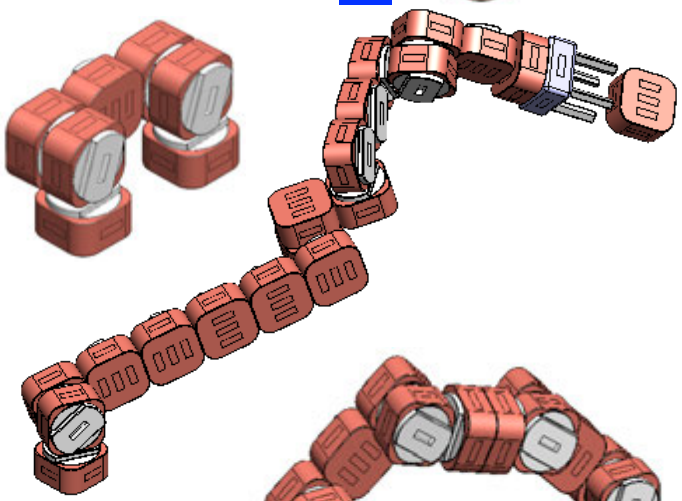
SuperBot Vision for Space

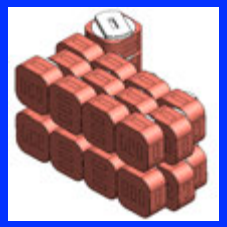
Modular, Multifunction, Self-Reconfiguration

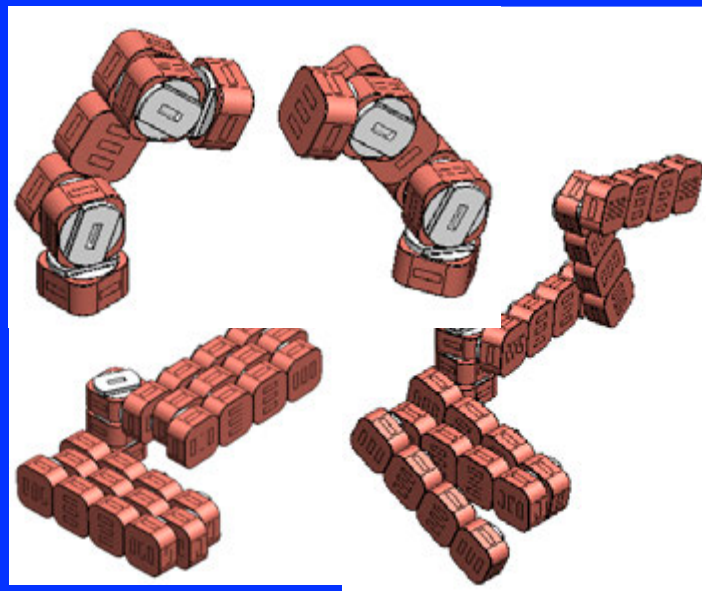
StarCells

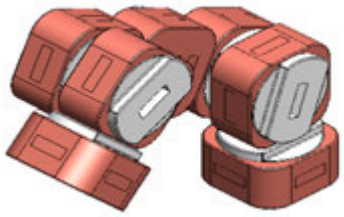


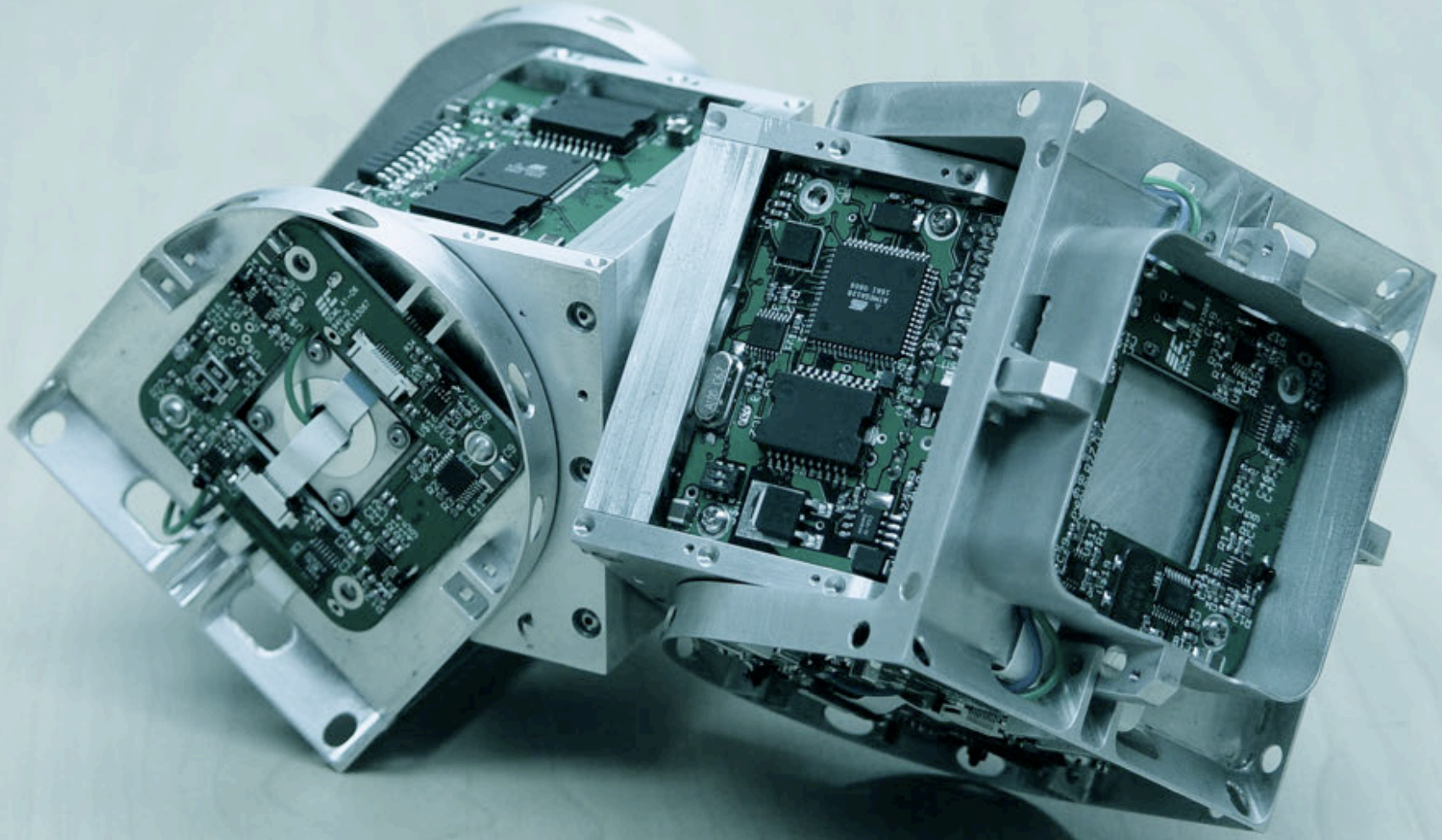












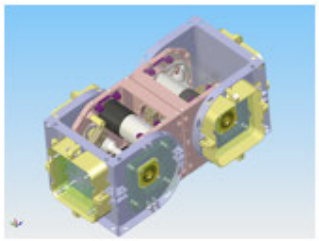
SuperBot: Key Concepts

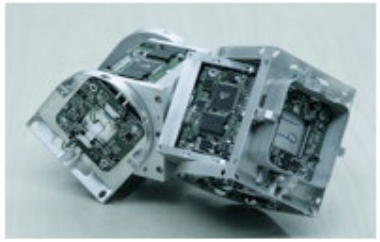
Invention
- Development of connectable modules with onboard power, connectivity, motion, sensing, and intelligence

Key Features
- Capable of operating with other similar modules to morph into multi-shapes and multifunctional self-reconfigurable robots

Applications
- High-risk environments, such as space, underwater, nuclear sites, military, civilian, and reconfigurable/flexible manufacturing

"SuperBot" module specification
- 3-DOF rotational mechanism
- 6 connectors for reconfiguration
- Onboard batteries (7.4v 1.6Ah)
- 2 micro-computers (Atmage128)
- Communication by 4x6 infrared
- 3D gravity sensors
- 4x6 Proximity sensors
- Radio remote control
- Real-time operating system
- C programming environment
- Distributed control
- Topology-triggered behaviors



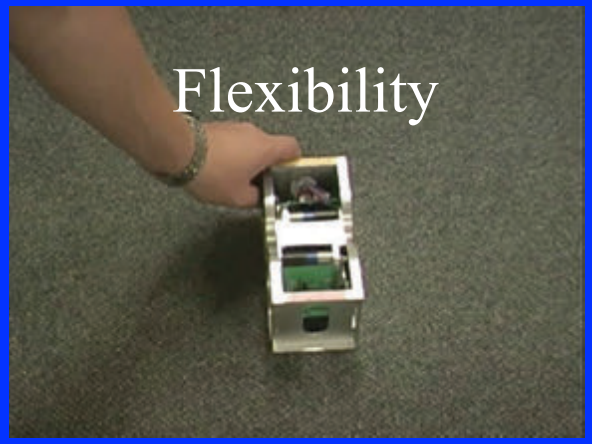


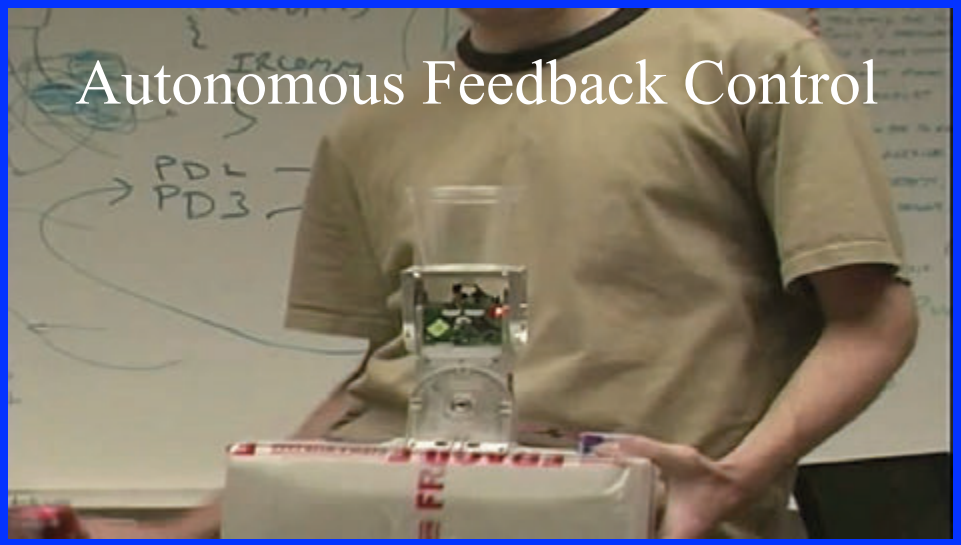

Dr. Wei-Min Shen, www.isi.edu/robots/, shen@isi.edu, 310-448-8710

Key Inventions & Patents

- Mechanical mechanisms
  - 3D reconfigurable modules
  - Self-healing connectors (US Patent)
  - Multiplex thrusts (US Patent)
- Electronic Components
  - Power Sharing, local and global communication
  - Multifunctional Mechatronics Architectures
- Intelligent Control Software
  - Digital hormones (US Patent)
  - Surprise-based Learning and Adaptation
  - Self-reconfiguration on-line planning
  - Self-assembly and self-healing

Single Module Autonomous Robots





Possible Configurations

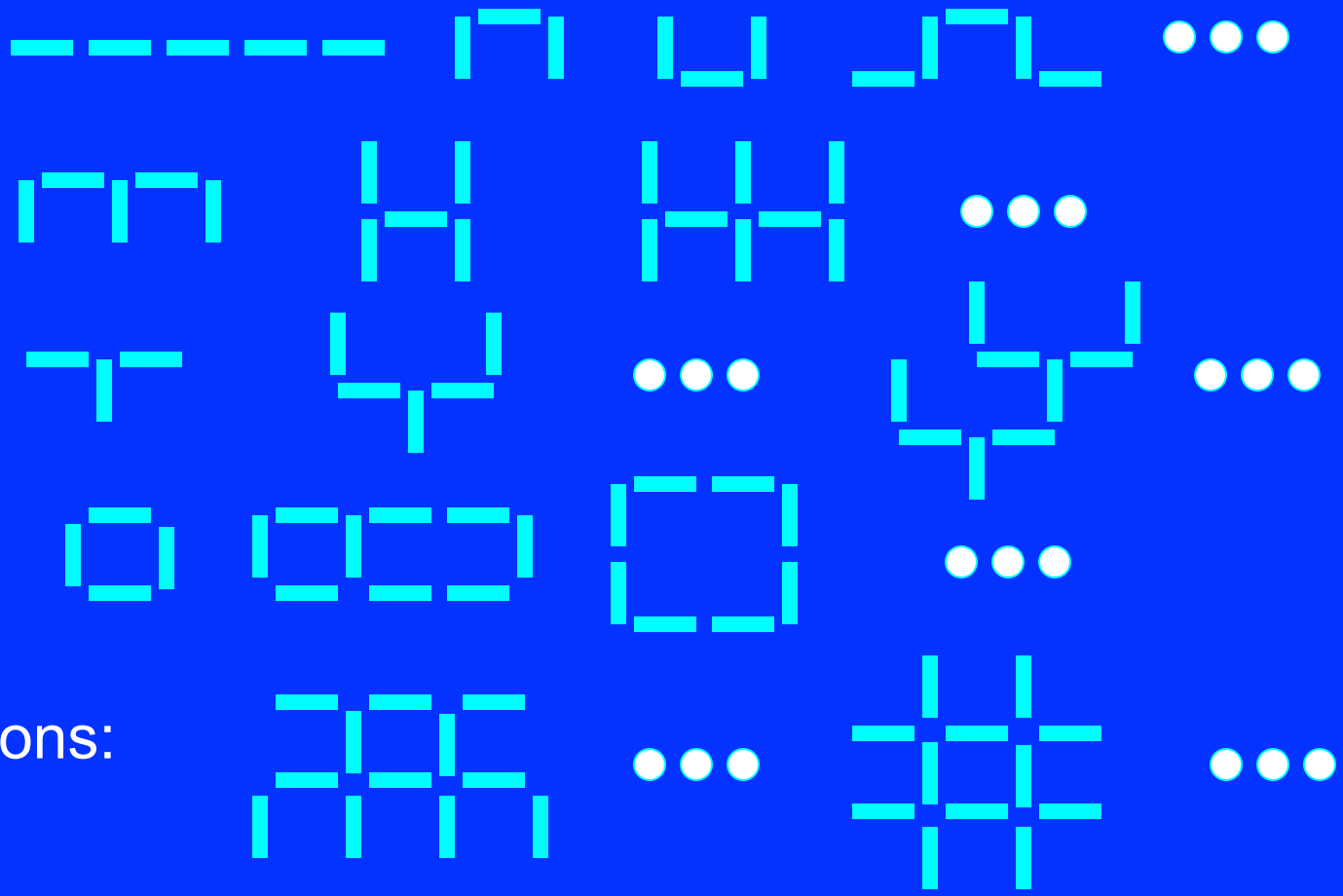
Chains:

Legs:

Trees:

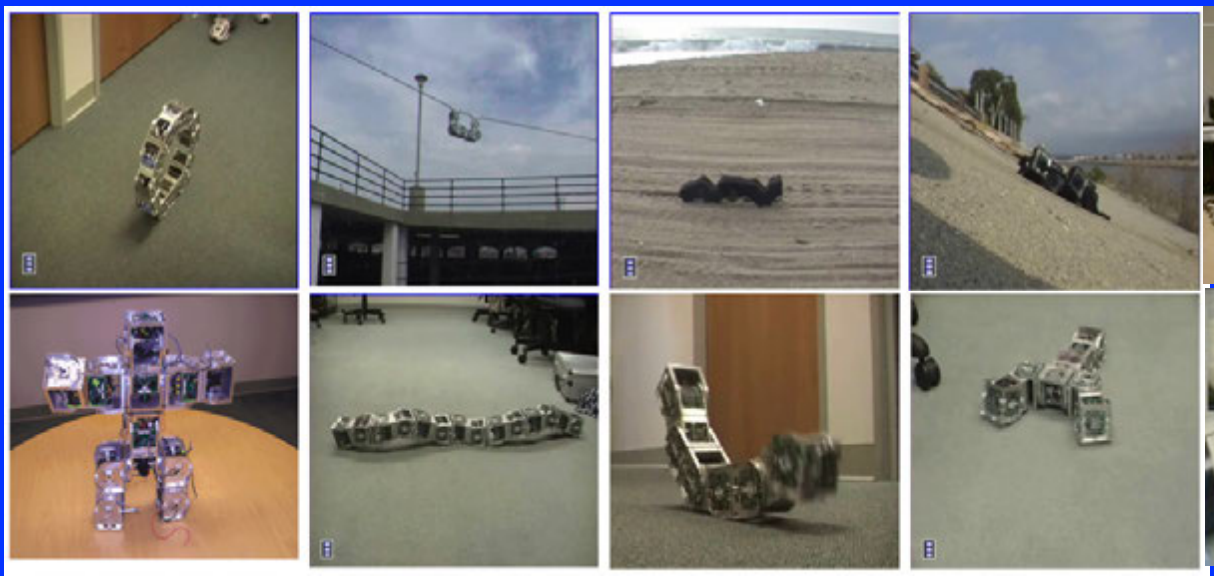
Loops:

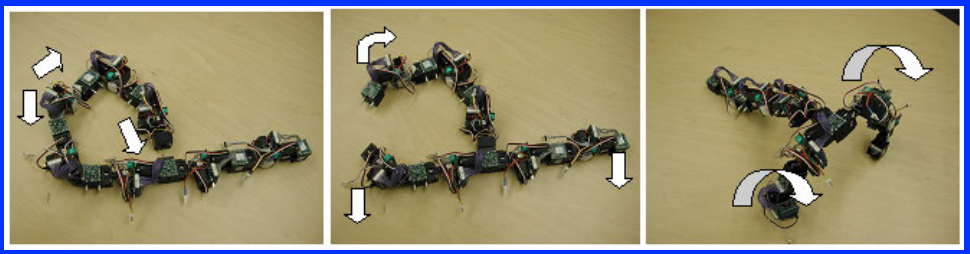
Combinations:

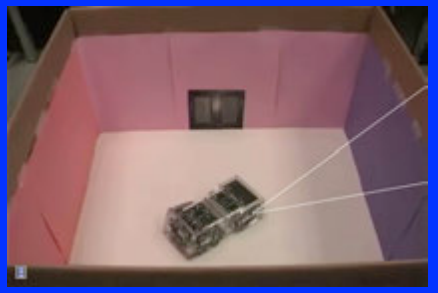


Modules in Configurations

Multifunction, Reconfiguration, Distributed Control, & Learning

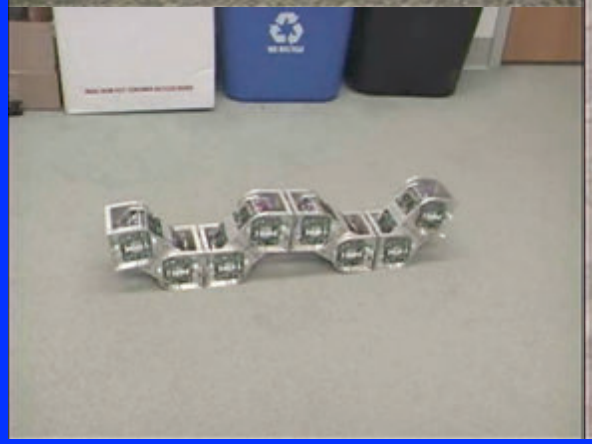






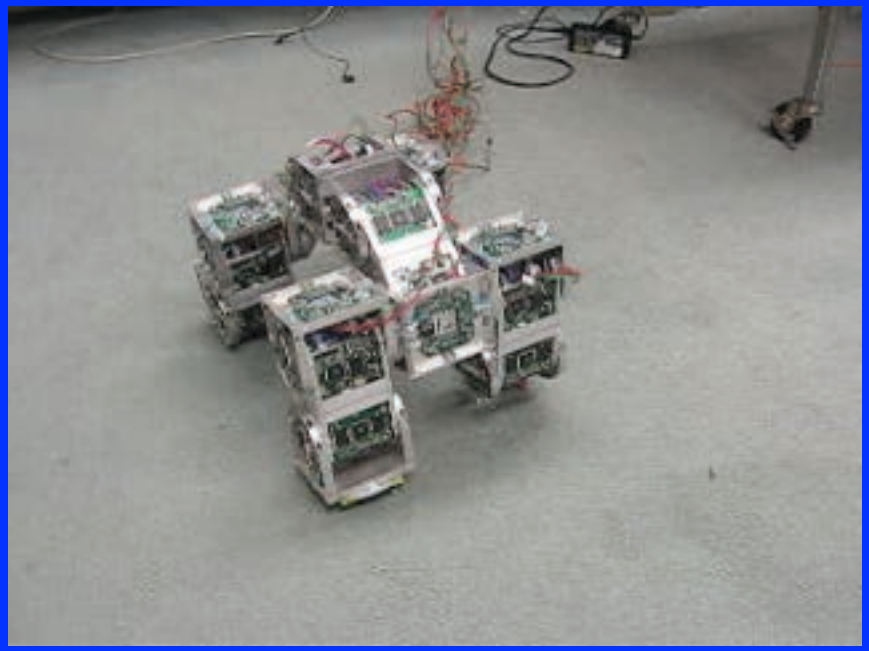
Dr. Wei-Min Shen, www.isi.edu/robots/, shen@isi.edu, 310-448-8710

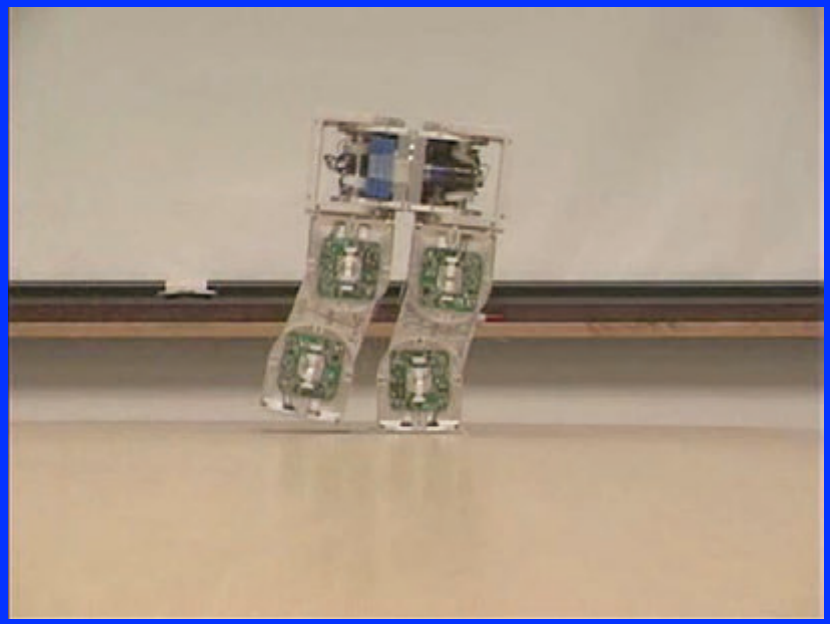
Caterpillars on Beach



Locomotion Forms

Walking with Legs



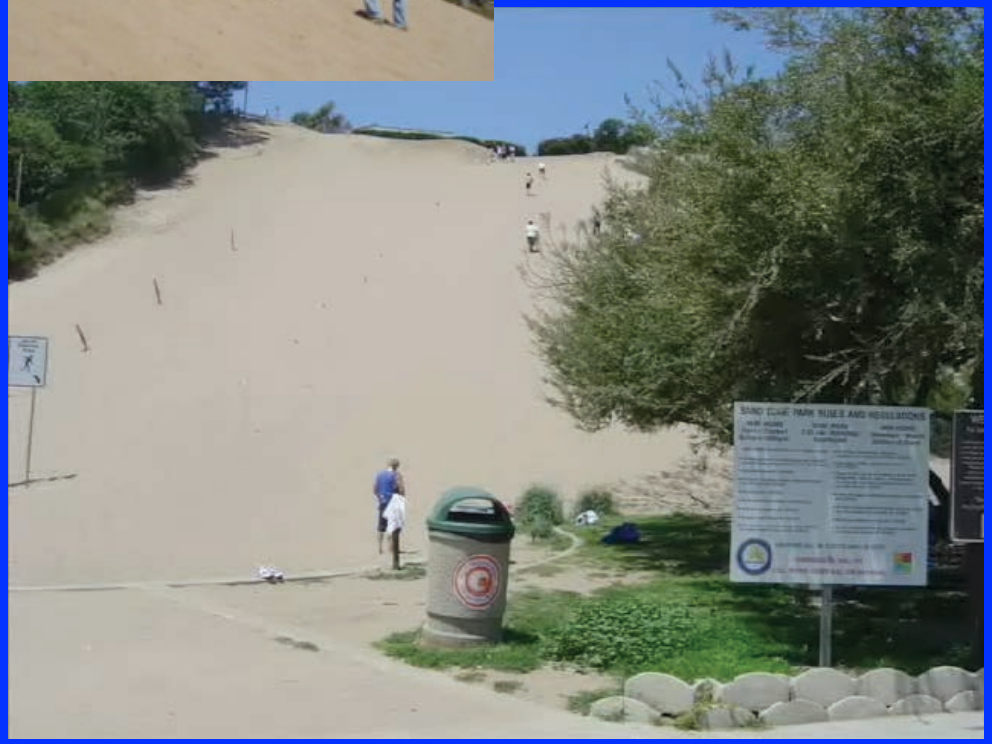


Digging and Burying

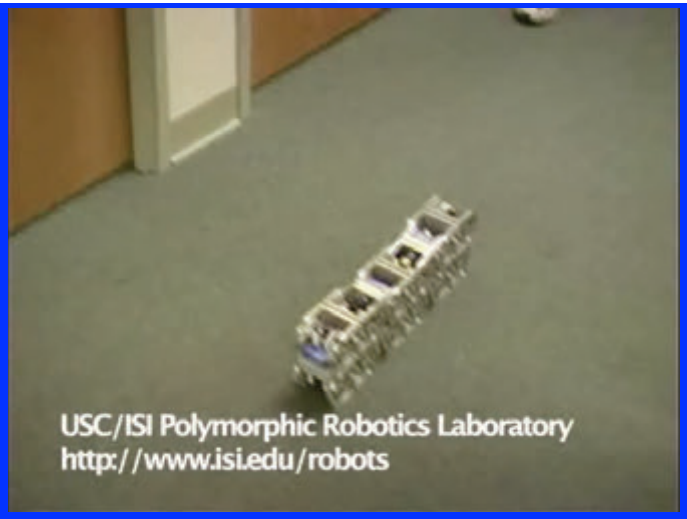
Going through Pipes

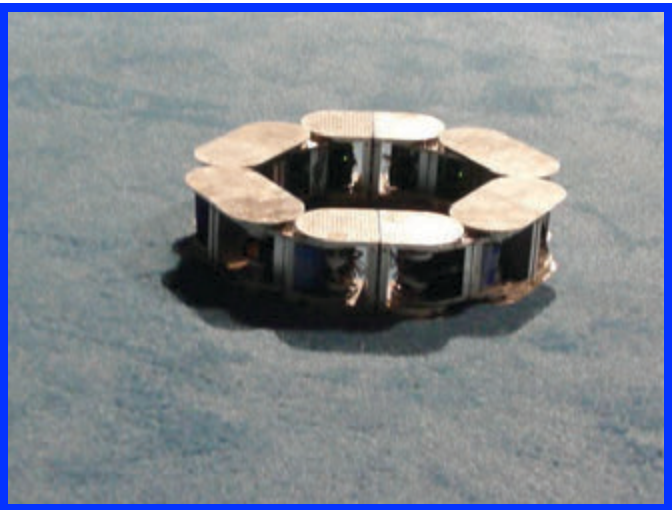
Crawl up Steep Slopes

Sand Dune Experiments



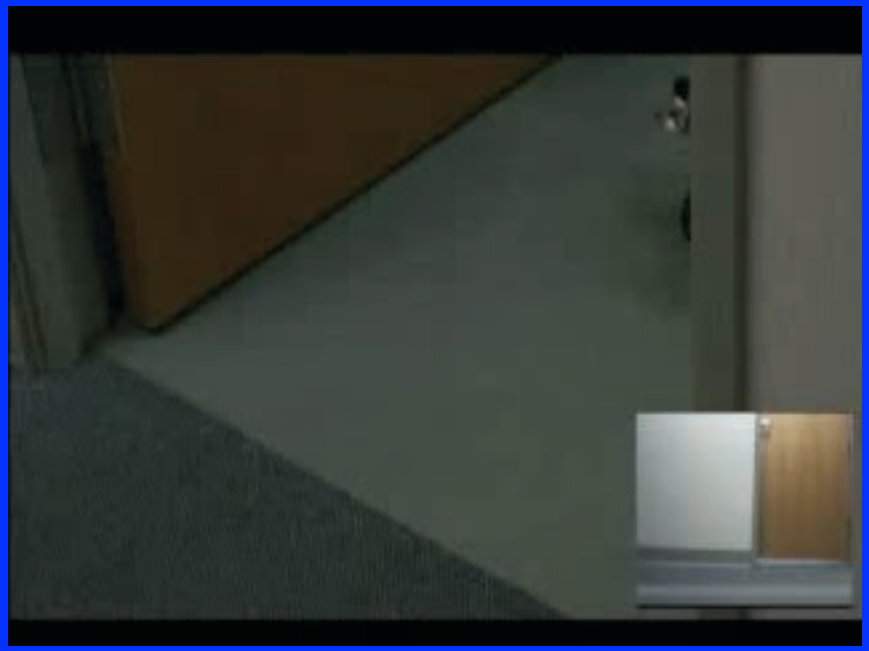
Rolling, Recovering, & Turning

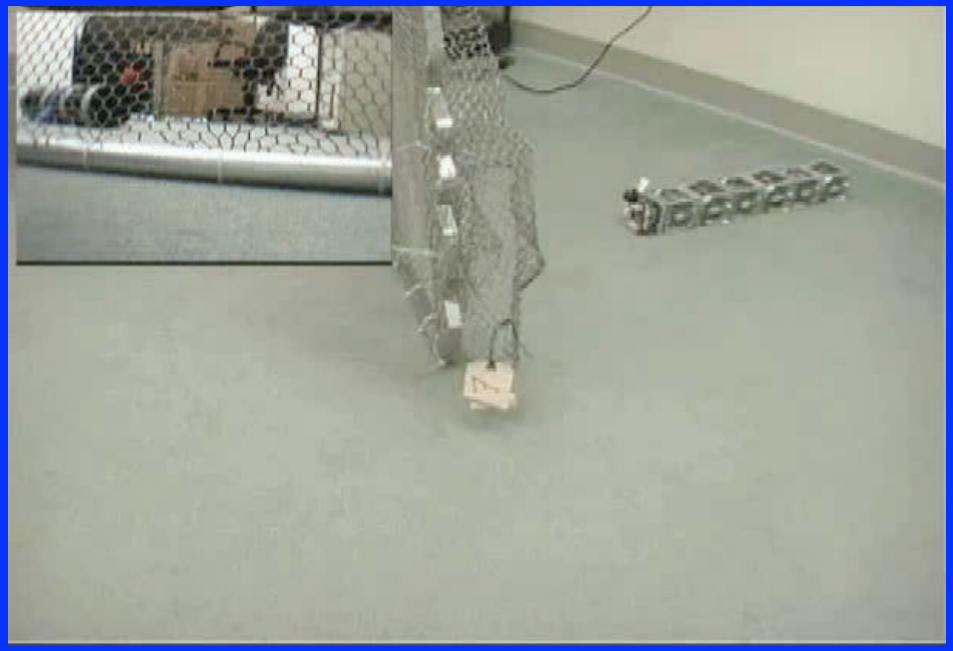




Climbing Ropes

Tele Operations

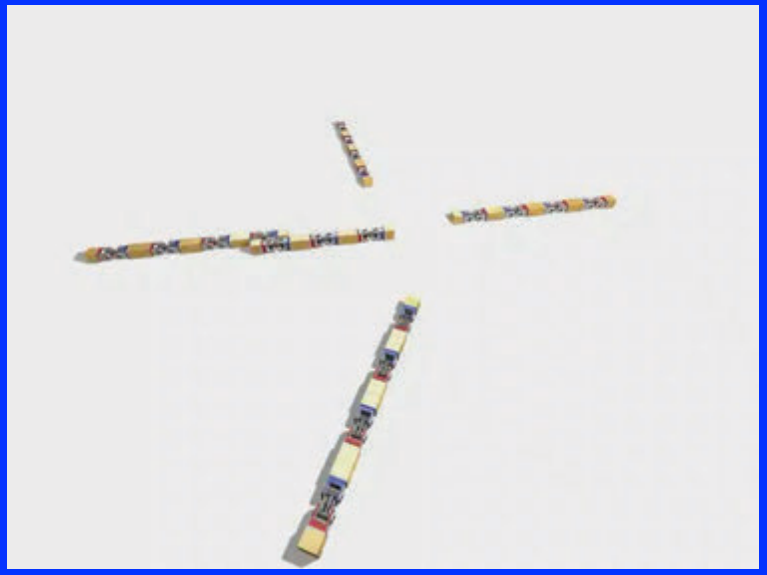




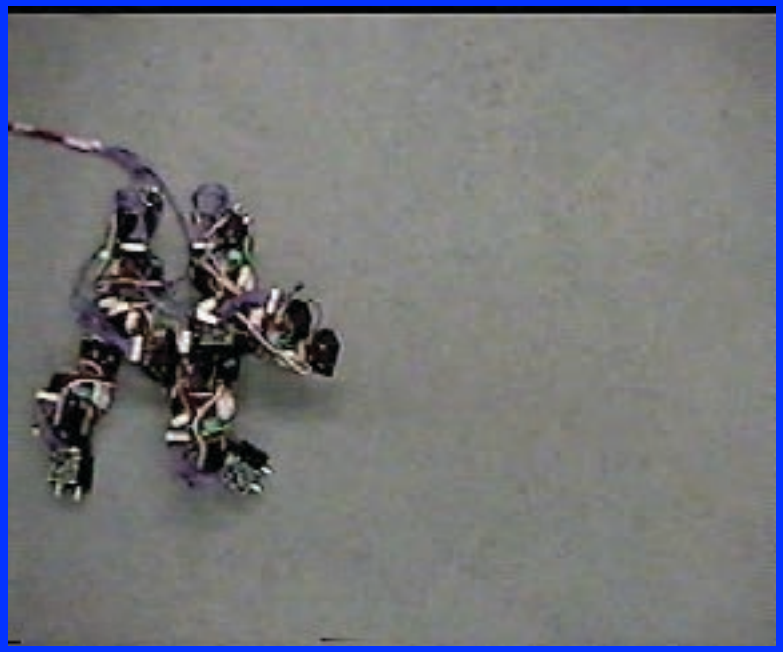
Roller Skates and Payload

Rolling on Disks

Self Reconfiguration



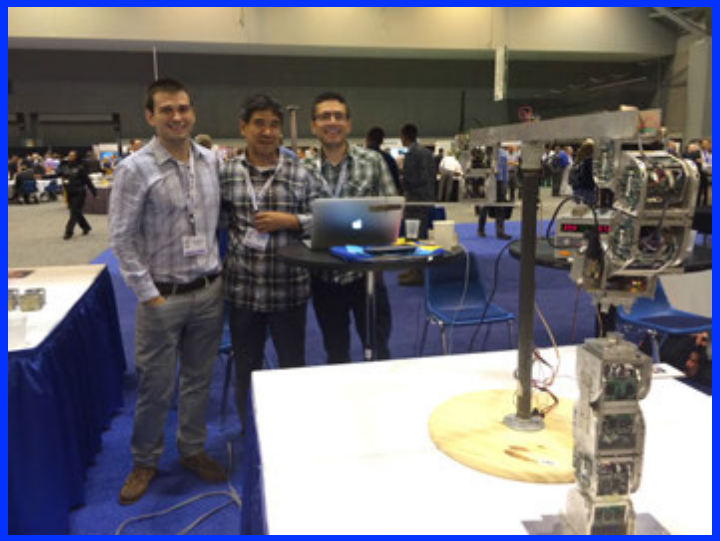
Resilient to Shape Changes

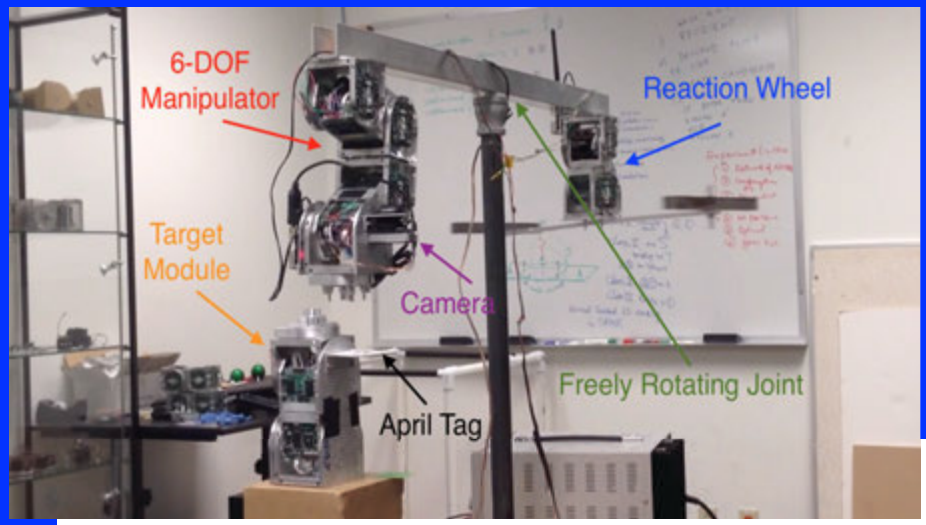


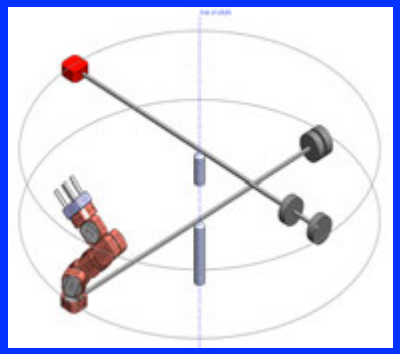
Infrastructure Inspections

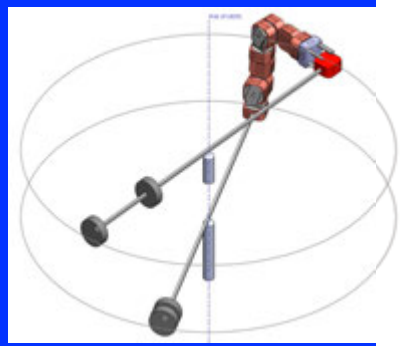
6D Autonomous Docking (in ~0G)

Featured at DARPA Wait-What Forum 2015

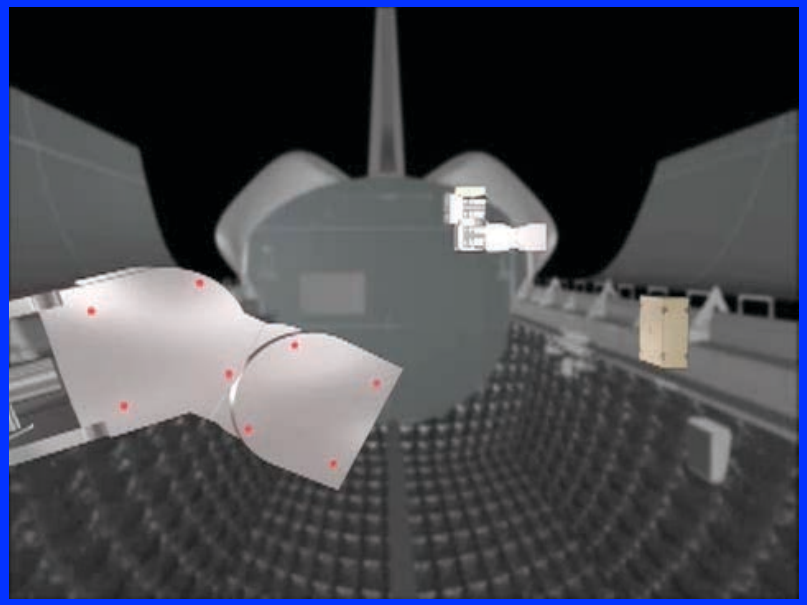


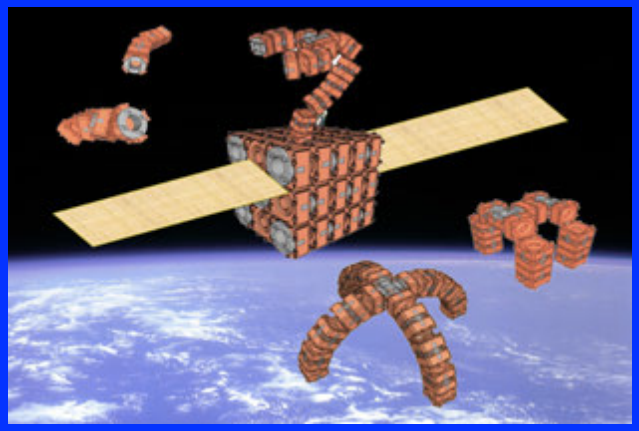






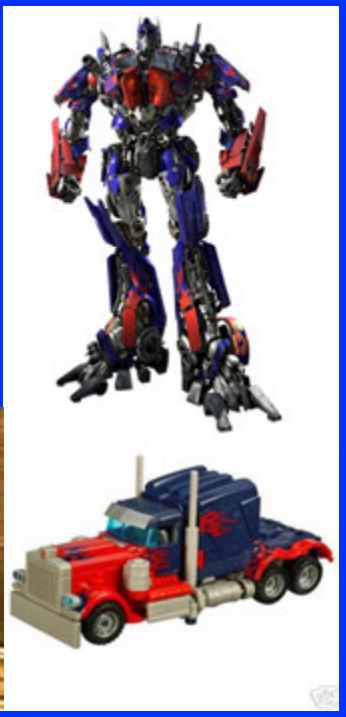
Future Operations in Space





Research Challenges

- How to make modules?
  - Modularity, nano, fluid, ...
- How to control a configuration?
  - One structure but many uses
- How to adapt configurations?
  - Many structures, adaptive



Advantages

Flexibility, Reusability,
Repair-ability, Low-cost, …

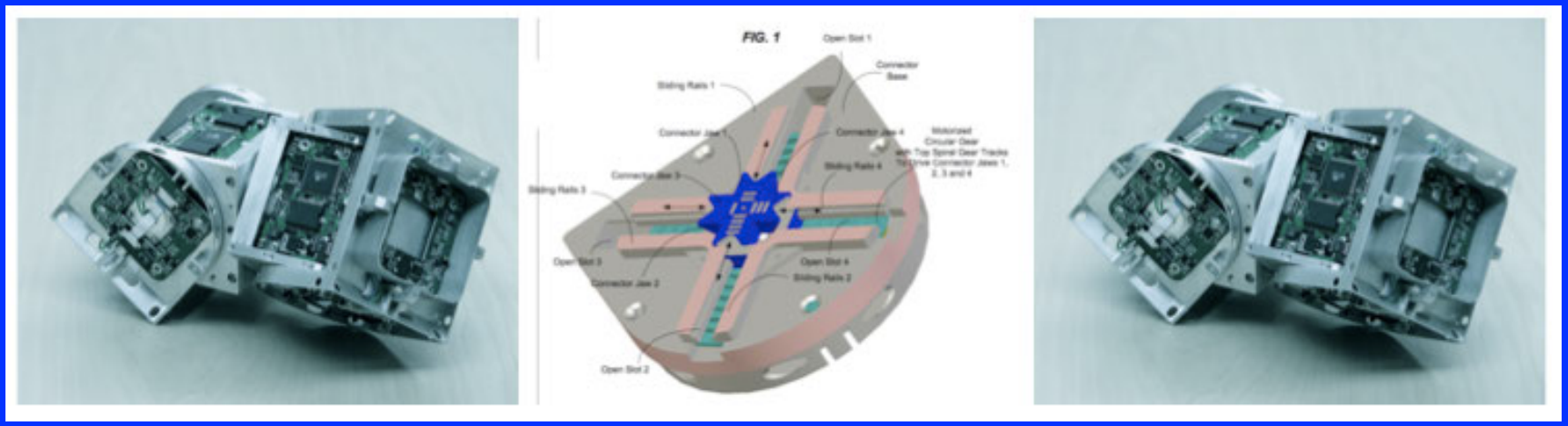
Challenges in Systems

- **Challenges in Mechanics**
  - Modularity
  - Homogeneous and/or heterogeneous modules
  - Flexible, connectable, strong, small/large, …
  - Robust, durable, replaceable, ...
- **Challenges in Electronics**
  - Sensors, actuators, communication, control, interferences, collaboration, ...
  - Real-time operating system
  - Programming environment
  - Debugging facilities
  - Power sharing
  - Integration of all these!
- **Challenges in Connectors**
  - Genderless
  - Single side connectable/releasable
  - Thin but strong profile (no back-drive)
  - Efficient when connected and disconnected
  - Support multi-orientation engagements
  - Tolerant to misalignment
  - Communication
  - *Power sharing*
  - *Dirt tolerant*
- **Challenges in Intelligent Control**
  - Intelligent behaviors
    - Mobility adaptive to tasks/environment
    - Maneuverability adaptive to tasks/environment
  - Adaptive shapes and structures
    - Higher level challenges for AI & Robotics
    - Self-organization
    - Self-healing
    - Reconfiguration deciding, planning, execution
  - New theory for self-reconfigurable systems

Challenges in Connectors

- Genderless
- Strong mechanical torque and endurance
- Data communication
- Power sharing
- Thin profile <15mm
- Dock-able between variable sizes
- Single side operational and releasable
- No back drive
- Efficient power consumption for connect/disconnect
- Multi-orientation engagements
- Tolerant to misalignment during docking process

SINGO Connectors
(US Patent 8,234,950)



A movie of connecting and disconnecting

Challenges in Distributed Control

- Distributed among all modules
  - No “fixed brains” or single-point-failures
  - Free from any unique global names or identifiers
  - Discover topology and negotiate tasks
  - Change behaviors by topological locations
- Dynamic network topology
  - Endure unexpected changes in the network
- Asynchronous
  - Collaborate without any synchronized or global clocks
- Scalable
  - Function with any size and shape
- Adaptive
  - Perform complex tasks in different environments
  - Recover from component failures and self-repair
  - Autonomous learning for unexpected situations and tasks

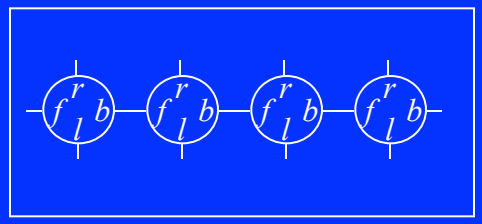
"Digital Hormones" (US Patent 6,636,781)

- Content-based messages
  - No addresses nor identifiers
  - Have finite life time
  - Trigger different actions at different sites
- Floating in a global medium
  - Propagated, not broadcast
  - Different from *pheromones*
    - Internal circulation, not external deposit
- Preserve local autonomy for individual sites
- Could modify module behaviors (as RNA)

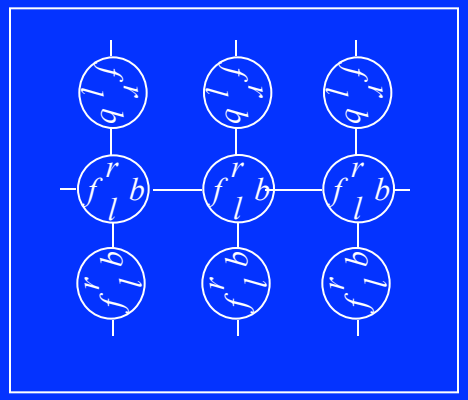
Configuration Representation

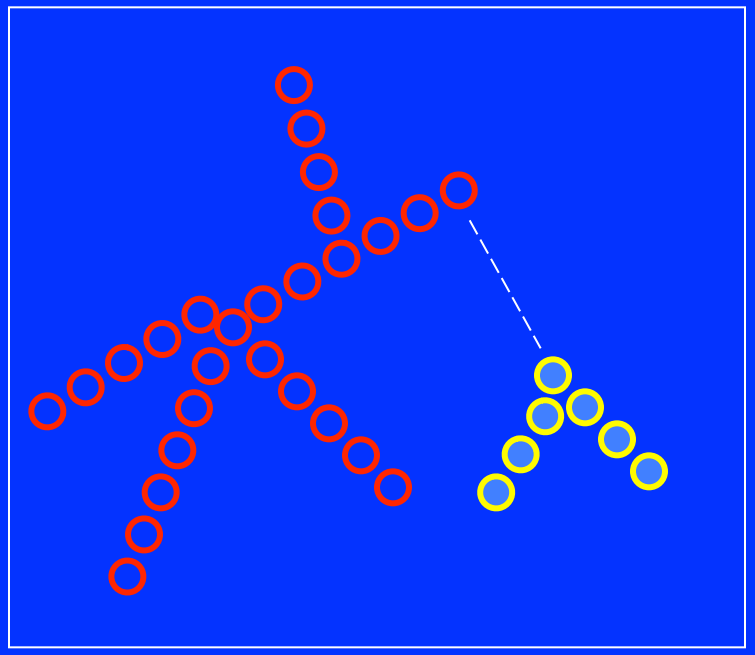
One module

A Snake



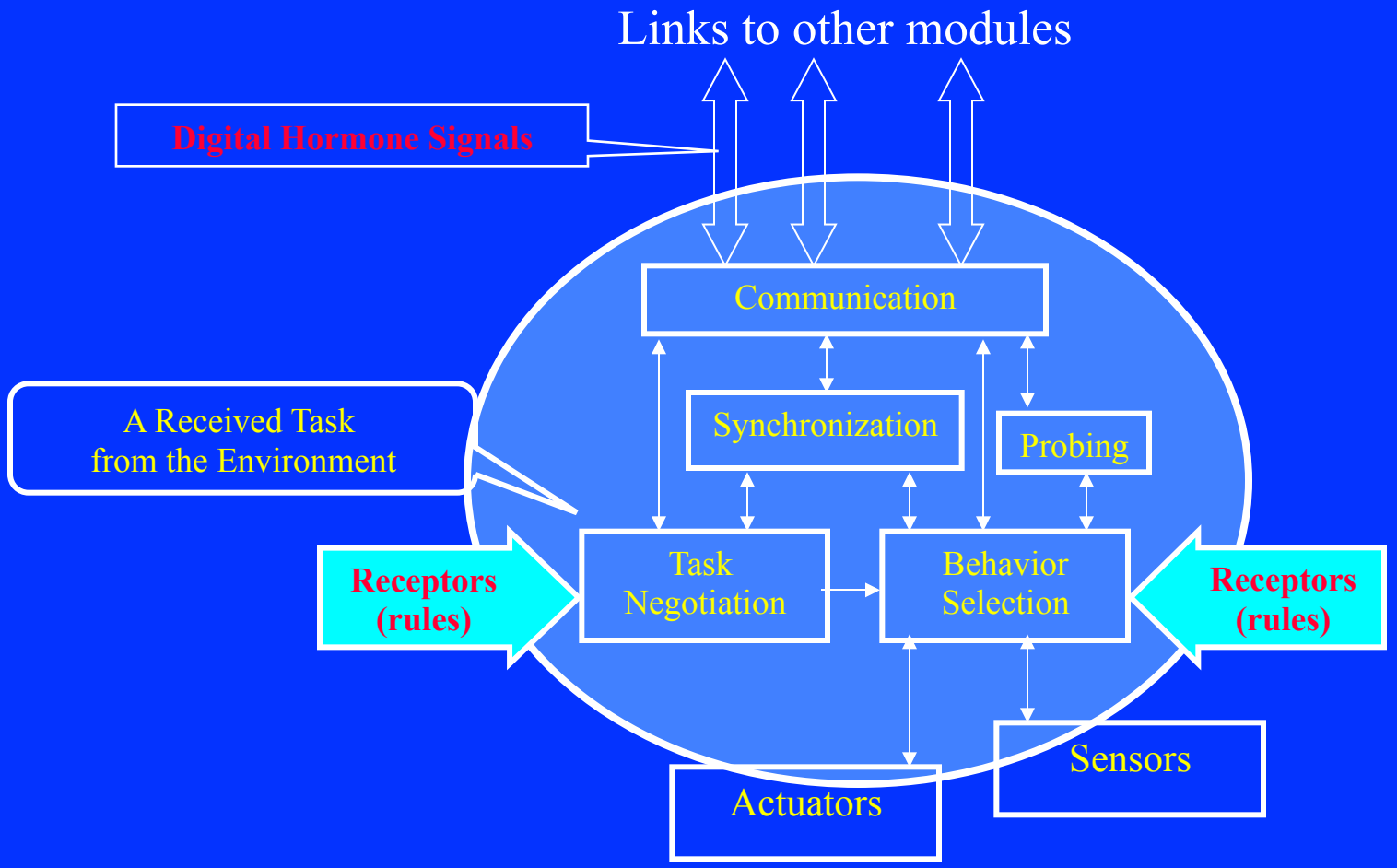
A 6-leg insect





Communication between two separate creatures

Internal Control Architecture

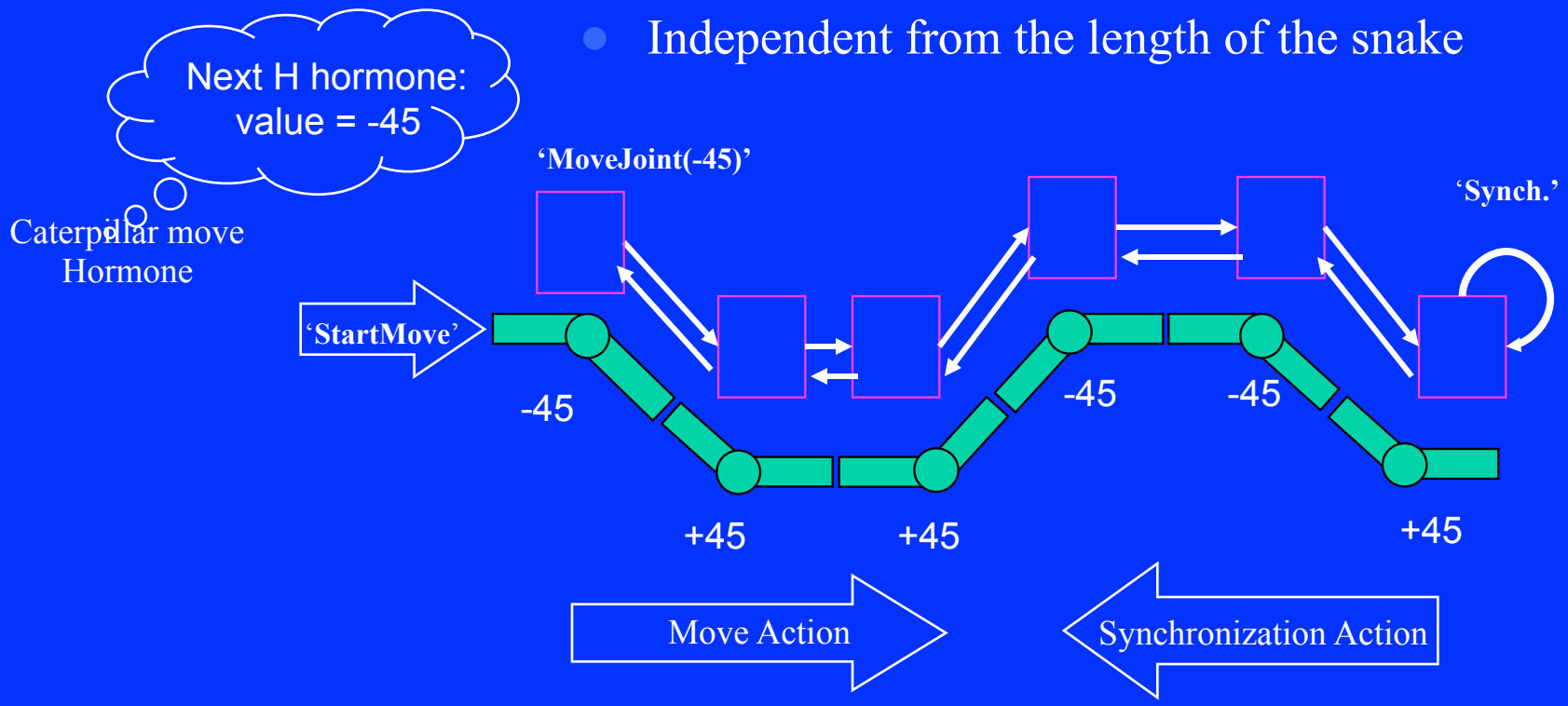


The Uses of Digital Hormones

- Communication in dynamic network
- Cooperation of autonomous robotic modules
  - Locomotion
  - Reconfiguration
  - Synchronization
  - Global effects by weak local actions
  - Conflict resolution (multi hormone management)
  - Navigation
- Configuration adaptation (vision, telescope, ...)
- Self-healing systems

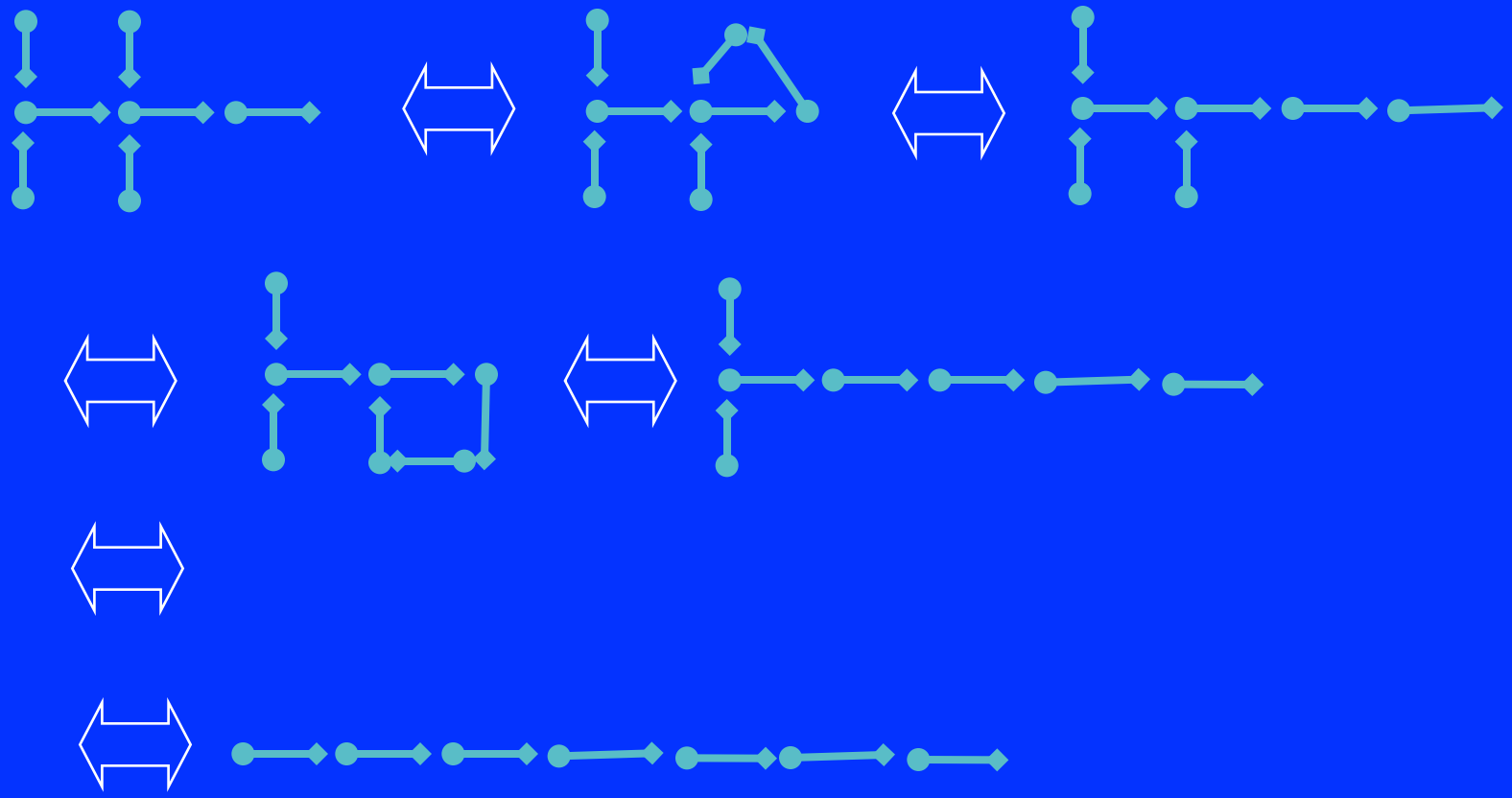
Hormones for Caterpillar Move

- A simple one-pass hormone from head to tail
- Controls and synchronizes all motor actions
- Independent from the length of the snake

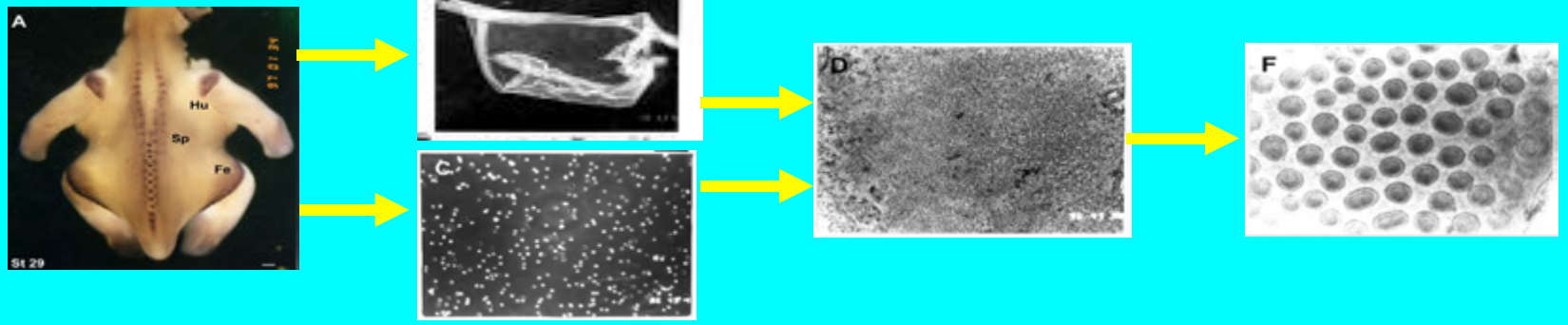


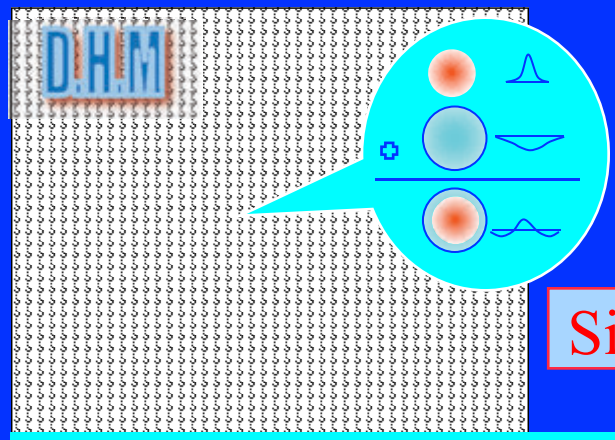
Hormones for Reconfiguration

Insect ←→ Snake

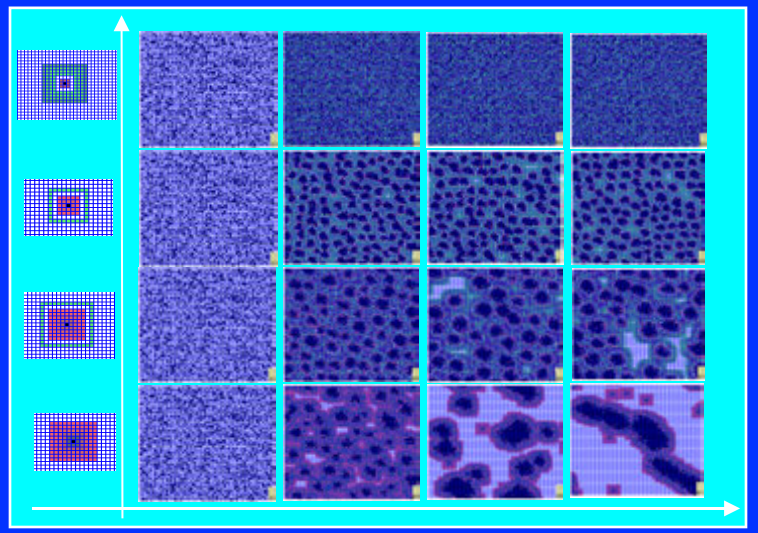


Hormones in “Feather Formation”



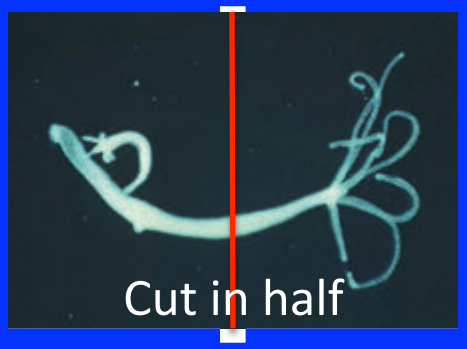


Simulation



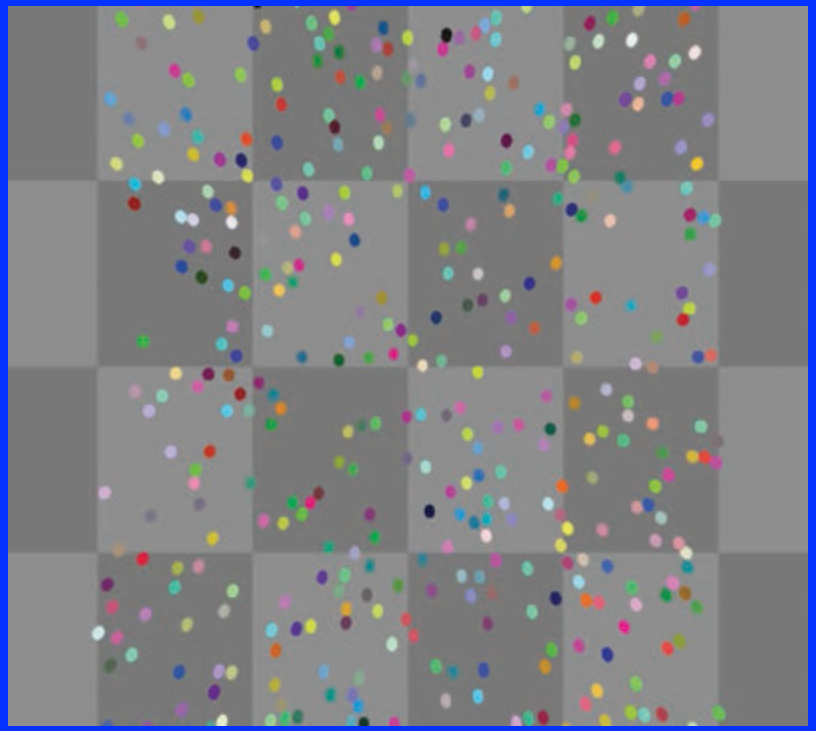
1. Reconfigurable networks
2. Reaction-Diffusion model
3. Stochastic action selection

Scalable Self-Healing

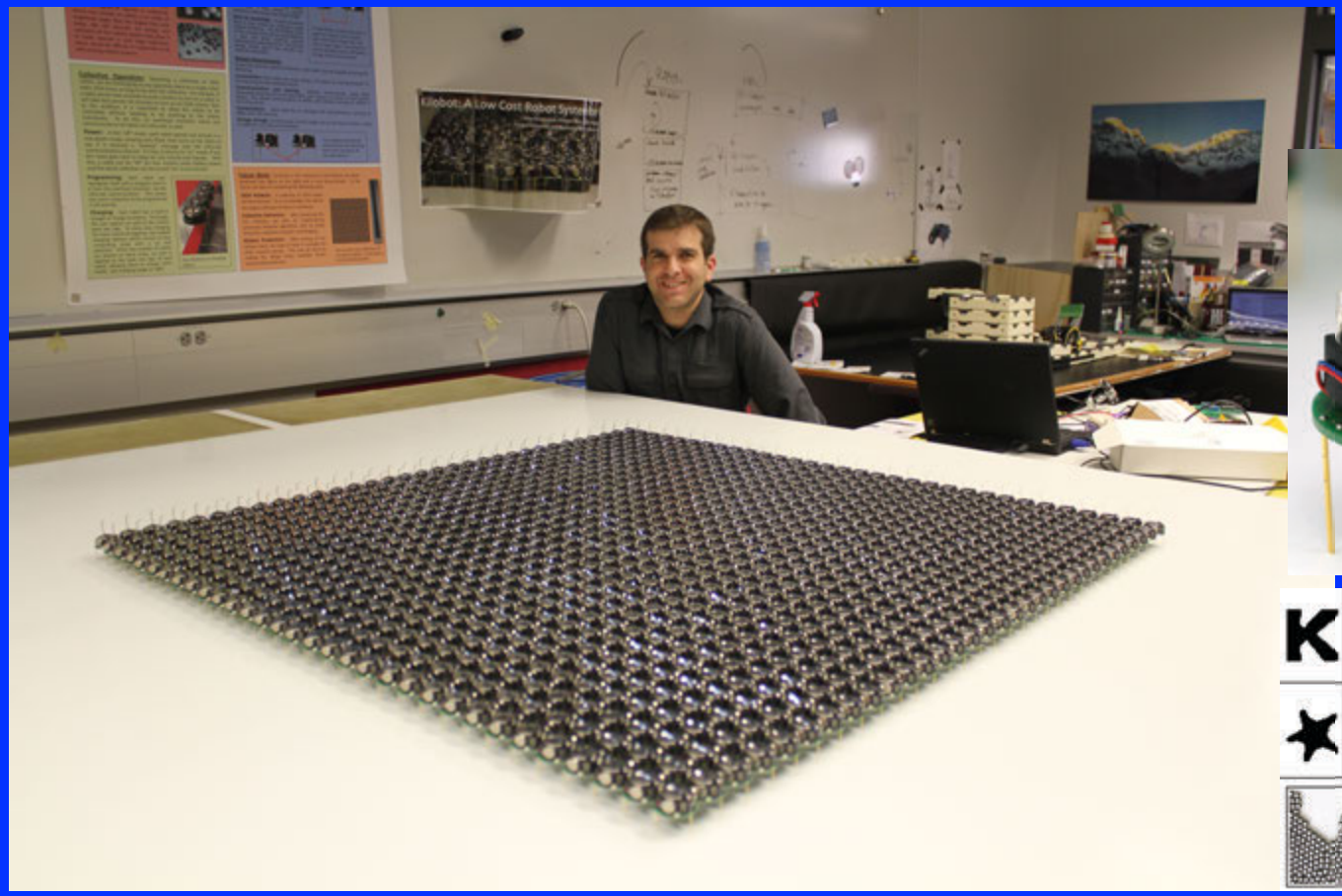


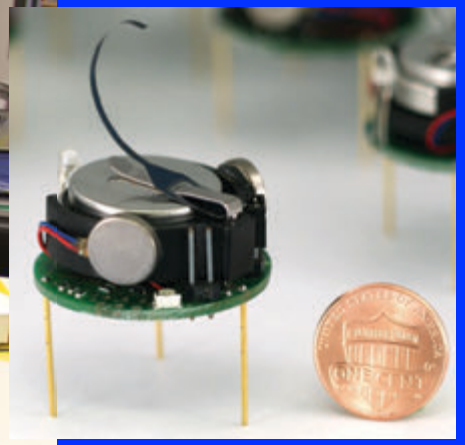
Self-Healing

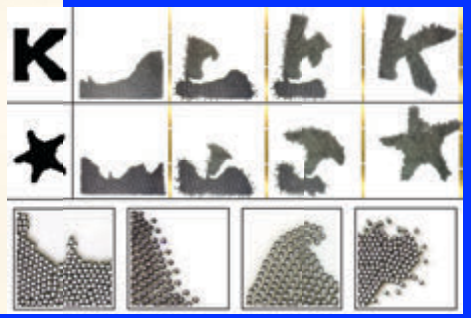
Self-Healing



Robotic Swarms



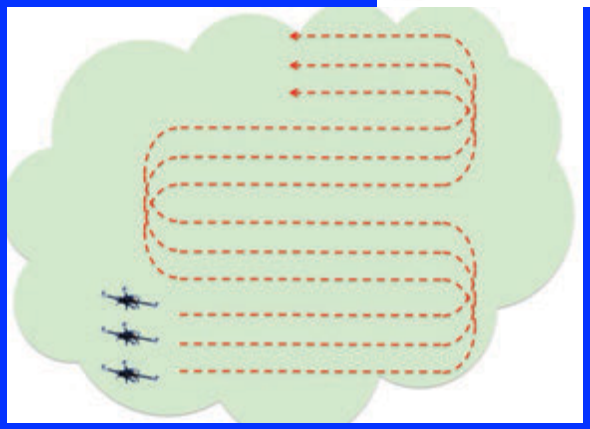


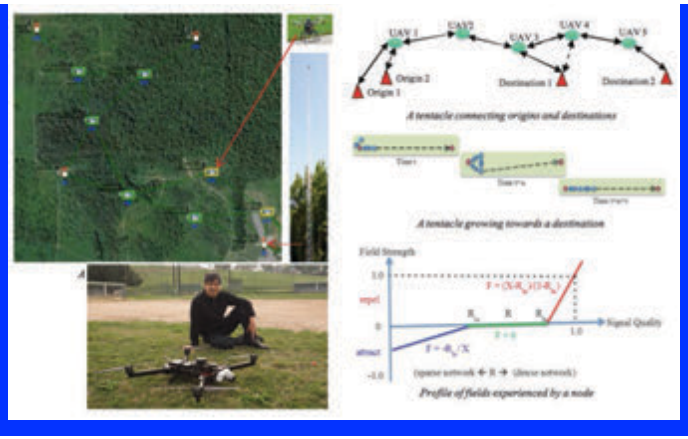


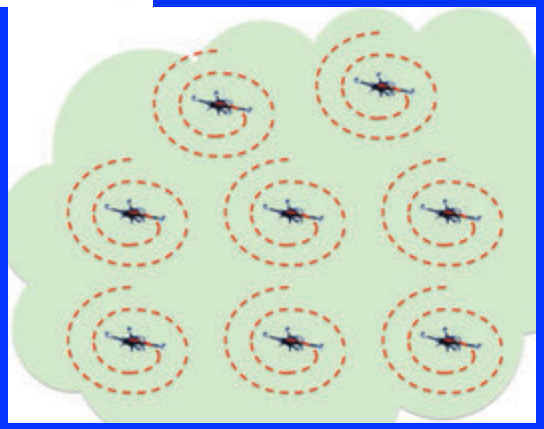
UAV Swarm and Applications

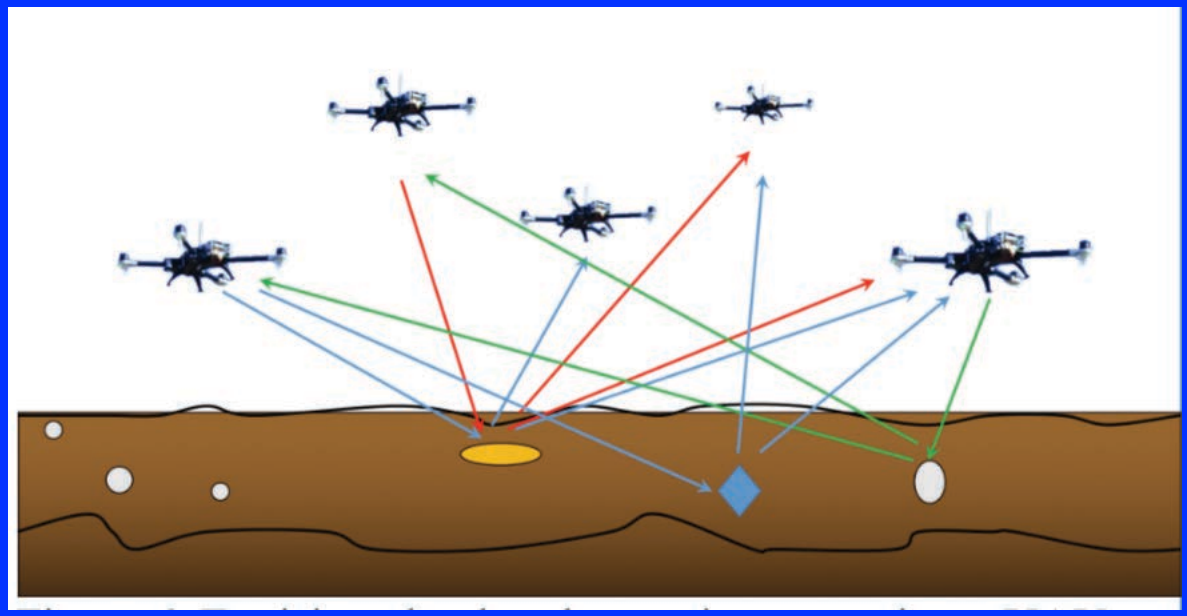
UAV Swarms

Mine Detection by UAV Swarms
(Masri, Moghaddam, Shen)

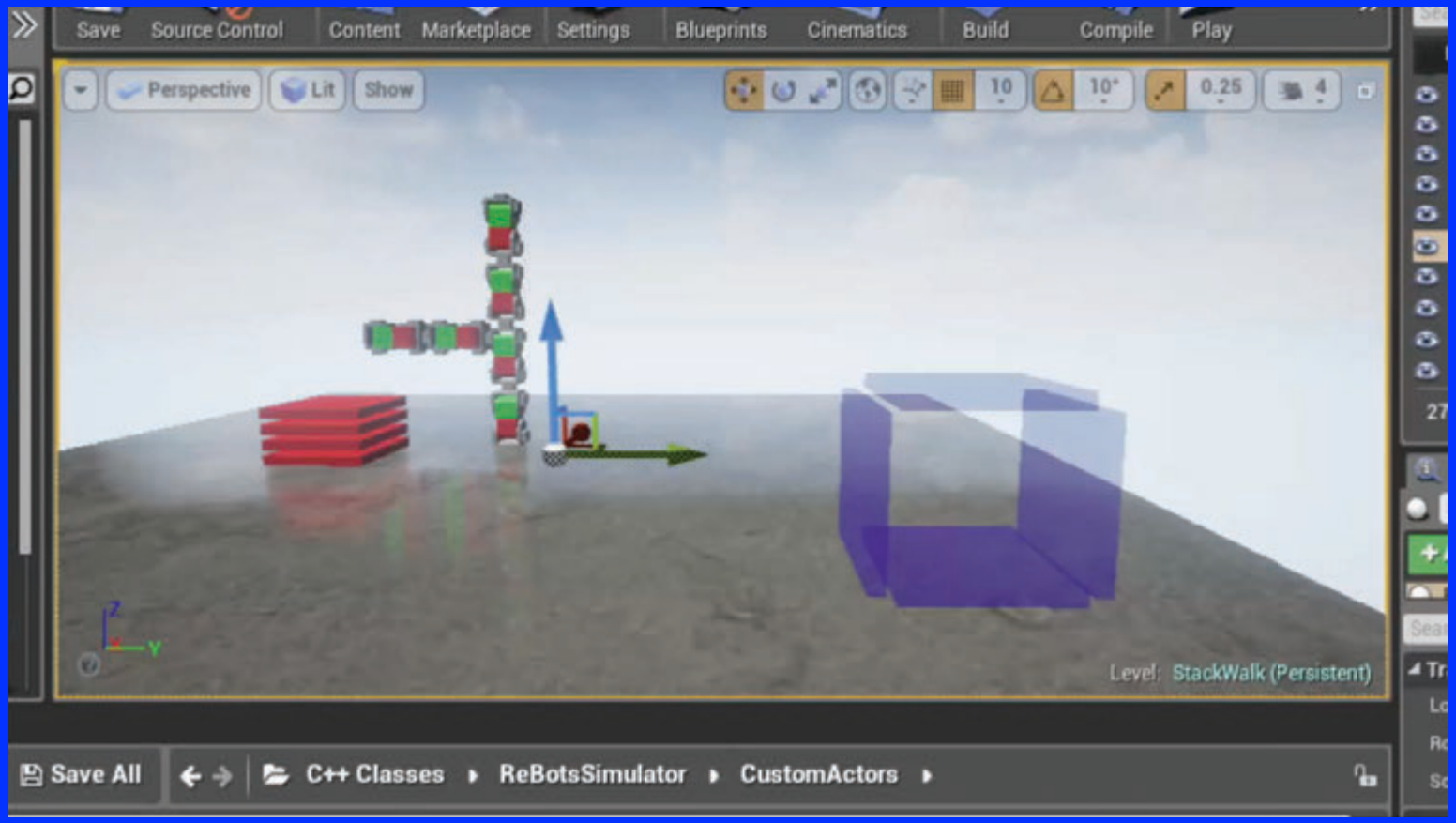








Self-Assembly

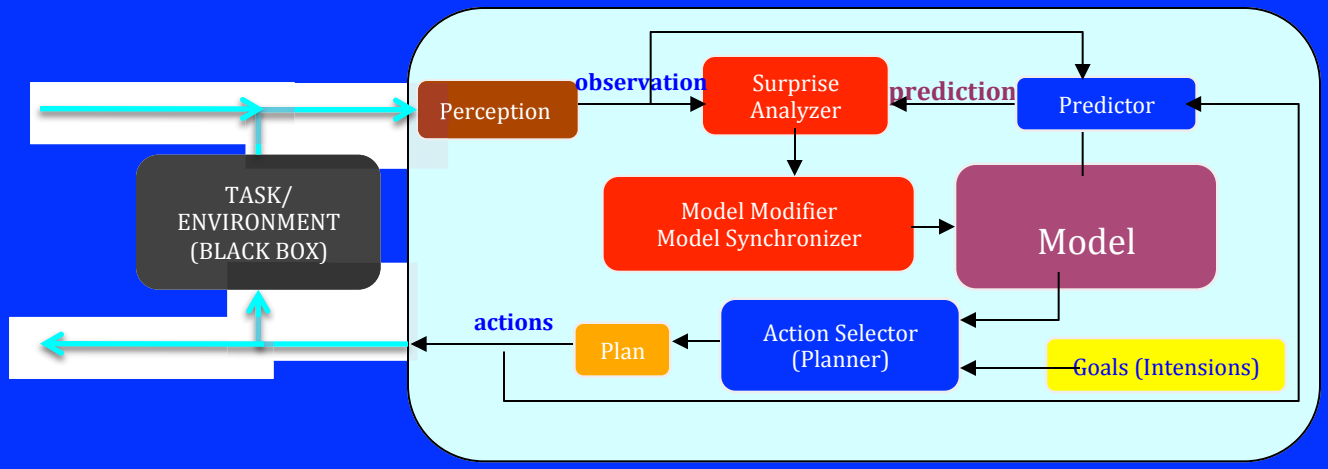


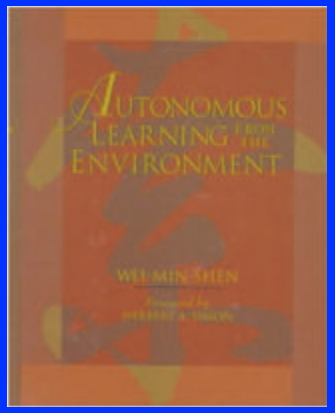
Very-Large Physics-Based Simulation

Surprise-Based Learning

- New Environment/Task → Learning → Knowledge/Skill
- Key Idea: To detect and learn from "surprises"
  - Adaptive to new environments
  - Learn to accomplish new tasks
  - Self-heal from unexpected failures or dynamics (e.g., inverted visions)
    - Know-how → Surprises → Learn → Recovery

Surprise-Based Learning





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

Surprise ==> Model ==> Prediction

Related Research Topics

- Biological morphgenesis
- Self-organizing swarm robots
- Self-healing (scalable) systems
- Underwater robot swarm
- Self-assembly in space
- Oil and energy robotics
- Self-configureble network
- Evolution of brains

Conclusions

- Self-Reconfigurable Robots Are Coming!
  - Fundamental research challenges
    - Physics, chemical, biology, structures, robots, networks, agents, ...
  - Offer critical capabilities for many applications
    - Search/rescue, homeland security, self-assembly, smart structures, space, ocean, deep-water, oil production, …
  - Has many future potentials
    - Autonomous, Resilient, Adaptive, and Learn
    - Intelligently selection of configurations (very challenging)

CSCI 561
Foundation of Artificial Intelligence

Professor Wei-Min Shen
(Dr. Nadeesha Ranasinghe)

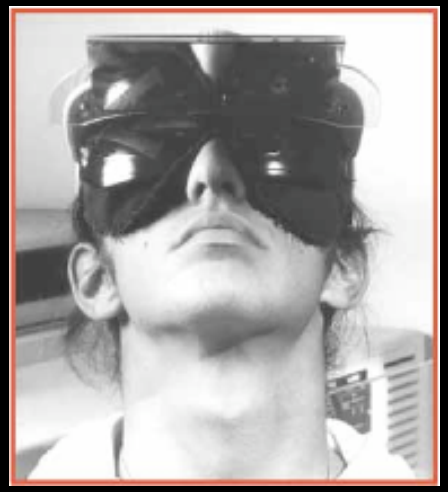
Session 25-1: Surprise-Based-Learning-1

Unsupervised Learning

- 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 have internal structures (not just “symbols”)
  - States are not known and must be learned

Why Surprise-Based Learning?

- New Environment/Task → Learning → Knowledge/Skill
- Key Idea: To detect and learn from "surprises"
  - 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)
    - Know-how → Surprises → Learn → Recovery



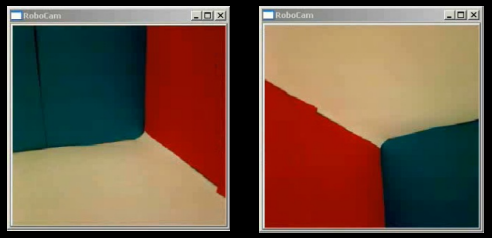
Technical Challenges (why hard?)

- 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

Potential 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



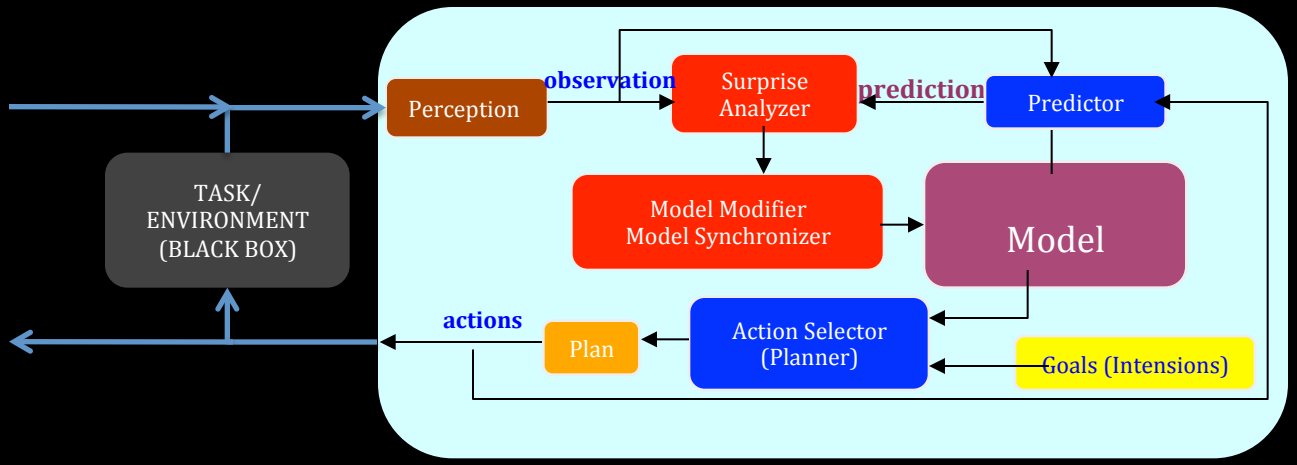
Rotated camera

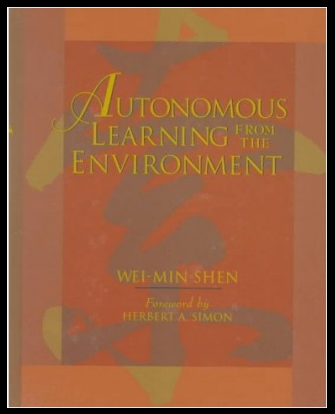
Gap

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

Surprise-Based Learning (CDL++)



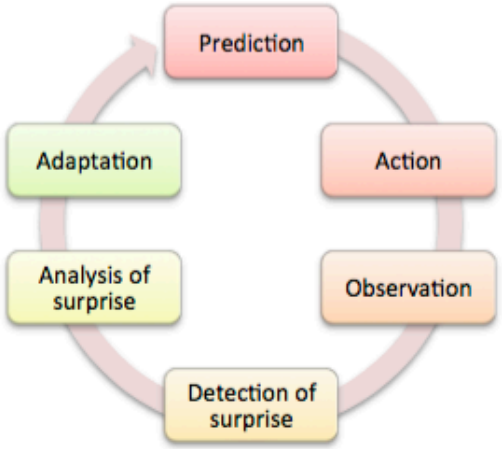


- The Learner continuously makes predictions, detects surprise, analyzes surprises, extracts critical information from surprises, and improves and use its action models

Surprise ==> Model ==> Prediction

The Cycle of SBL (CDL++)

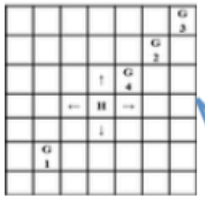
- Autonomously learn Predictive Rules from experience
  1. Select Actions for the current goals
  2. Make Predictions before actions
  3. Perform actions
  4. Observe events and entities via sensors
  5. Detect Surprises
  6. Analyze and explain Surprises
  7. Adapt or create prediction rules



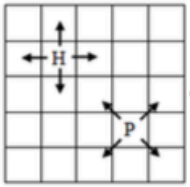
- **Detecting Surprises**
  - When a prediction does not match the actual outcome of an action
- **Explaining Surprises**
  - Finding the differences of a normal satiation and the unexpected situation
- **Revise Prediction Models**
  - Use the found differences to revise the relevant prediction rules
- **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, …, a_n, p_n]
  - Represent concepts such as plan, experiment, example and advice
  - Prediction sequences can capture temporal relations in observation sequences

Model Representation

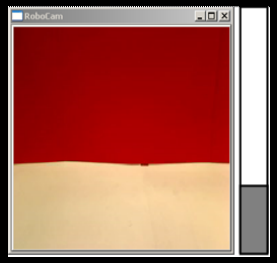
- Prediction Rules
  - [C,A]→P, e.g., [Raining, Step-out] → Get-Wet
  - In the form of probability **P**(R|C,A), where R is the prediction
  - Different from C→A (production rules or action policy)
- Prediction Sequences
  - PS = $(S_0, A_0, P_1, A_1, P_2, A_2, \ldots, A_n, P_n)$
- The multifunction of prediction sequences
  - Plan: if P_n is a goal
  - Exploration: if P_n guarantees a surprise
  - Experiment: if P_n may generate a surprise
  - Example: if the sequence will generate some surprises
  - Advice: 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 $O_t \nvDash P_t$

Background and Related Work

- Shen-Simon89, Shen92, Shen94(book), Nadeesha-Shen2011, Nadeesha2012, Collins-Shen 2017, 2018
- Three different attitudes towards unpredicted changes [Saffiotti97]
  - Get rid of it: with precisely engineered robots, e.g. industrial robots
  - Tolerate it: using redundancy with carefully designed contingency strategies
  - Reason about it: with techniques for the representation and manipulation of uncertain information
- Tolerate it
  - Detect changes using models of sensors, actions and environments
  - Sensors: Model is given [Visinsky91] [Roumeliotis98] or learned from examples [Murphy96] [Stronger06]
  - Actions: [Ferrell94] [Kececi08]
  - Goals: [Wolpert98] [Doya02]
  - The set of contingency strategies may not cover all unpredicted changes
- Reason about it = data-driven adaptation
  - Sensors: Sensor fusion [Pierce97] [Soika02]
  - Actions: [Lipson04] [Pierce97] [Bongard06] [Mugan10]
  - Goals: Update model & re-plan [Yildirum99] [Busch07] [Stone08]
  - None focus on simultaneous unpredicted changes to sensors, actions, goals & environment in an unsupervised manner

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 alention on the relevant features
  - Seek differences that are statistically significant

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)

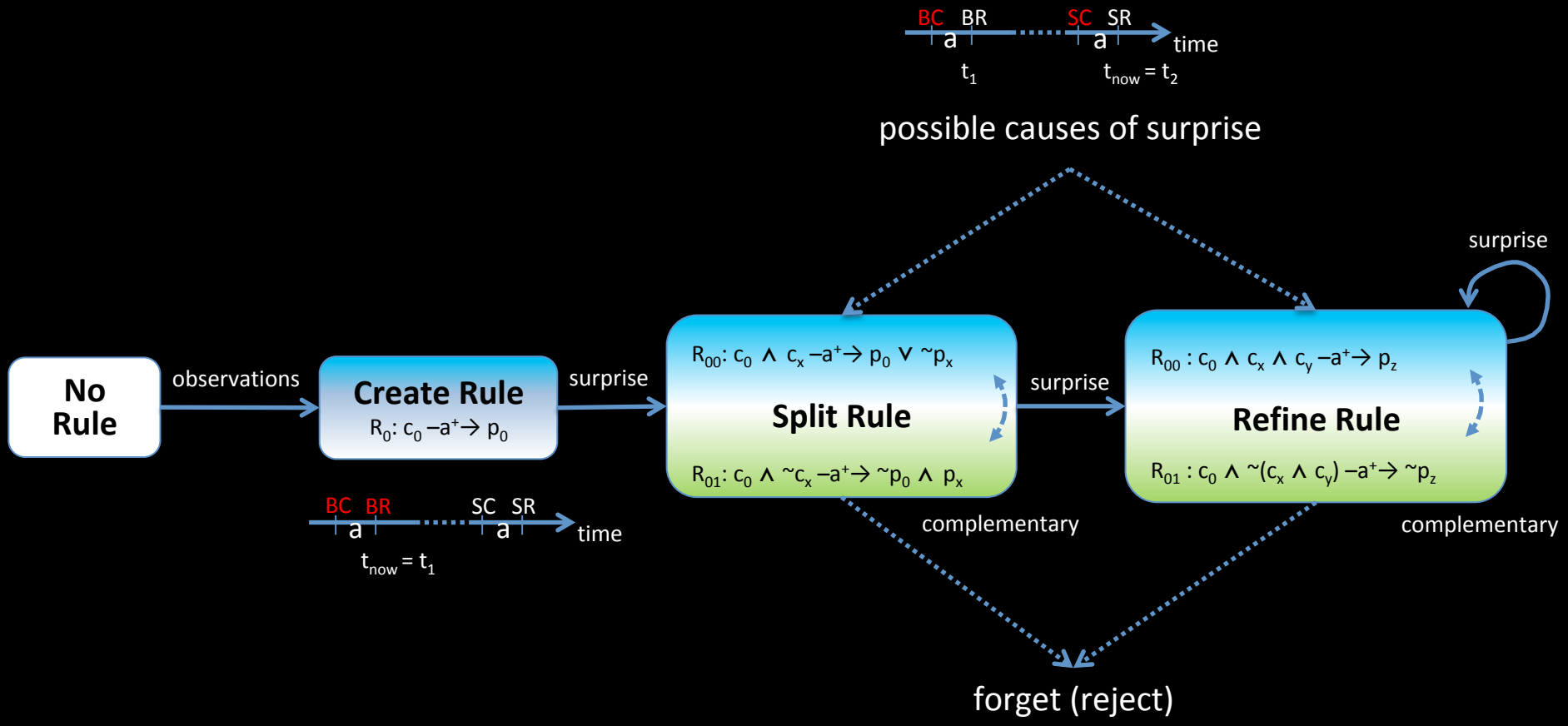
Desirable Features

- 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”

Previous Results

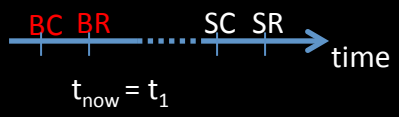
- Discrete or symbolic environments
  - Developmental learning to use novel tools
  - Scientific discovery of hidden features (genes)
  - Game playing
  - Learning from large knowledge bases
- Robotics environments
  - Sensor recovery in office environment
  - Recognize/predict visual acDons from video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

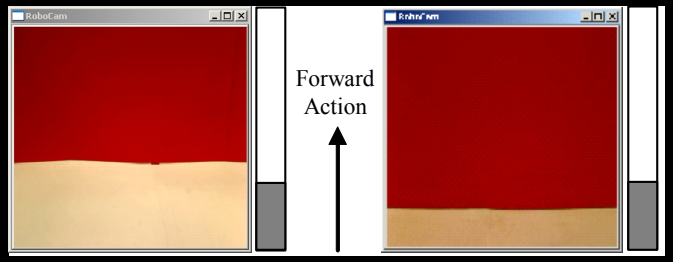
Life Cycle of a Prediction Rule



Creation of New Prediction Rules

- When to create rules?
  - No predictions exist
  - No change indicated in all predicted rules (general rules)
  - All predicted rules have been rejected
- How to create rules?
  - Base-condition 'BC' = observation made at time t_1-1 before action was executed
  - Base-result 'BR' = observation made at time t_1 after action was executed
  - Compare BC and BR with operators {%, ~, ↑, ↓} & create a rule for each difference

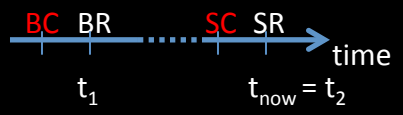


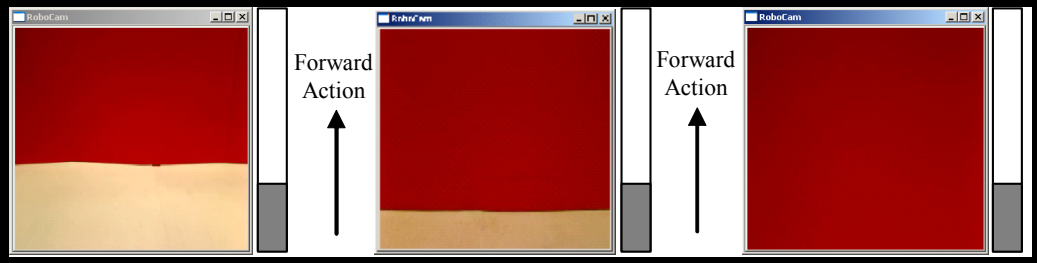


  - e.g.
    - RuleR1: (White, All, %, 0) → FORWARD → (White, S, <, Value1) // size of white decreased
    - RuleR2: (White, All, %, 0) → FORWARD → (White, Y, >, Value2) // y-location of white increased
    - RuleR3: (Red, All, %, 0) → FORWARD → (Red, S, >, Value3) // size of red increased
    - RuleR4: (Red, All, %, 0) → FORWARD → (Red, Y, >, Value4) // y-location of red increased

Surprise Detection & Analysis

- Detect surprises by comparing the prediction against the observation
- Surprise analysis identifies possible cause(s) of the surprise
  - Surprised-condition 'SC' = observation made at time t_2-1 before action was executed
  - Surprised-result 'SR' = observation made at time t_2 after action was executed
  - Compare the current SC to a previous BC where the rule succeeded, with the given operators (typical set {%,~,>,<,=,!=})
  - Return a set of possible causes [c_1, …, c_z]
    - Each cause is an expression of an entity or attribute that was true in BC but false in SC
    - SBL assumes that a good cause can be found, no hidden states [Shen94]

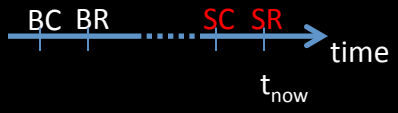




  - e.g.
    - Causes of surprise for RuleR1 = [(Red, S, <, Value5), (Red, Y, <, Value6), (White, S, >, Value7), (White, Y, <, Value8)]

Splitting Prediction Rules

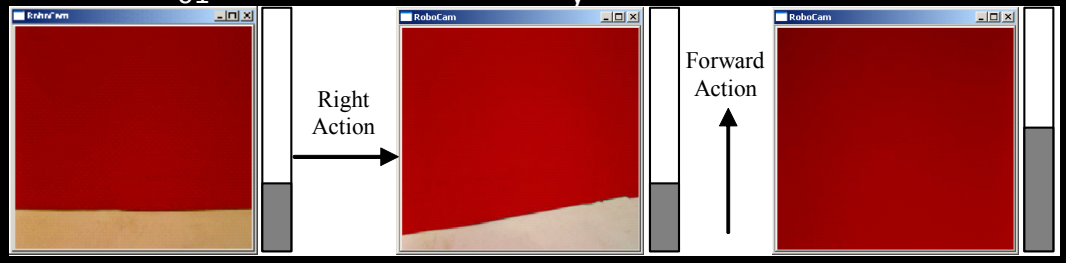
- When to split rules?
  - A surprise occurs in a single rule for the first time
    - Single rule R_0: $c_0 \text{ –}a^+\text{→ } p_0$
- How to split rules?
  - Get possible causes C = [c_1, …, c_z] from surprise analysis
  - For each cause c_x identify the expected change p_x from SC and SR
  - Create a complementary pair of rules
    - Specialization R_{00}: $c_0 \wedge c_x \text{ –}a^+\text{→ } p_0 \vee \sim p_x$
    - Generalization R_{01}: $c_0 \wedge \sim c_x \text{ –}a^+\text{→ } \sim p_0 \wedge p_x$



  - e.g. surprise analysis for RuleR1 returns 4 possible causes and 3 new consequences resulting in 12 pairs of rules:
    - RuleR1.1.1: (White,All,%,0) ∧ (Red,S,<,Value5) → FORWARD → (White,S,<,Value1) ∨ ¬(Red,S,>,Value9)
    - RuleR1.1.2: (White,All,%,0) ∧ ¬(Red,S,<,Value5) → FORWARD → ¬(White,S,<,Value1) ∧ (Red,S,>,Value9) …

Refinement of Prediction Rules

- When to refine rules?
  - Any further surprises occur in a complementary rule
    - R_{00}: $c_0 \wedge c_x \rightarrow^{+a} p_0 \vee \sim p_x$
    - R_{01}: $c_0 \wedge \sim c_x \rightarrow^{+a} \sim p_0 \wedge p_x$
- How to refine rules?
  - Get possible causes C = [c_1, …, c_z] from surprise analysis
  - When R_{00} is surprised, for each cause c_y that is not specified in c_0 or c_x
    - R_{00} : $c_0 \wedge c_x \wedge c_y \rightarrow^{+a} p_z$
    - R_{01} : $c_0 \wedge \sim(c_x \wedge c_y) \rightarrow^{+a} \sim p_z$
  - If no new cause, add the entire observation (e_1, … , e.b.v_n)
  - Similarly, when R_{01} is surprised, add c_y to its conditions



  - e.g. by RuleR2.1.1 a forward action predicts that the y-location of white will increase, but it disappears, leaving 6 causes:
    - RuleR2.1.1.1.1: (White,All,%,0) ∧ (Red,S,<,Value5) ∧ (Red,X,>,Value12) → FORWARD → (White,Y,>,Value2) v¬(Red,S,>,Value9)
    - RuleR2.1.2.6.2: (White,All,%,0) ∧ ¬((Red,S,<,Value5) ∧ (White,S,>,Value17))→ FORWARD →¬(White,Y,>,Value2) v(Red,S,>,Value9) …

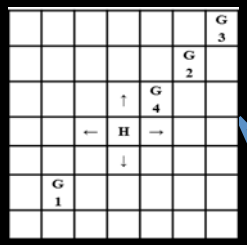
Dealing with Unpredicted Changes

- 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

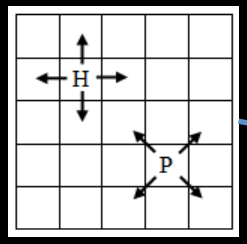
Simulated hunter-goal

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

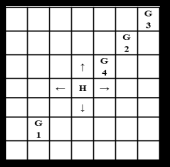
Sensors: tracks
Entities: actors, objects, rel. distance
Attribute: s, x, y, velocity
Actions: 12 verbs
Goal: seen=learned verb

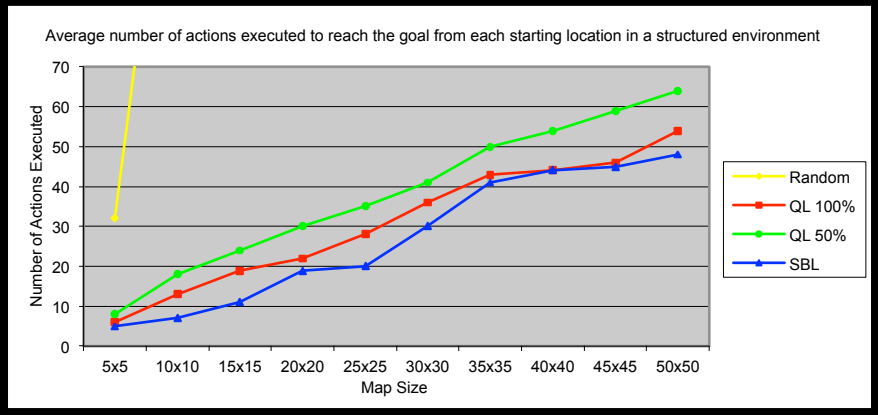
Contributions

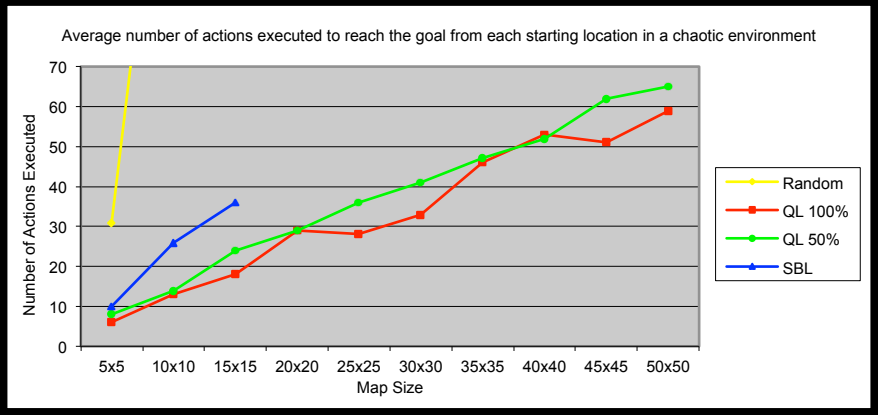
- Structure Learning
- Learning from Uninterpreted Sensors & Actions
- Detecting and Adapting to Unpredicted Changes
- Detecting and Reasoning with Interference

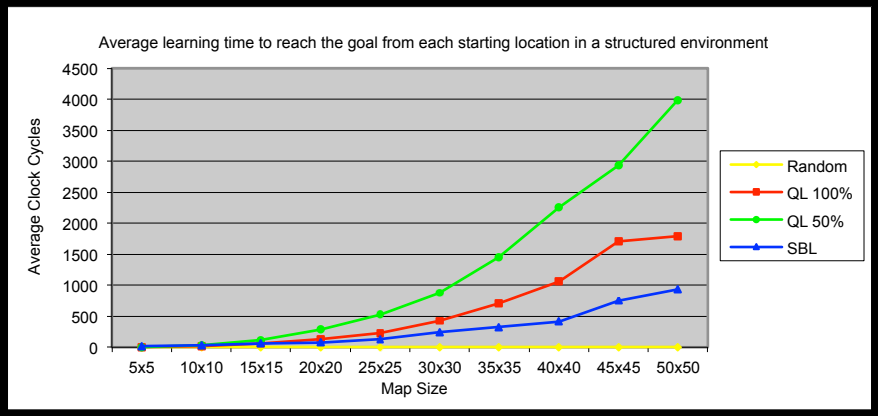
Structure Learning Results

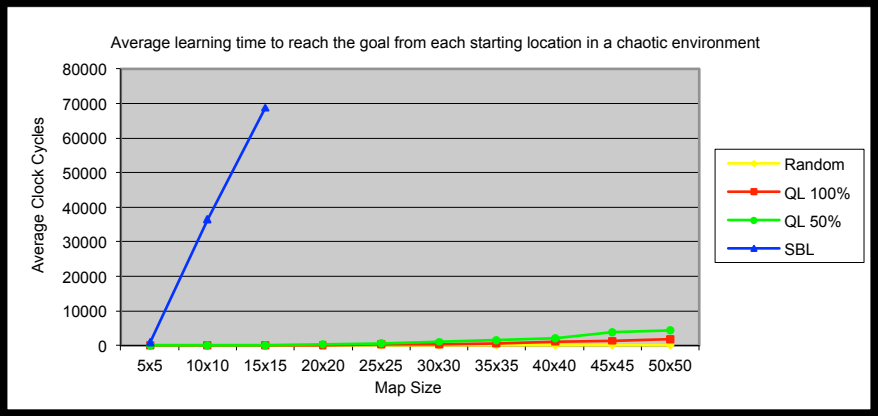
- Structured environment
  - Properties monitored by sensors demonstrate patterns associated with each action
  - Scales by exploiting structure
    - Number of prediction rules proportional to structure, not size of environment
    - In chaotic environments SBL memorizes the data





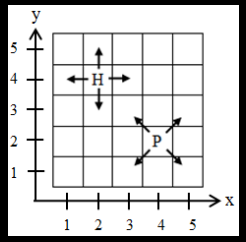


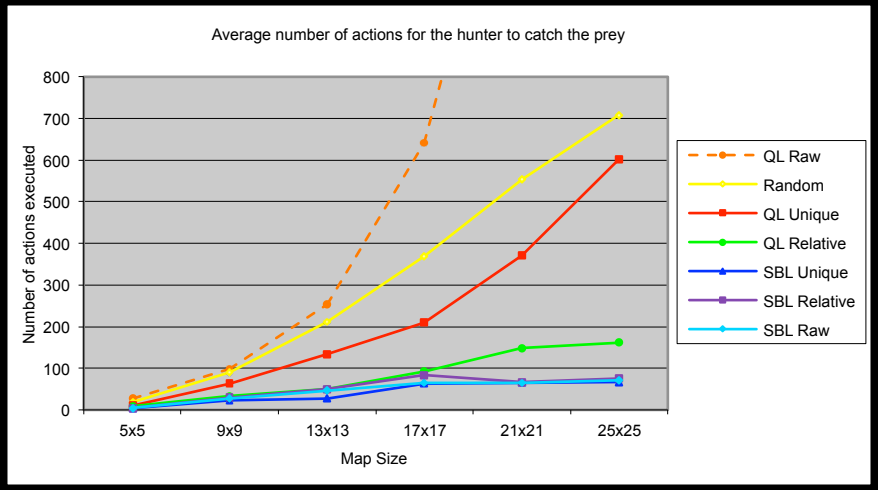




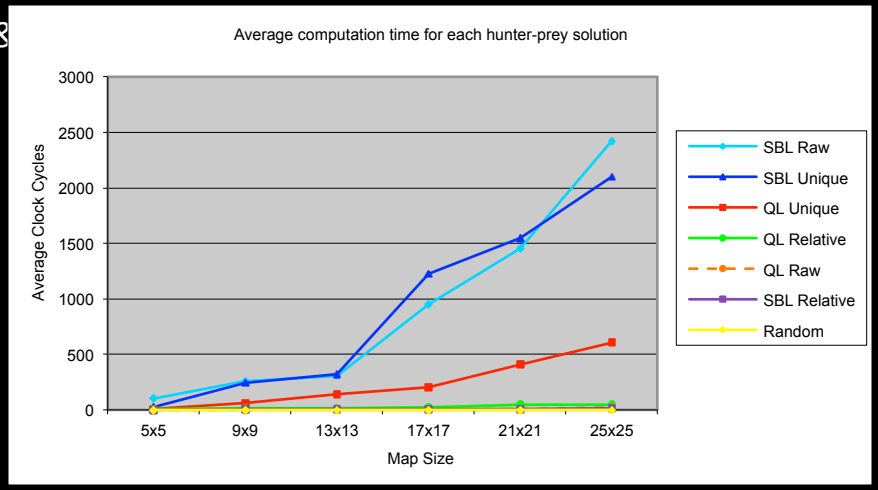
Learning from Uninterpreted Sensors & Actions

- Discretizes continuous data
  - Identifies data ranges and trends with comparison operators
    - e.g. Learned prediction rules forecast wrap around at boundaries
- Combines multiple uninterpreted sensors
  - Dimension reduction without hand-crafted approximation functions
    - Unique hcF: Locations of H & P merged into (map_width*map_height)^2 states
    - Relative hcF: Relative location of P from H (map_width*map_height) states



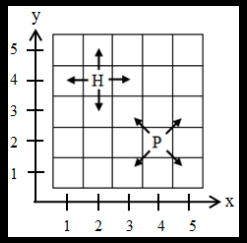


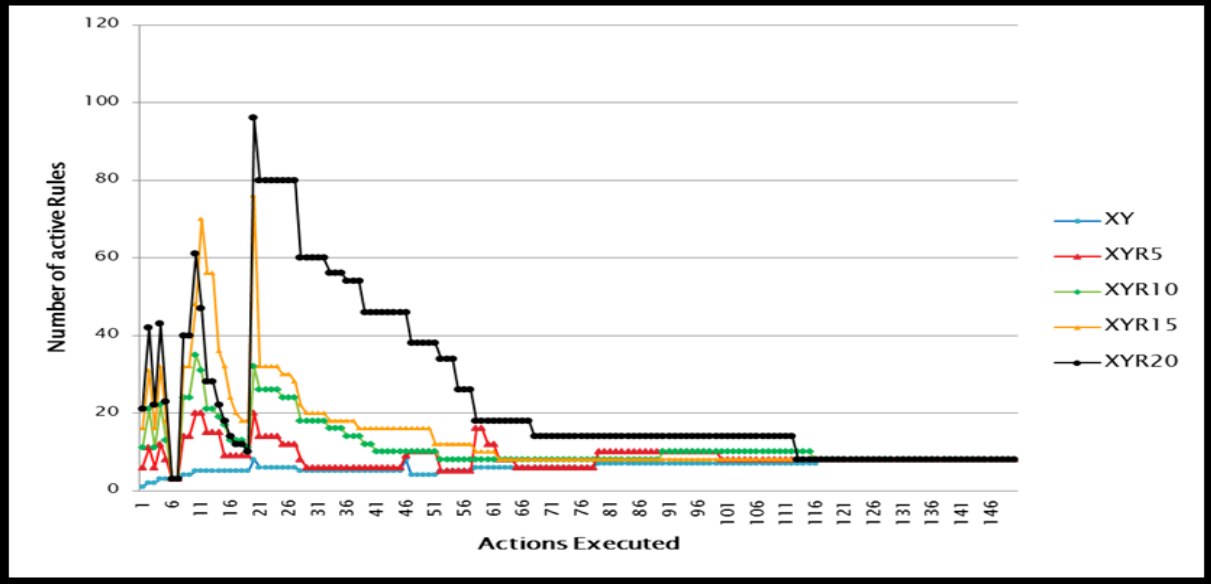
&



Scalability in Sensors

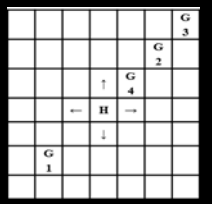
- Scales by exploiting structure, and sensor & action relevance
  - A different number of random number sensors was added to each test
  - The same sequence of 150 actions was executed in each test

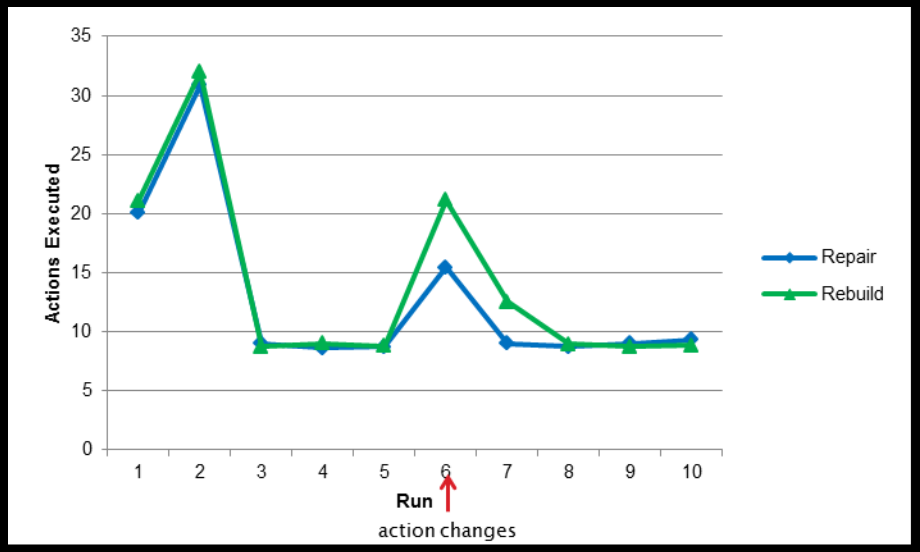


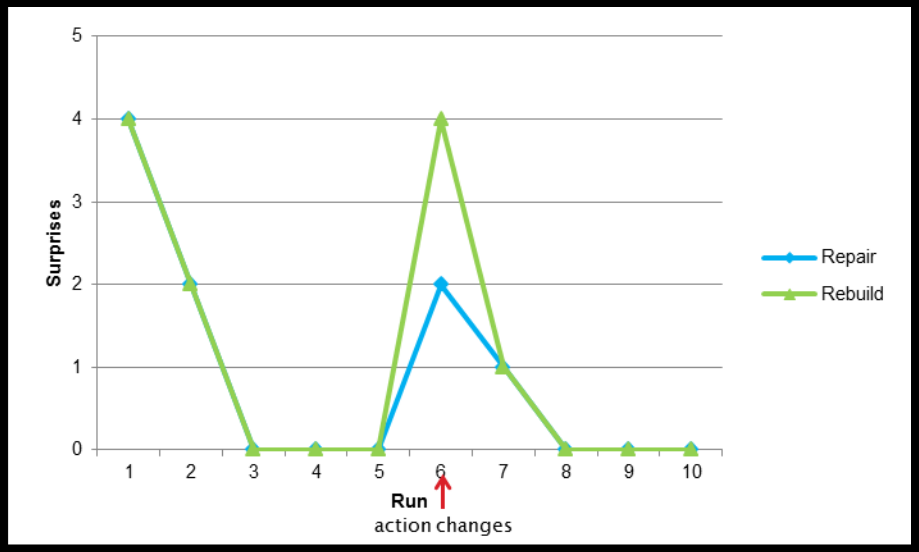


Repairing the Learned Model vs. Rebuilding from Scratch

- SBL detects unpredicted changes through surprises
  - Naturally repairs, does not rebuild from scratch
  - Forced to rebuild from scratch in this experiment
- Repairing preserves past experiences and prevents resource wastage
  - Run completed when goal reached, alternated between goals G_1 & G_2
  - In run 6, swapped N & E

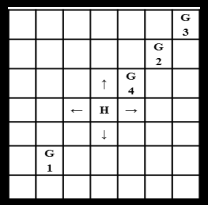


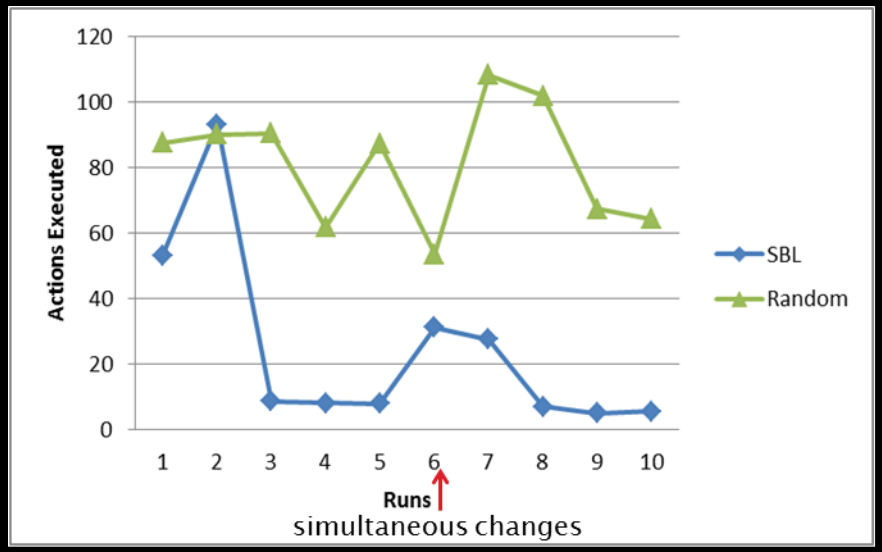


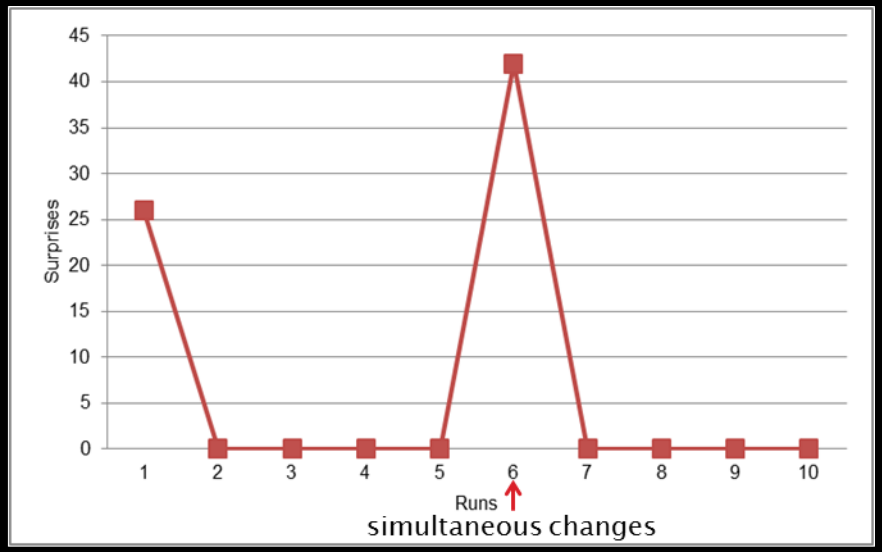


Detect and Adapt to Simultaneous Unpredicted Changes in Actions, Sensors & Goals

- SBL is open-minded for all possible causes, and eventually adapts
  - Run completed when goal reached, alternated between goals
  - In run 6, swapped N & W, removed u, moved G1 & G2 to G3 & G4

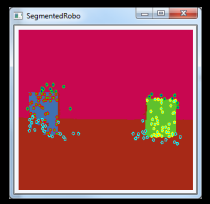


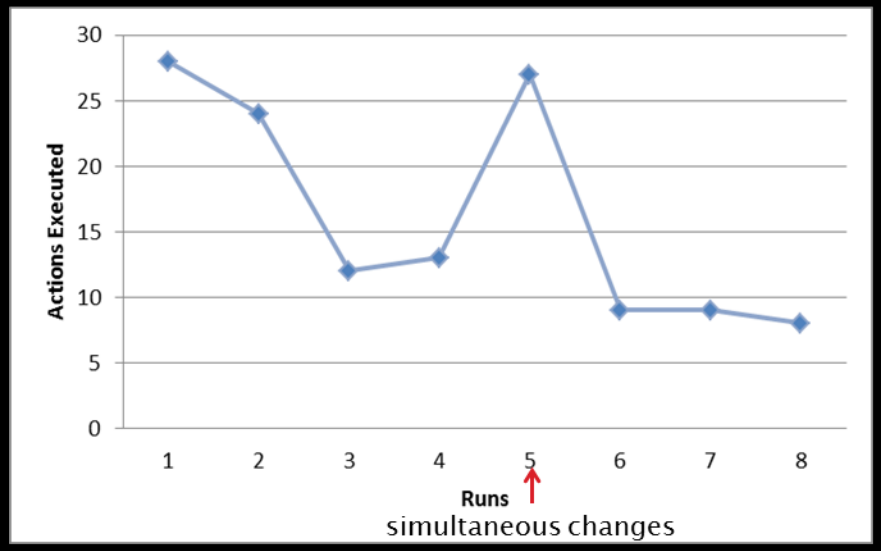


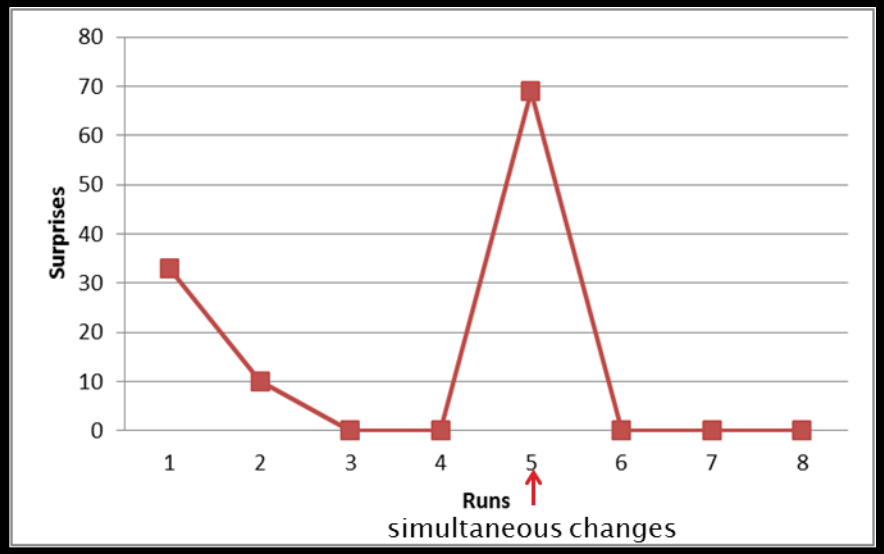


Detect and Adapt to Simultaneous Unpredicted Changes in Actions, Sensors, Goals & Environment

- SBL is feasible for real-world robots in unpredictable environments
  - Run completed when goal reached, alternated between goals
  - In run 5, swapped L & R, rotated 90° & translated camera, changed one goal book, moved the other goal book to a closer wall

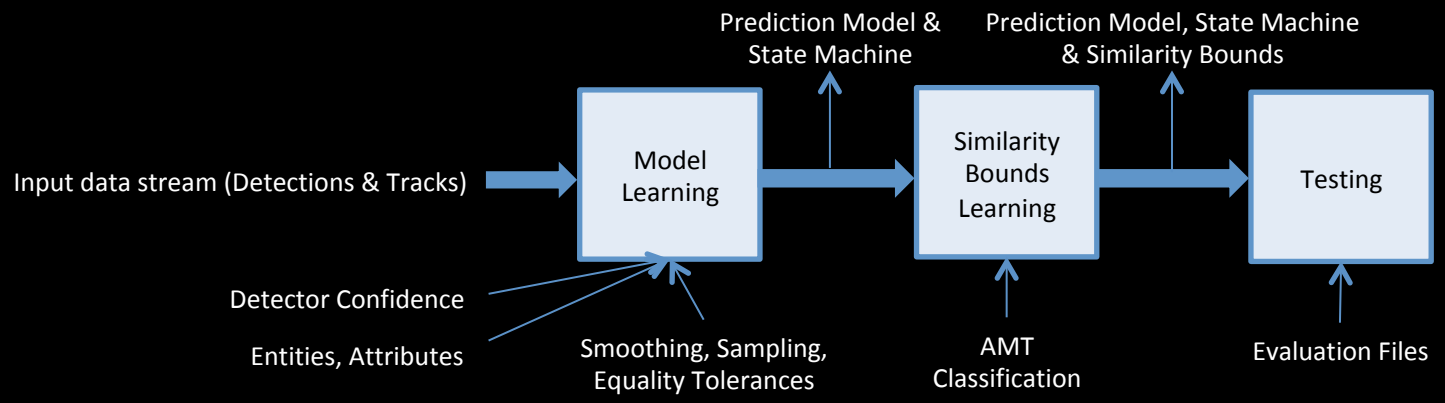






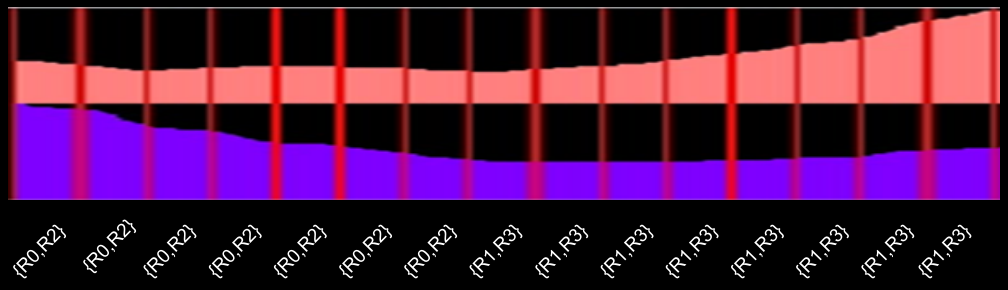
Detecting and Reasoning with Interference

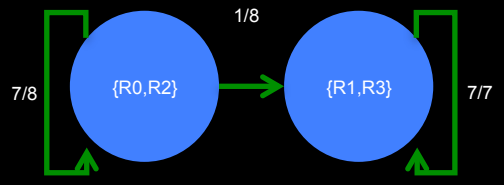
- Interference is short term disturbance to sensors or actions
  - Only “Noise” & “Missing data” (Gaps) were considered
  - Detected as surprises
  - Do not adapt the learned model as the cause is unidentifiable
- SBL applied to action recognition & gap filling in video data streams
  - approach, arrive, bounce, carry, catch, chase, collide, drop, exchange, exit, flee, go, haul, leave, lift, move, pass, pick up, push, put down, raise, replace, throw, walk
  - Entities: Actors, Objects, Relative Distance Composite
  - Attributes: x-location, y-location, velocity, size



Action Recognition – Learning I

- 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





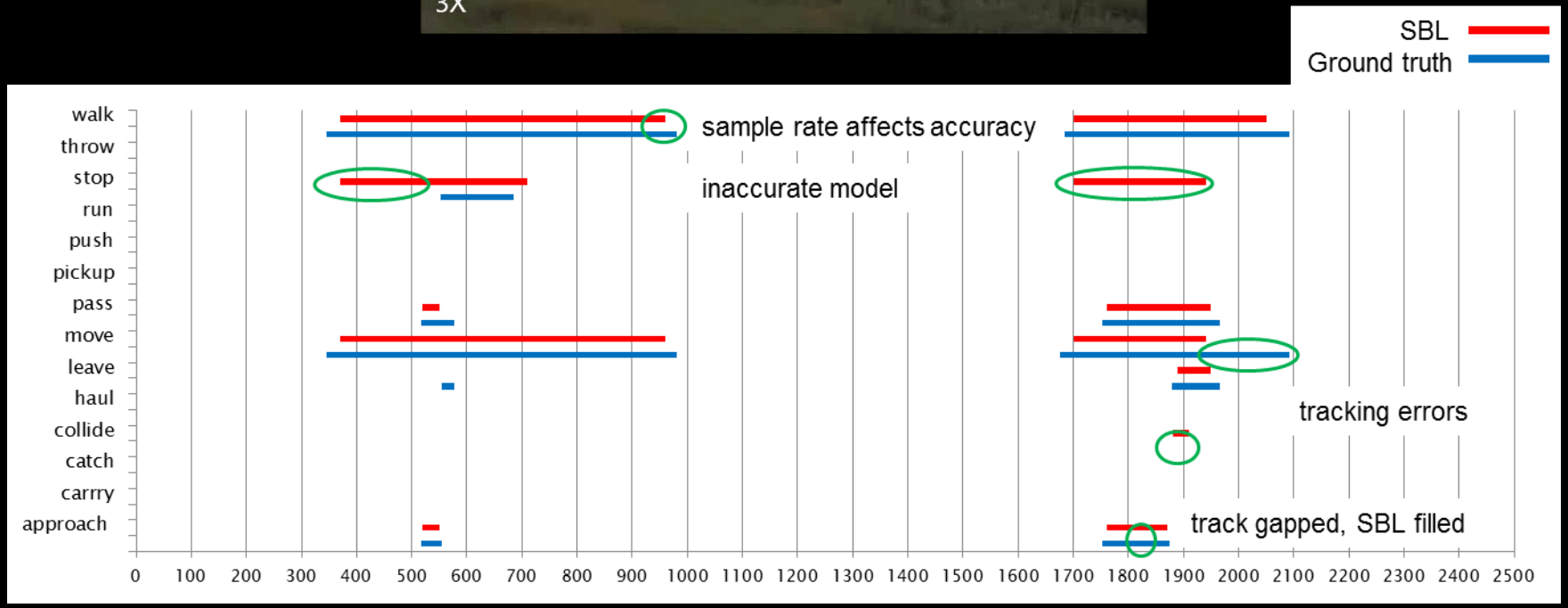
Action Recognition – Learning II

- Similarity Bounds Learning
  - Compute a prediction sequence by applying a model to the data stream
  - Calculate similarity metric to validate the observed state sequence
    - Valid state transitions score 0
    - Fixed negative penalties for invalid start state, end state and state transitions
    - similarity(sequence, MC) = 0 –noStartPenalty –noEndPenalty –n*transitionPenalty
    - Allows comparison of actions and videos with variable durations
  - Record lowest similarity value observed for all training examples
  - Highest similarity value = 0
  - Provides noise tolerance by accepting detections between these bounds

Action Recognition - Testing

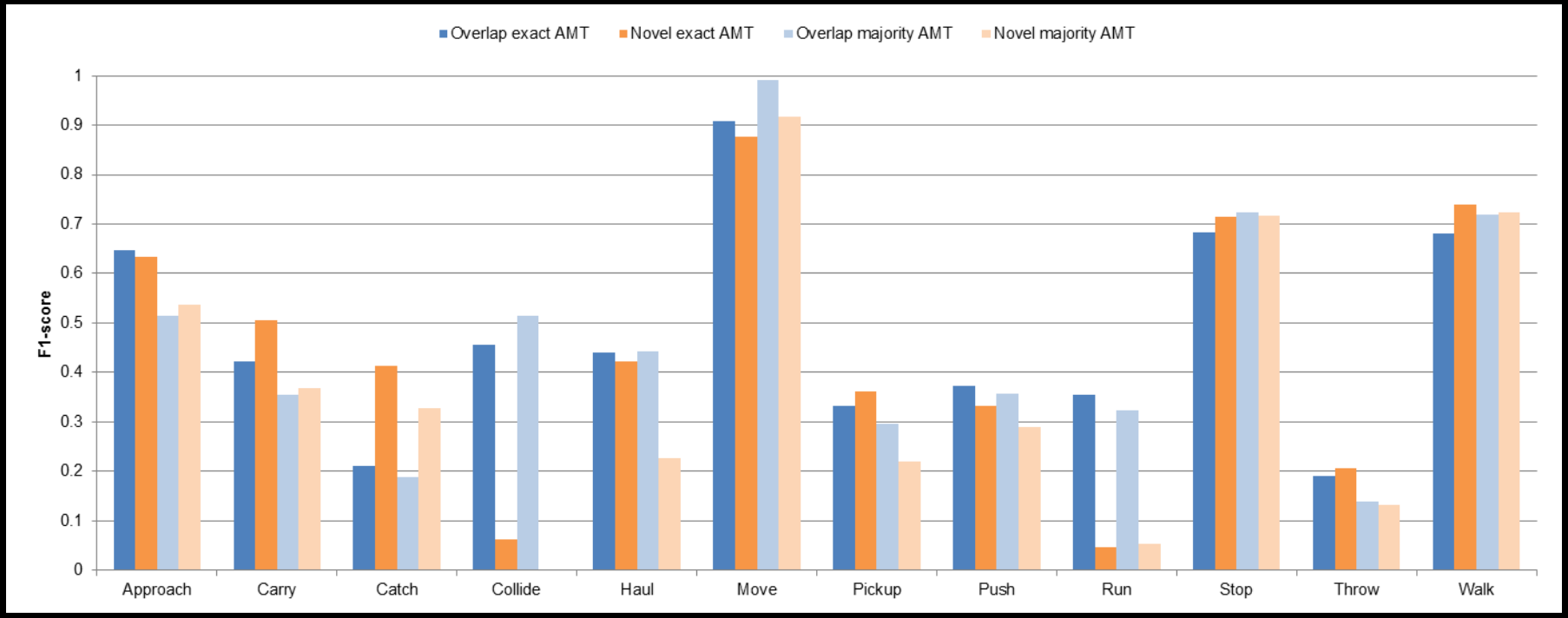
- Variable binding problem
  - Entities in a learned model may match several entities in the testing data
- For an unclassified video, given a model
  - Select a valid binding and extract a prediction sequence
  - Compute similarity metric for the prediction sequence
  - Test every valid binding and keep the highest value
  - If the highest value is within the learned similarity bounds the detection is true for that model
- When an action occurs multiple times in a video
  - Report matching start and end states as separate detections

Recognition Example



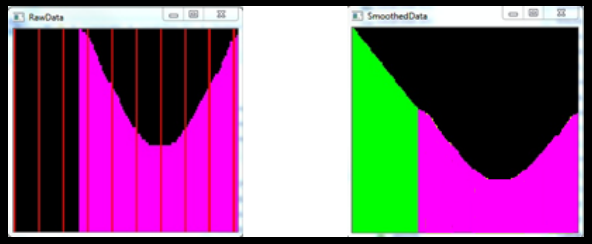
Recognition Results

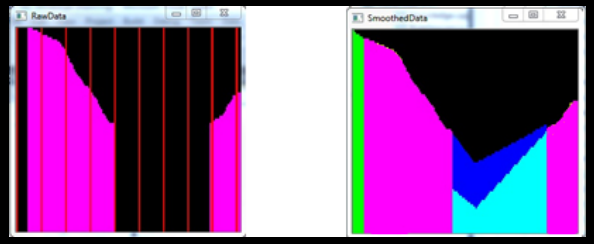
- Visint.org Year 1 Dataset, USC tracks, 1200 videos
- F1 = (2*precision*recall)/(precision+recall)
- Outperformed hand-coded SAM by more than 10%
- Reasoning with detector accuracies below 20%

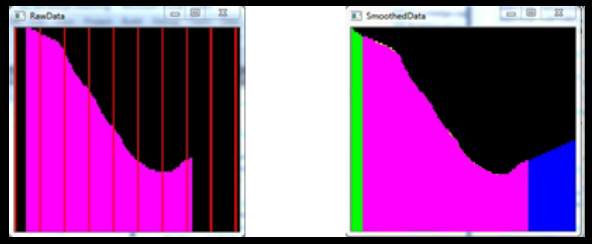


Missing Data - Gap Filling

- 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

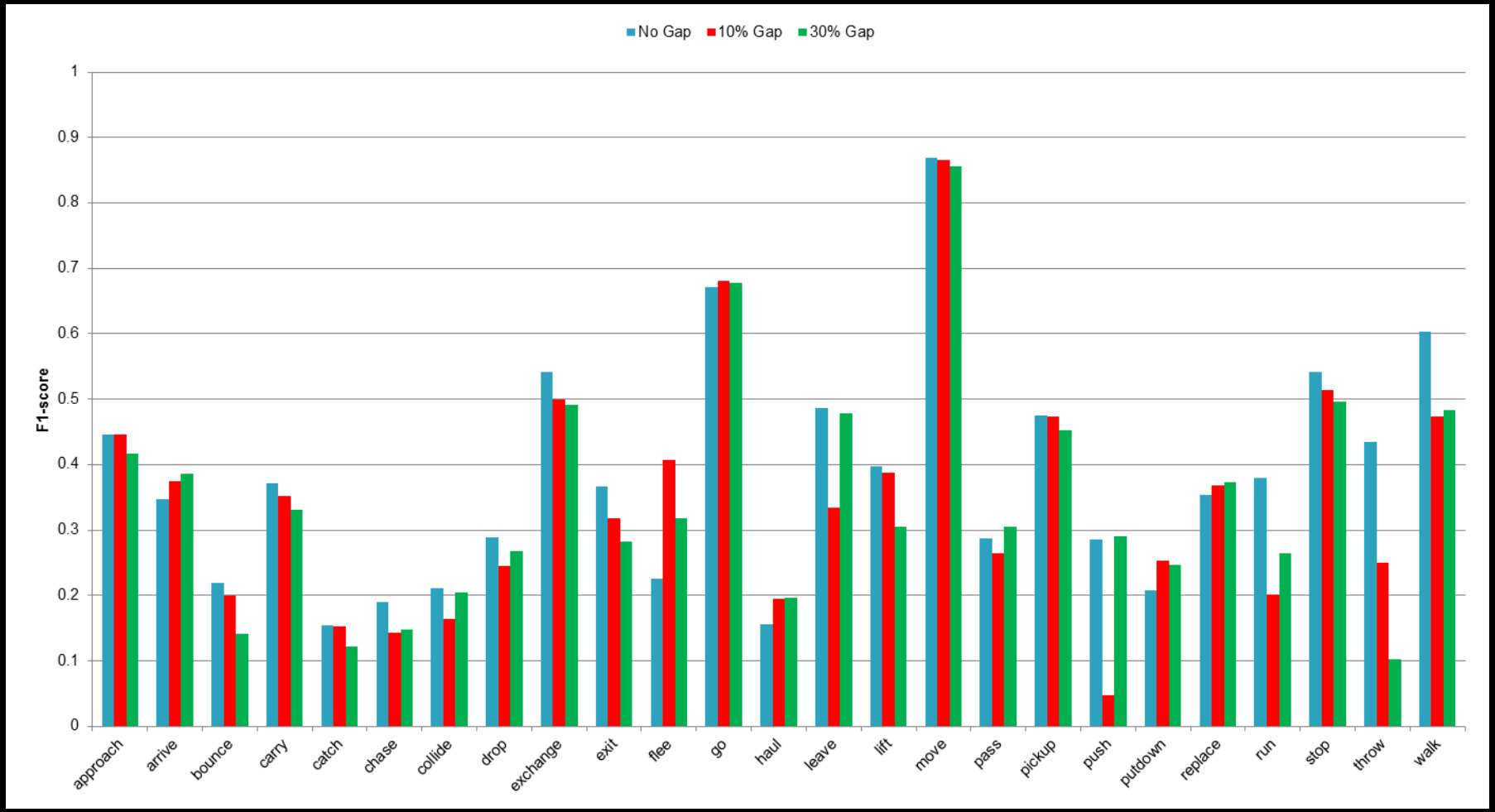






Gap Filling Results

- Synthetic gaps introduced into 700 annotated videos



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

Next Improvements

- 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 [Shen90]
    - 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

Conclusions

- SBL is a general learning method for structural learning from stream observations
- Useful for predictive learning, monitoring, and detecting abnormality
- Easy interface and comprehensible for human operators
- Has a great potental for the other projects

Stochastic Distinguishing Experiments

Thomas Joseph Collins and Wei-Min Shen

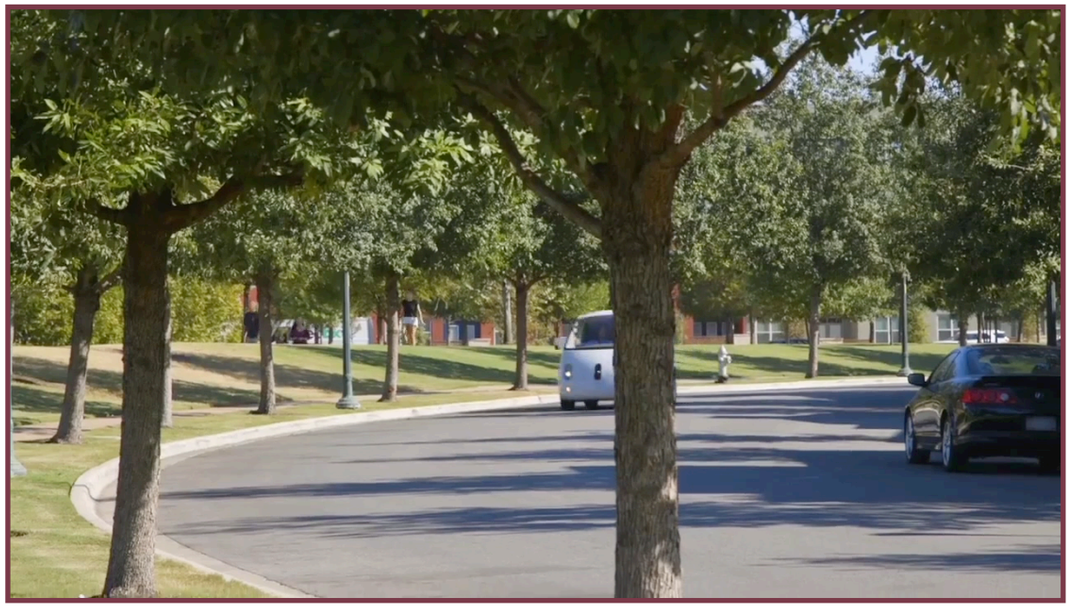
USC Viterbi
School of Engineering
Information Sciences

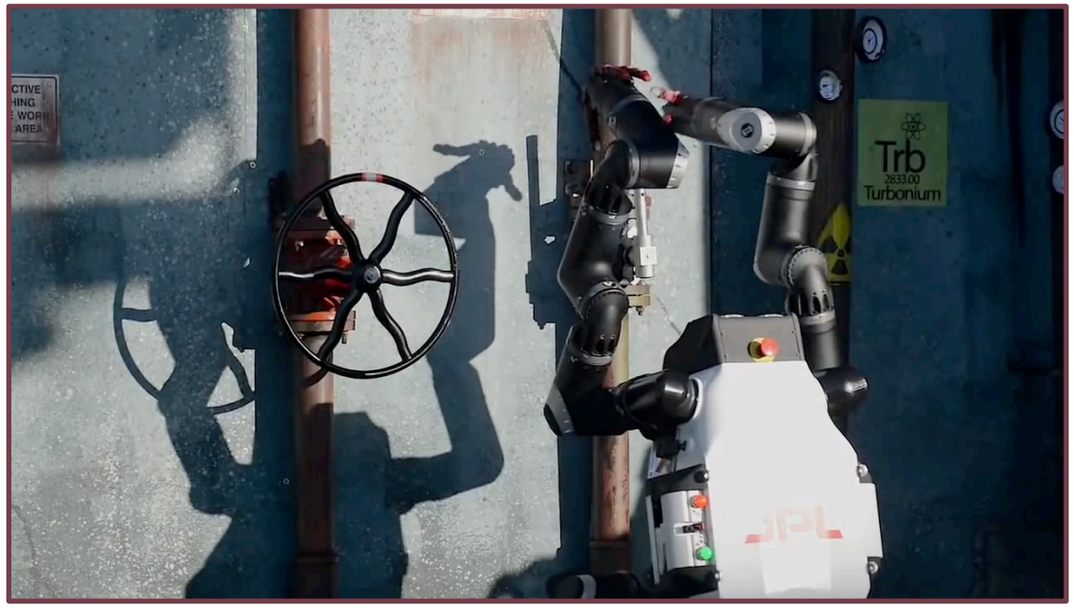
Overview

- Motivation
- Problem
- Challenges
- Related Work
- Stochastic Distinguishing Experiments
- Surprise-based POMDPs (sPOMDPs)
- Learning sPOMDPs
- Results
- Future Work

Motivation

- There is an ever-increasing need for autonomous robotic systems that are able to adapt to and operate in a range of challenging environments.

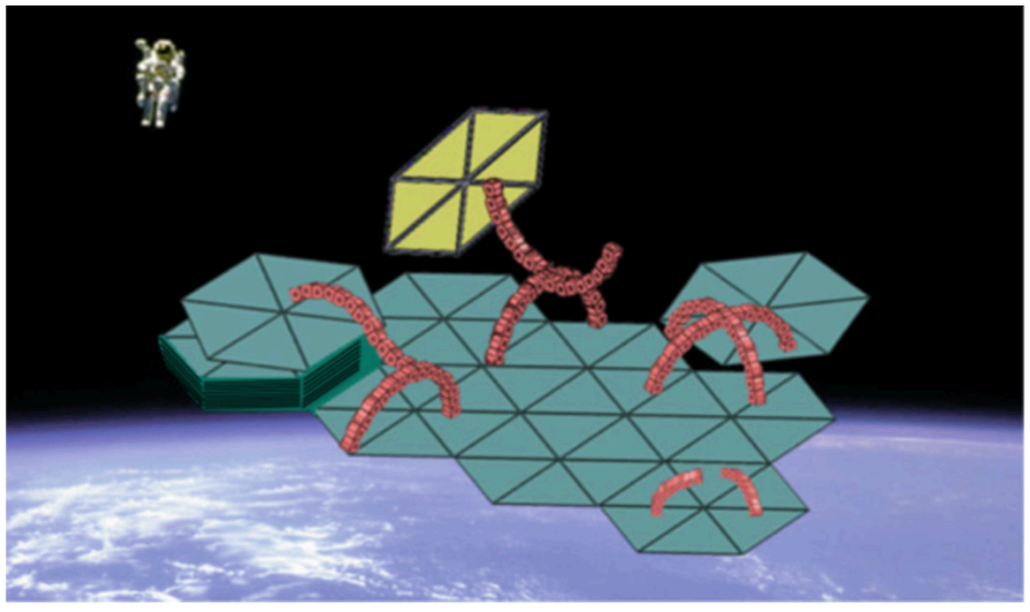


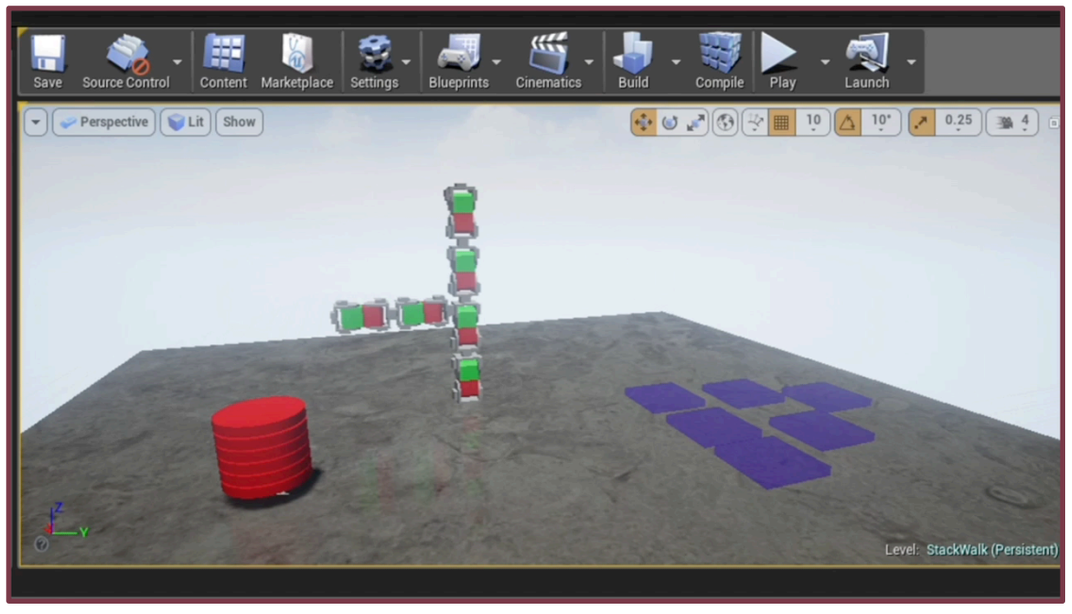


- These environments vary substantially depending on the task, but virtually all environments of interest exhibit high-levels of **partial-observability** and **stochasticity**.

Motivation

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





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

The Problem

Autonomous Learning from the Environment (**ALFE**, first formulated by Shen 1993) in which an autonomous agent is tasked with **actively** learning a **task-independent** model of the **state** and **dynamics** of an unknown discrete, partially-observable and stochastic environment from a stream of experience (**actions** and **observations**).

The Problem

Autonomous Learning from the Environment (**ALFE**, first formulated by Shen 1993) in which an autonomous agent is tasked with **actively** learning a **task-independent** model of the **state** and **dynamics** of an unknown discrete, partially-observable and stochastic environment from a stream of experience (**actions** and **observations**).

- This problem is highly unique in requiring that agent simultaneously decide:
  - **What actions to take** in the absence of external rewards.
  - **How much experience is needed** to build a useful environment model.

The Problem

- Solving ALFE is an important theoretical step toward **lifelong model learning and decision making** in unknown partially-observable and stochastic environments.
- By **lifelong**, we mean a learning procedure that is meant to be invoked multiple times (or even continuously).

Key Challenges

- Partial-observability
  - **Must consider (potentially infinite) history.**
- Stochasticity
  - **Noise vs. surprise.**
- Minimal prior knowledge
  - **Learned model must have nonparametric form that grows/shrinks with data.**
- Action selection
  - **How to choose actions with no reward/goal?**
- State space formalization
  - **Formalize the learned model to be used in place of human-designed state spaces.**

Related Work

- Latent state (hidden variable) approaches:
  - **Representation/deep learning** (Bengio et al. 2013, Goodfellow et al. 2016)
  - **Probabilistic graphical models/temporal models** (Koller and Friedman, 2009)
    - **HMMs, DBNs** (Koller & Friedman 2009)
  - **Reinforcement learning** (Murphy 2000, Pineau et al. 2003, Mnih et al. 2015)
    - **Intrinsically-motivated** (Chentanez et al. 2005)
    - **Curiosity-based** (Pathak et al. 2017)
- Predictive (action/observation) approaches:
  - **Predictive state representations** (Littman & Sutton 2002, Boots et al. 2010)
  - **Stochastic distinguishing experiments** (Collins & Shen, 2017)
- **Stochastic Distinguishing Experiments** and **Surprise-based POMDPs** are a novel hybrid latent-predictive approach in which the results of predictive experiments are used to uniquely identify latent states emitting the same observation.

Stochastic Distinguishing Experiments (SDEs) [BICA 2018]

- **Stochastic Distinguishing Experiments** (Collins & Shen 2017) are sequences of actions – i.e., **experiments** the agent can perform – that statistically disambiguate identical-looking states.
- Let M be a set of latent (environment or model) states. We say that $s = (a_1, \ldots, a_k)$ is a **Stochastic Distinguishing Experiment (SDE)** for m^i and m^j if the most likely sequences of observations generated by executing s from m^i and m^j differ ($g^*_{m^i} \neq g^*_{m^j}$).
- The higher the probability of observing $g^*_{m^i}$ from state m^i and $g^*_{m^j}$ from state m^j, the more useful this SDE is in distinguishing m^i and m^j because $g^*_{m^i}$ must have low probability when s is executed from m^j and vice versa.

Surprise-based partially-observable Markov decision processes (sPOMDPs)

- A **Surprise-based POMDP (sPOMDP)** is a POMDP augmented with a set of SDEs S. Each $s \in S$ statistically disambiguates at least two latent model states $m^i \neq m^j \in M$. Each $m \in M$ is represented by the unique most likely sequence of observations resulting from executing the associated SDE from that model state, g^*_m.
- Named for the **surprise-based** algorithm for actively learning sPOMDPs in unknown environments from agent experience.

Stochastic Distinguishing Experiments: Predictive Definition [BICA 2017]

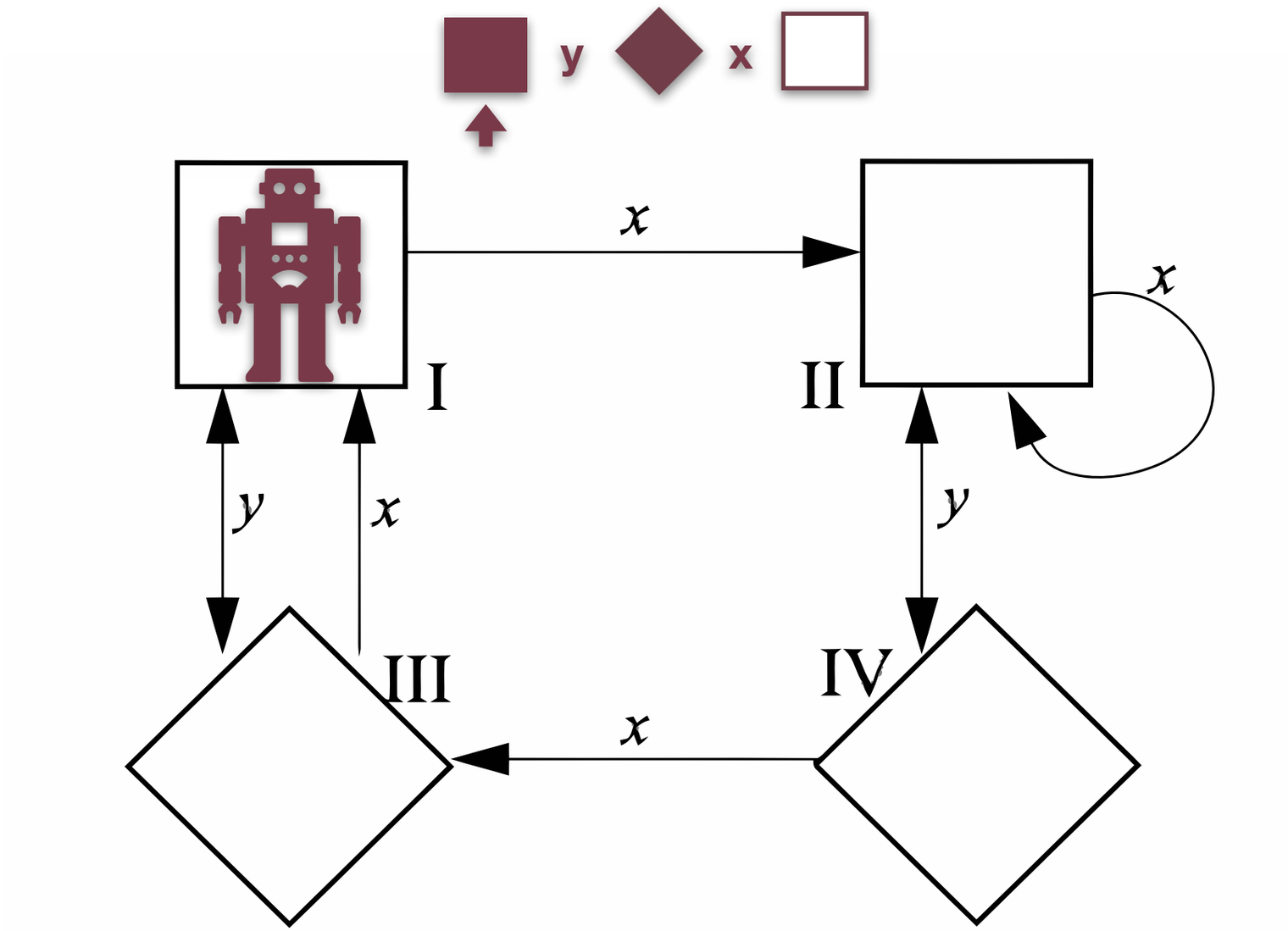
- SDEs were originally defined as **time-invariant conditional probability distributions** over the next observation given a finite, variable-length sequence of ordered actions and expected observations up to the present.
- The **predictive SDE learning algorithm** incrementally extends these experiments — beginning with a single SDE whose experiment is null — while ensuring the **conditions (experiments)** of SDEs are **mutually exclusive and exhaustive at each step**, with each possible agent history matching the experiment of **exactly one SDE**.

$$P(O_{t+1} \mid o_{t-k+1}, a_{t-k+1}, \ldots, o_t, a_t)$$

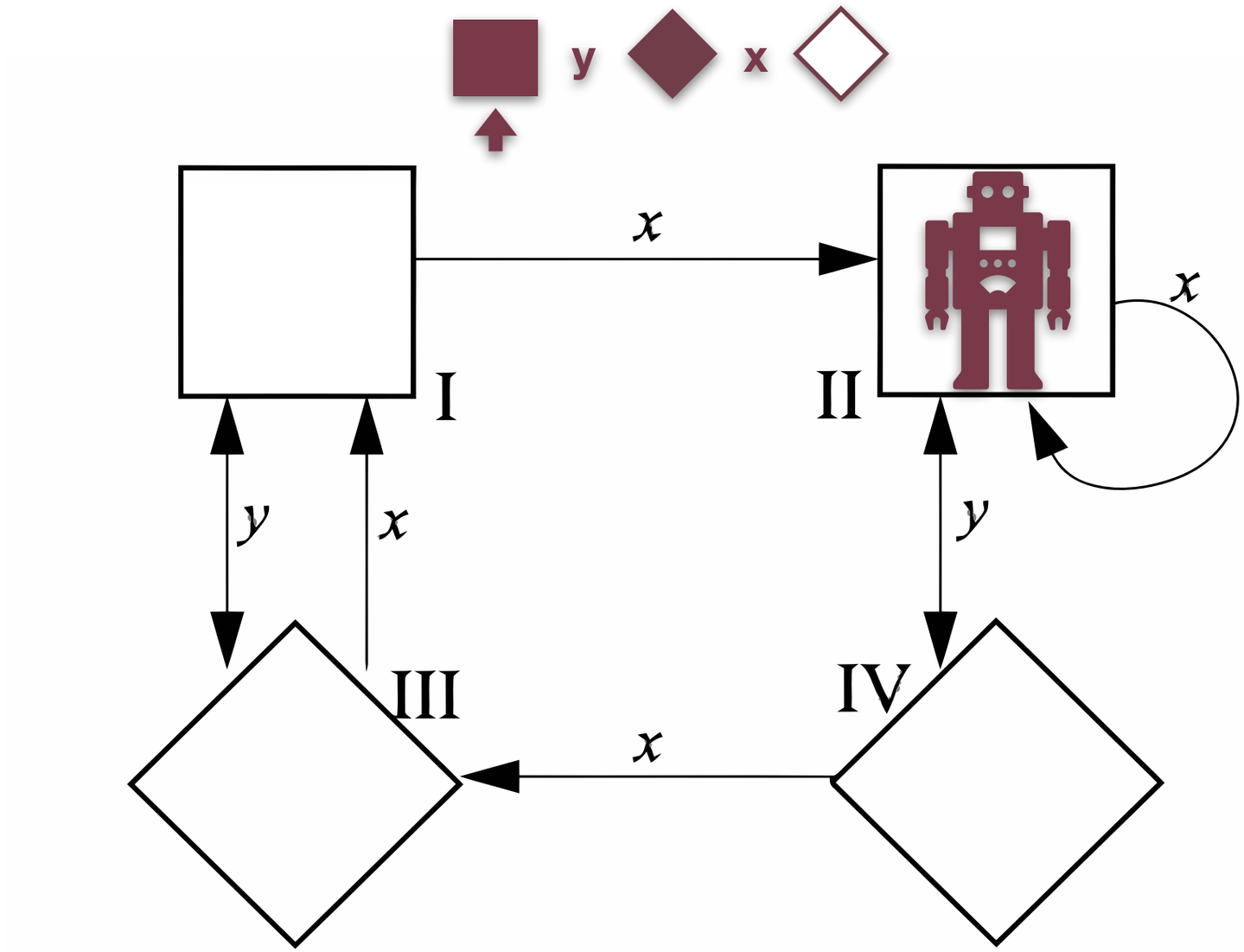
Random variable representing the next observation

Experiment

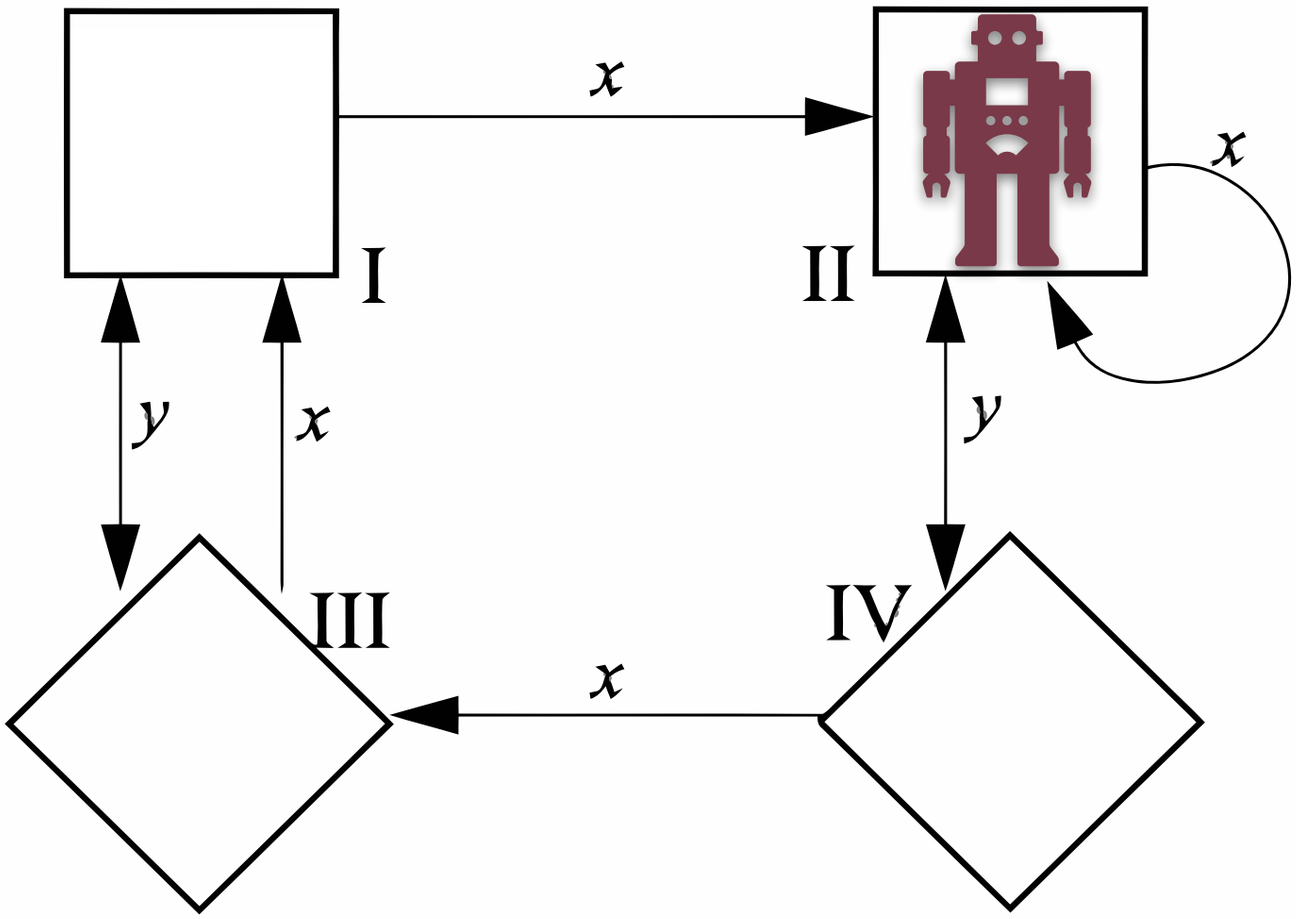
Surprise-based partially-observable Markov decision processes (sPOMDPs)

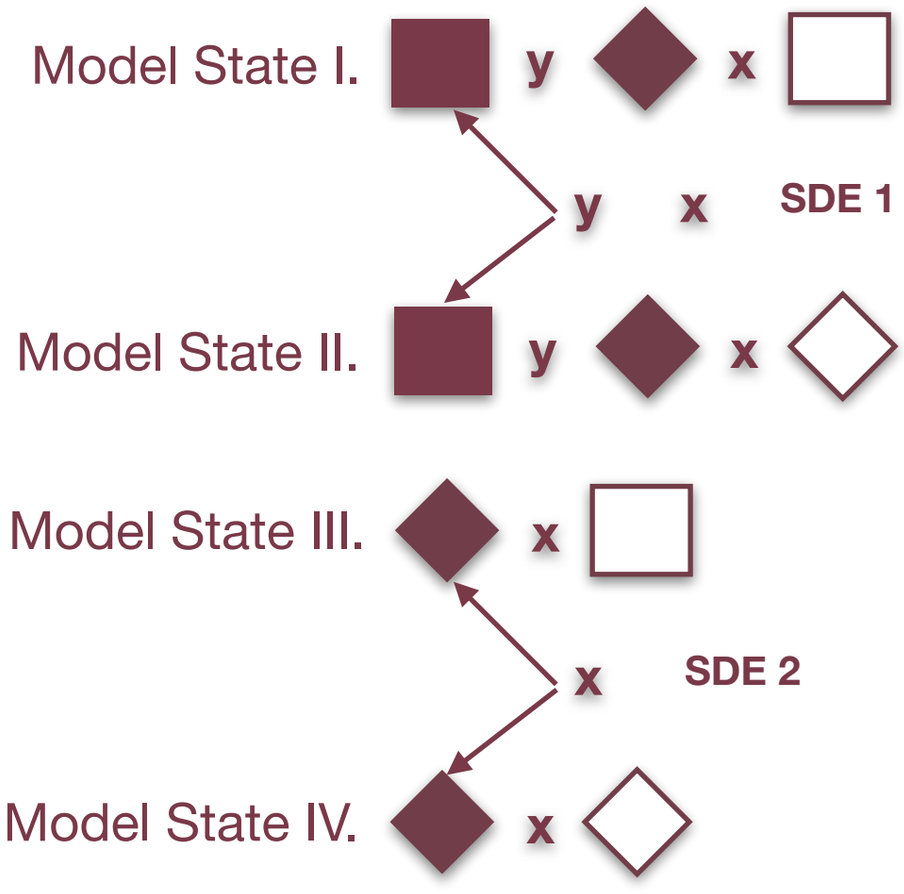


Surprise-based partially-observable Markov decision processes (sPOMDPs)

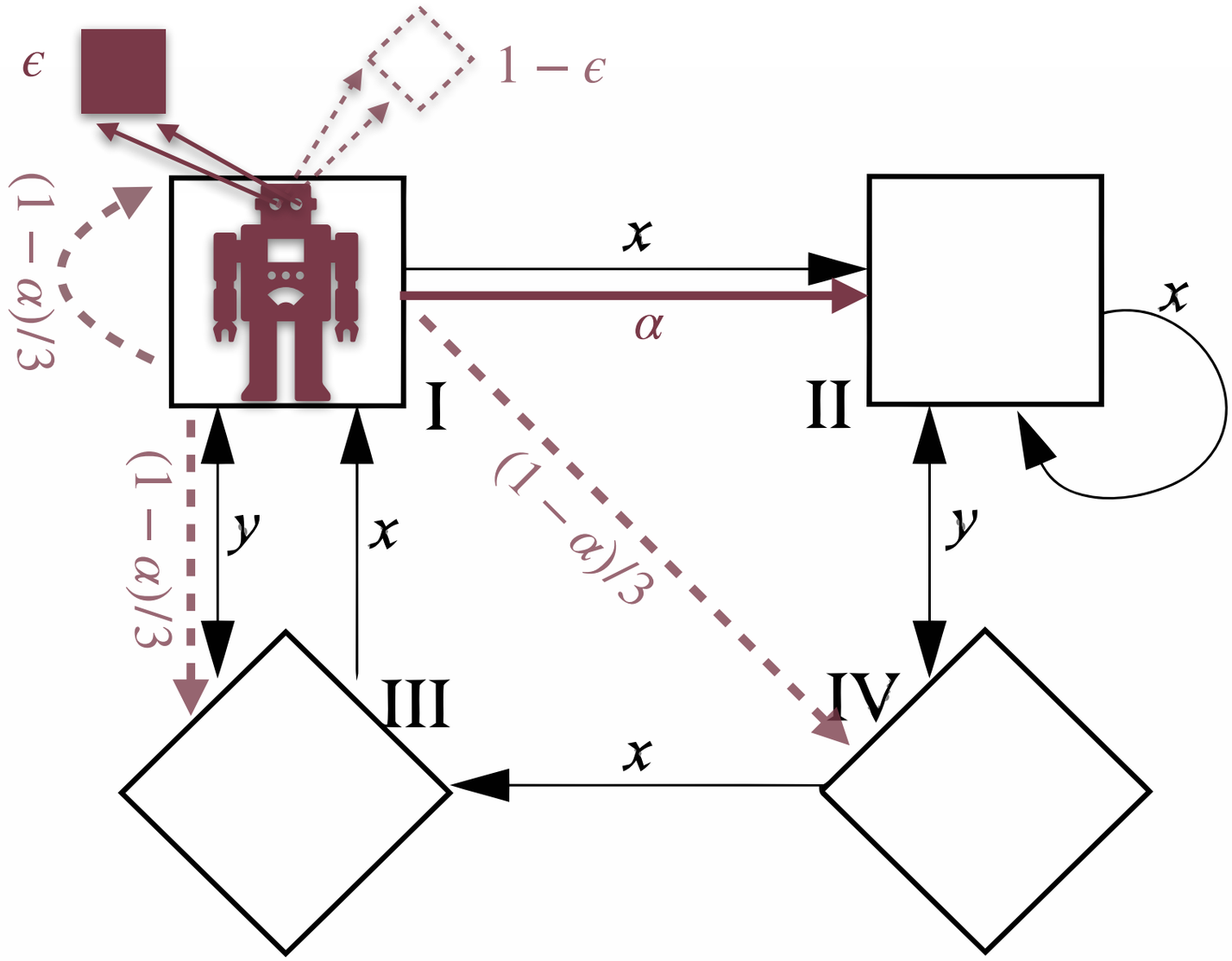


Surprise-based partially-observable Markov decision processes (sPOMDPs)



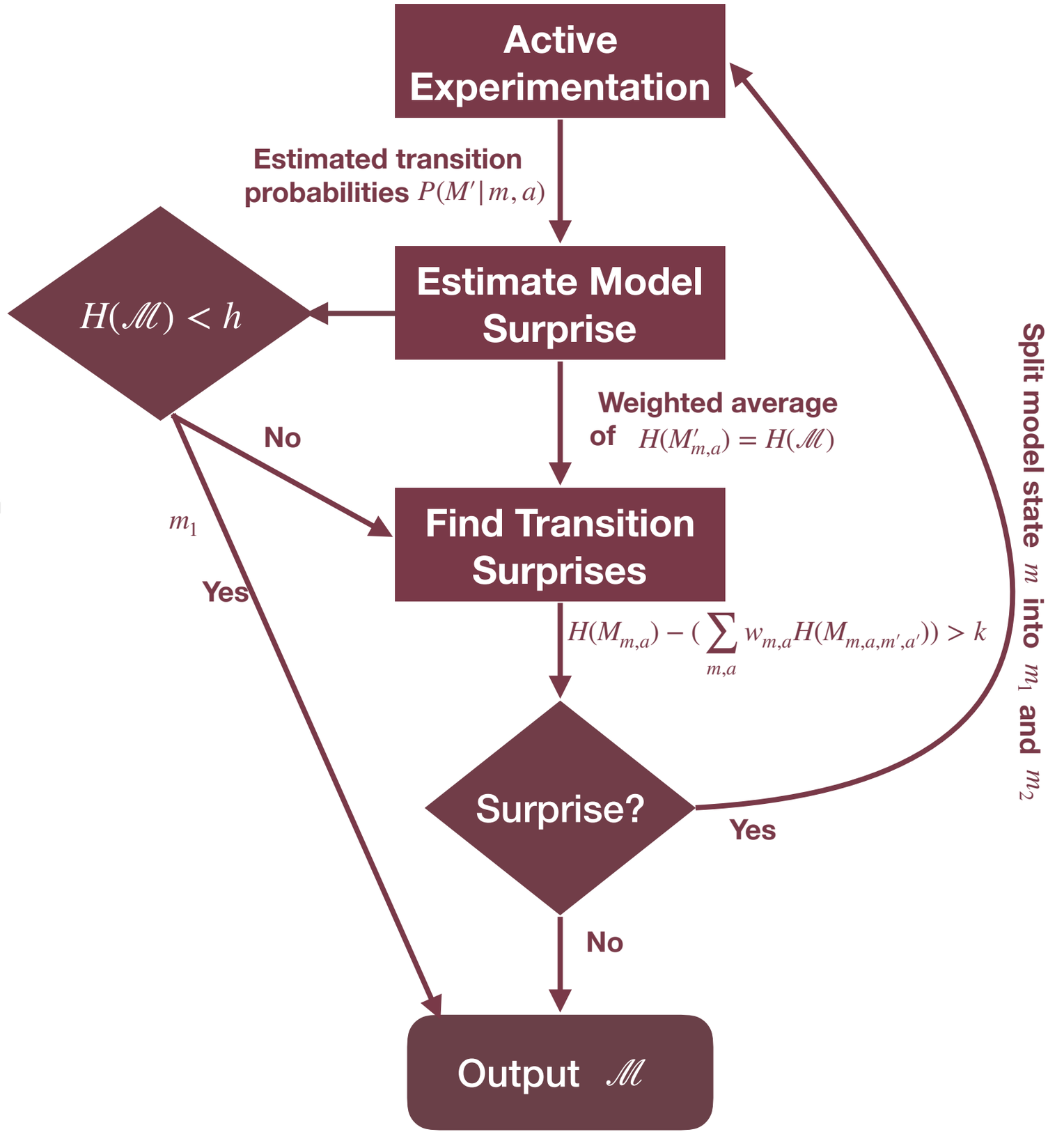


Alpha-epsilon POMDP environments

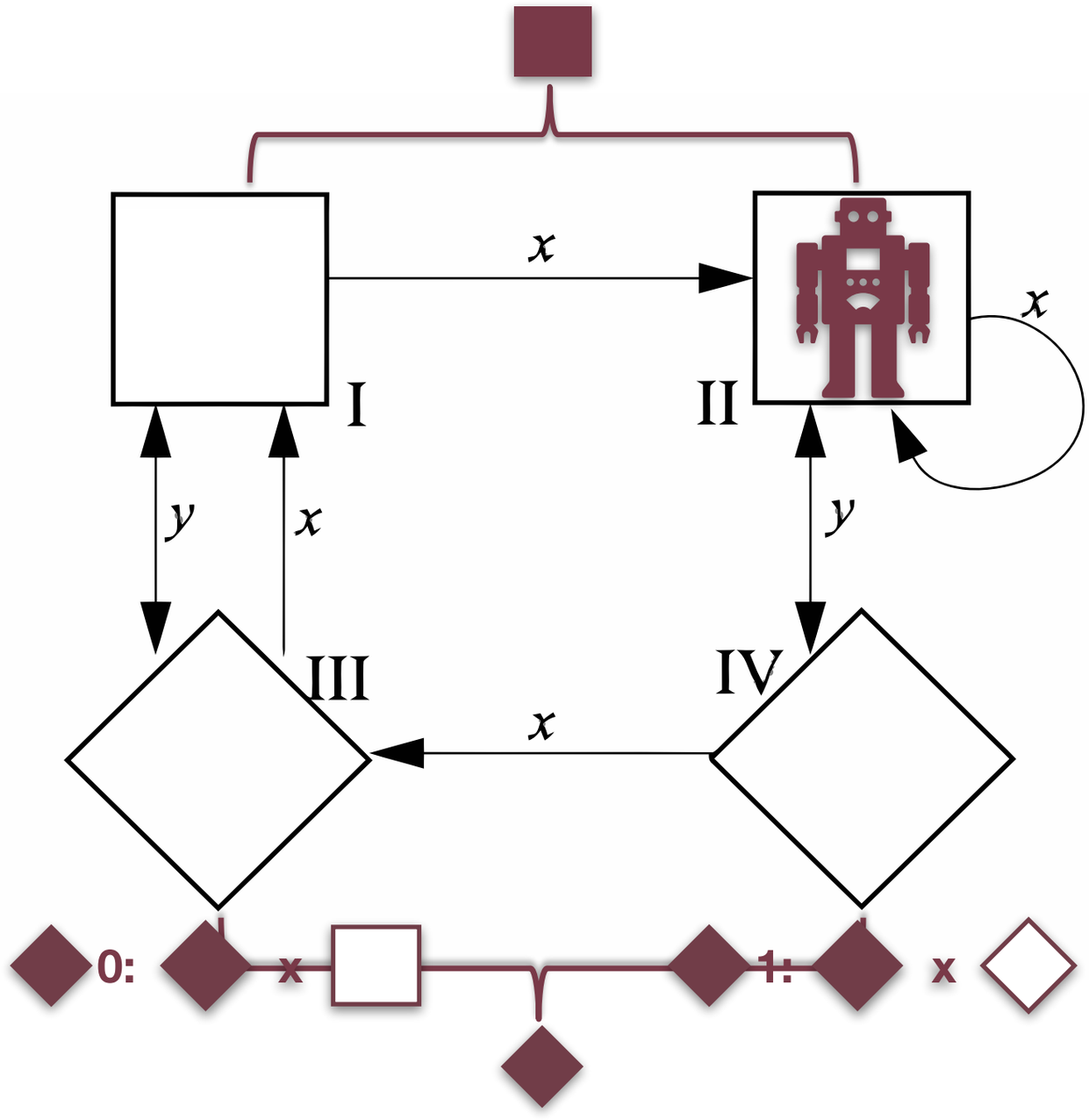


Learning sPOMDPs

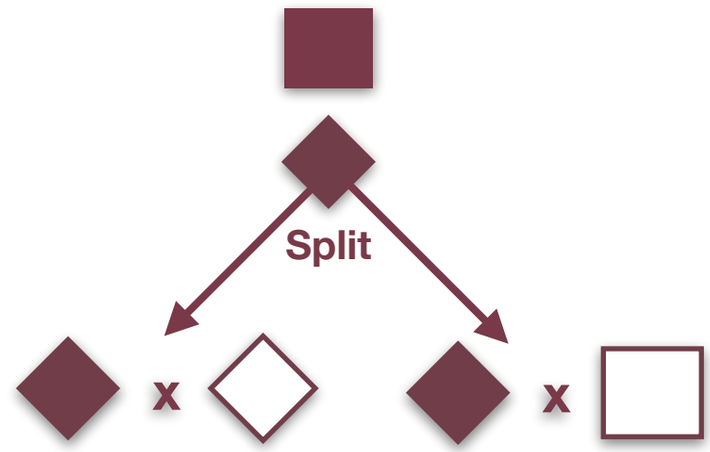
- Build a model whose **surprise value** (**weighted sum of transition distribution entropies**) is as low as possible.
- Begin by assuming observations are a suitable state space and look for experimental evidence that is **surprising** under this assumption.
- Large **reductions in transition entropy** resulting from considering an additional model state-action pair of history indicate a **violation of the Markov property**, and we **split** model state **m** into two model states:
  - **Intuition**: Model state **m** is representing underlying environment states with **differing dynamics**.
- By executing the **actions of SDEs** as well as random actions (with some probability), the agent can remain **localized in its model** while simultaneously **learning model state transition probabilities.**



Learning sPOMDPs



Current Model States:



Transition Probabilities:

$P(\{\blacksquare, \blacklozenge\} | \blacksquare, x) = \{0.99, 0.01\}$

$P(\{\blacksquare, \blacklozenge\} | \blacksquare, y) = \{0.01, 0.99\}$

$P(\{\blacksquare, \blacklozenge\} | \blacklozenge, x) = \{0.51, 0.49\}$

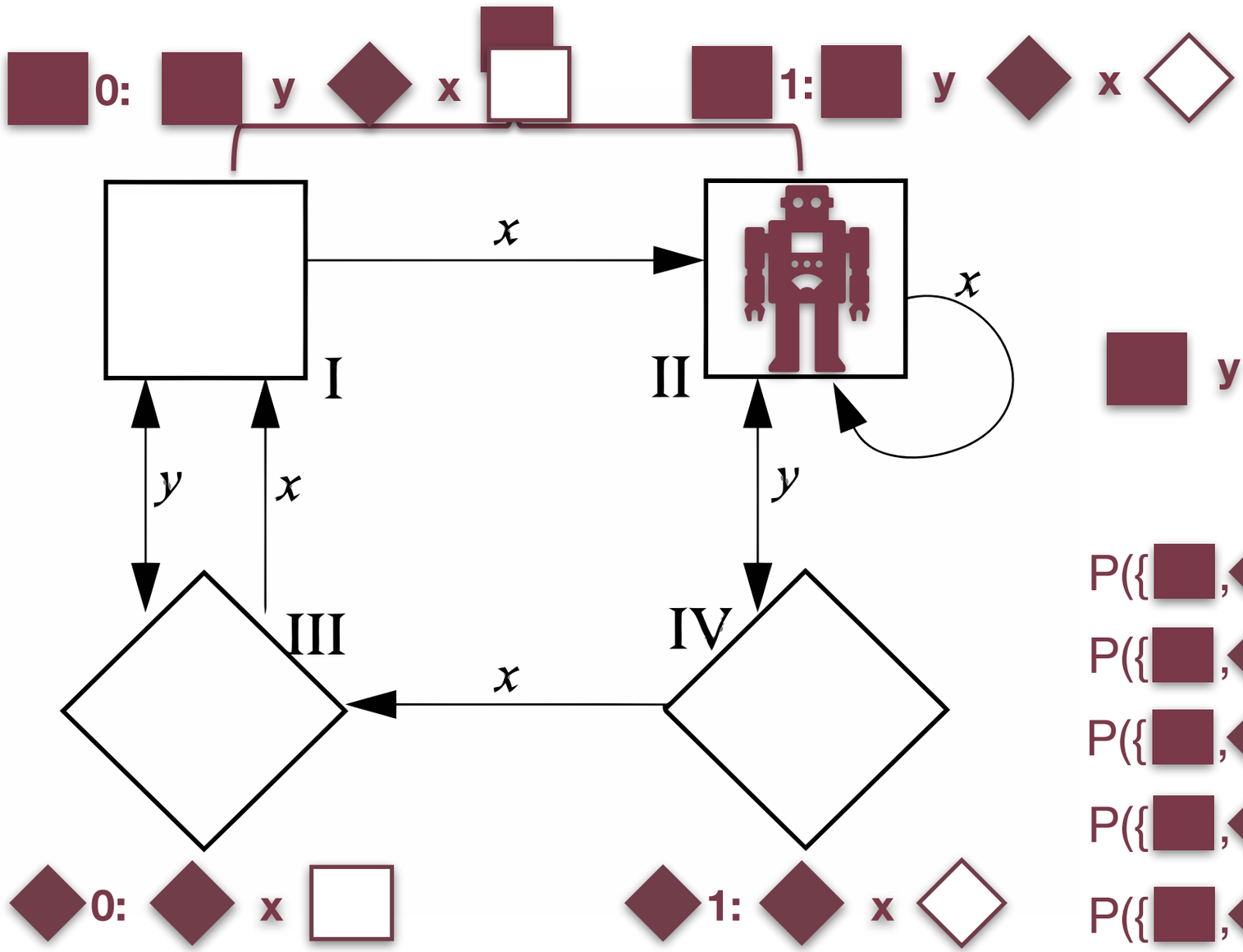
$P(\{\blacksquare, \blacklozenge\} | \blacklozenge, y) = \{0.99, 0.01\}$

Surprise!

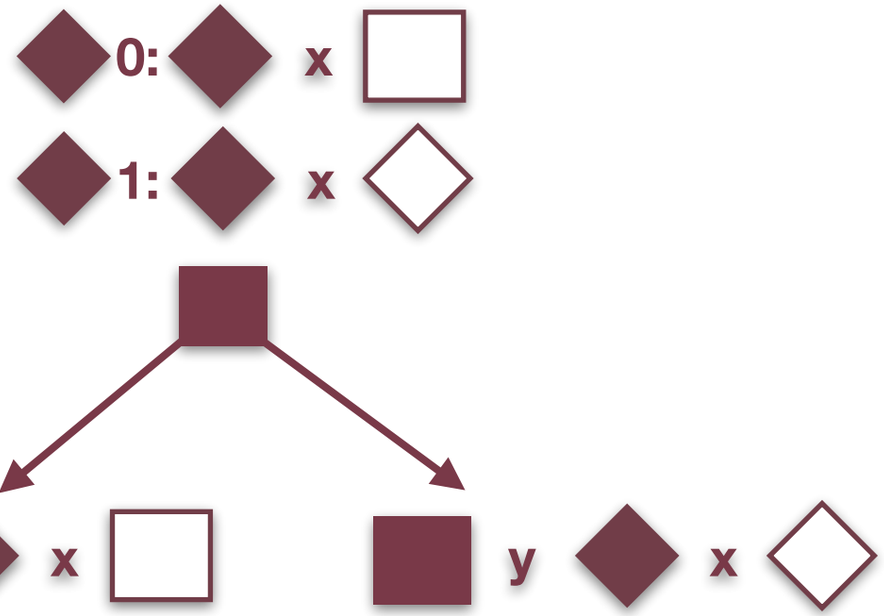
$$H(M_{m,a}) - \left(\sum_{m,a} w_{m,a} H(M_{m,a,m',a'})\right) > k$$

USC Viterbi School of Engineering
Information Sciences

Learning sPOMDPs



Current Model States:



Transition Probabilities:

$P(\{\blacksquare, \blacklozenge\,\mathbf{0}, \blacklozenge\,\mathbf{1}\} \mid \blacksquare, x) = \{0.98, 0.01, 0.01\}$

$P(\{\blacksquare, \blacklozenge\,\mathbf{0}, \blacklozenge\,\mathbf{1}\} \mid \blacksquare, y) = \{0.01, 0.33, 0.66\}$

$P(\{\blacksquare, \blacklozenge\,\mathbf{0}, \blacklozenge\,\mathbf{1}\} \mid \blacklozenge\,\mathbf{0}, x) = \{0.98, 0.01, 0.01\}$

$P(\{\blacksquare, \blacklozenge\,\mathbf{0}, \blacklozenge\,\mathbf{1}\} \mid \blacklozenge\,\mathbf{0}, y) = \{0.98, 0.01, 0.01\}$

$P(\{\blacksquare, \blacklozenge\,\mathbf{0}, \blacklozenge\,\mathbf{1}\} \mid \blacklozenge\,\mathbf{1}, x) = \{0.01, 0.98, 0.01\}$

$P(\{\blacksquare, \blacklozenge\,\mathbf{0}, \blacklozenge\,\mathbf{1}\} \mid \blacklozenge\,\mathbf{1}, y) = \{0.98, 0.01, 0.01\}$

Surprise!

$$H(M_{m,a}) - \left(\sum_{m,a} w_{m,a} H(M_{m,a,m',a'})\right) > k$$

USC Viterbi
School of Engineering
Information Sciences

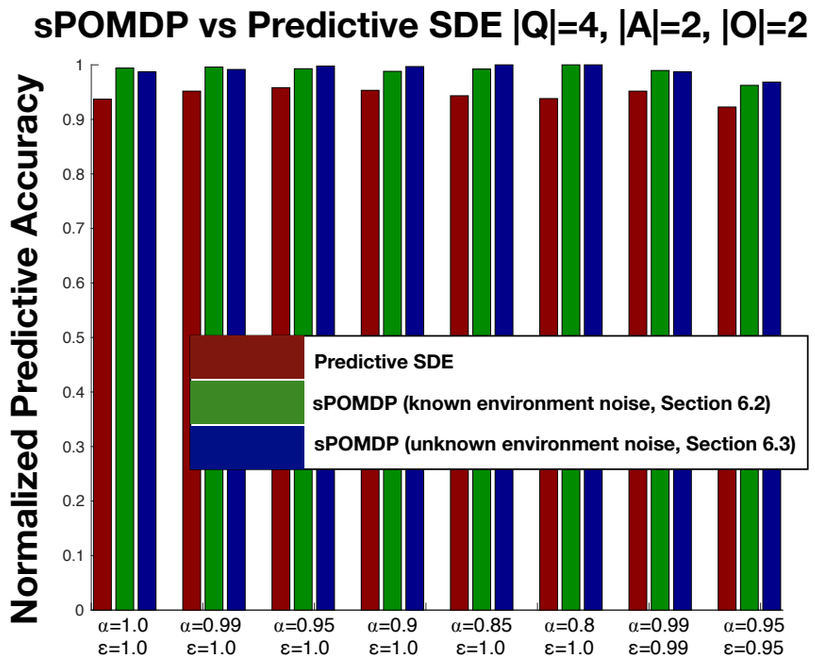
Theoretical Results

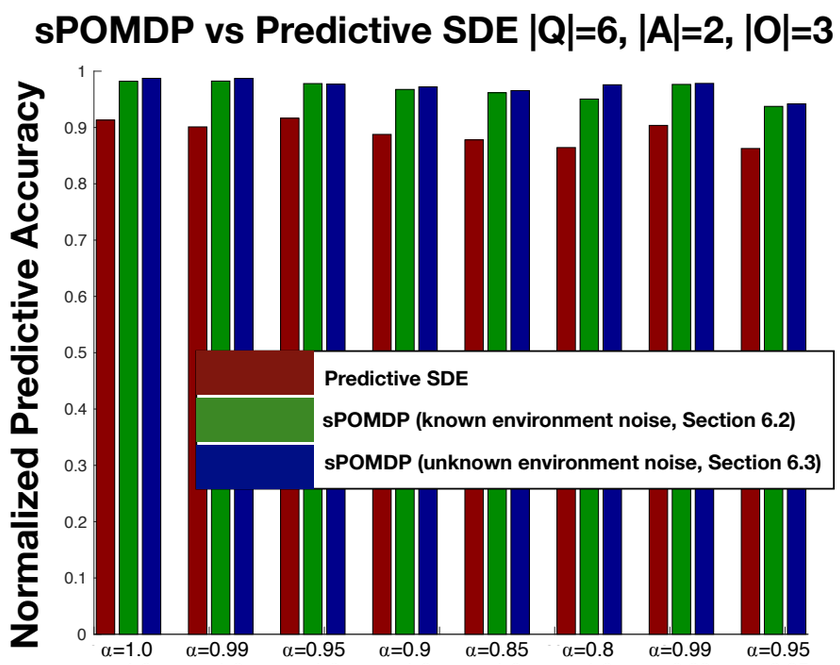
Theorem 1. *For any Moore machine environment $\mathcal{E}$, there exists a* perfect *sPOMDP model $\mathcal{M}$ where $|M| \leq |Q|$, $|S| \leq |Q| - 1$, and each SDE $s \in S$ has, at most, $|Q| - 1$ actions, where $|Q|$ is the number of environment states in the* minimal *representation of $\mathcal{E}$.*

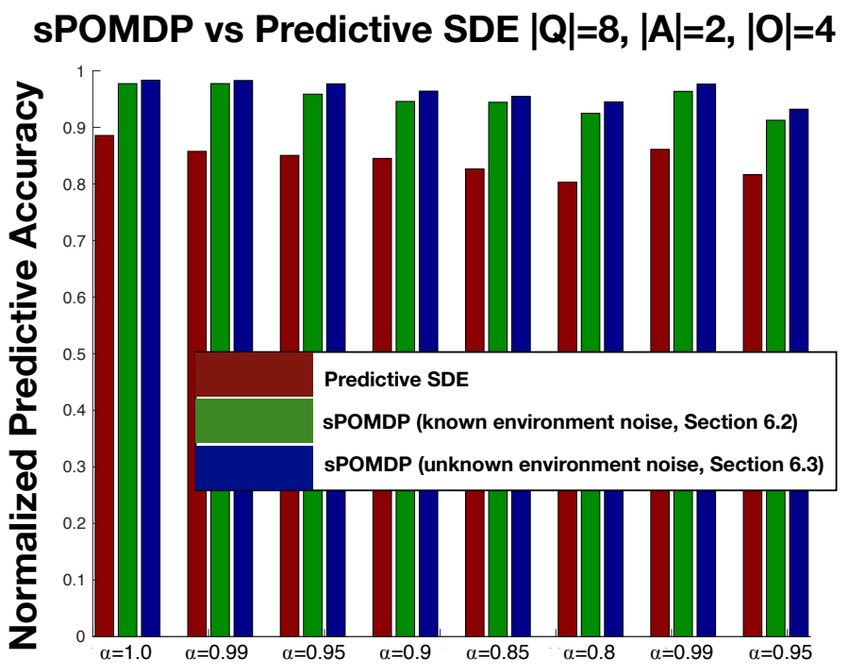
Theorem 2. *For any rewardless $\alpha\epsilon$-POMDP environment $\mathcal{E}$ in which $\alpha > 1/|Q|$ and $\epsilon > 1/|O|$, there exists a* perfect *sPOMDP model $\mathcal{M}$ where $|M| \leq |Q|$, $|S| \leq |Q| - 1$, and each SDE $s \in S$ has, at most, $|Q| - 1$ actions, where $|Q|$ is the number of environment states in the* minimal *representation of $\mathcal{E}$.*

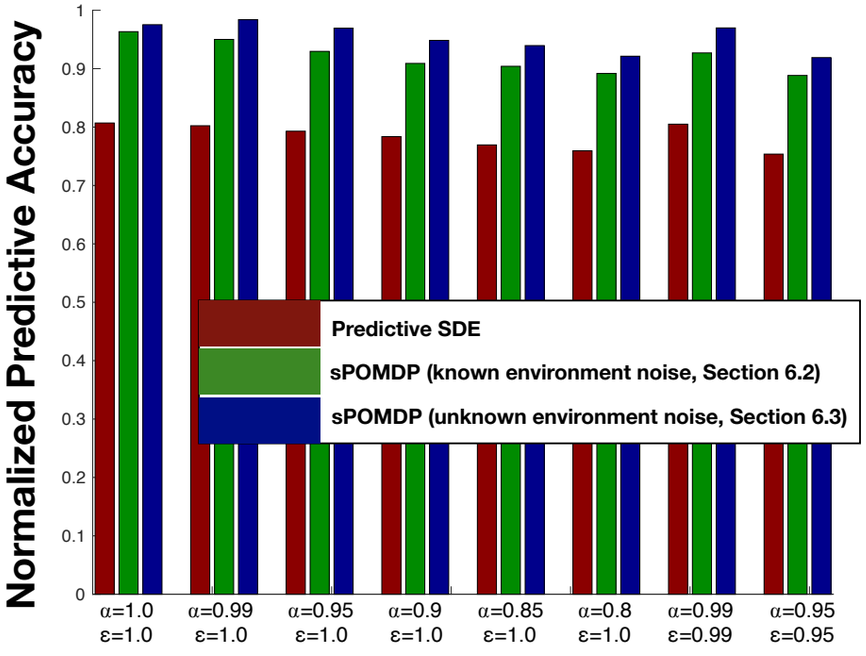
- The constructive proof of **Theorem 2** leads to an **optimal sPOMDP learning procedure**, where, given certain bounds on **alpha** and **epsilon** and an **oracle** that provides the true **model state to model state transition probabilities**, we can provably learn a perfect sPOMDP representation of the given environment.

Experimental Results



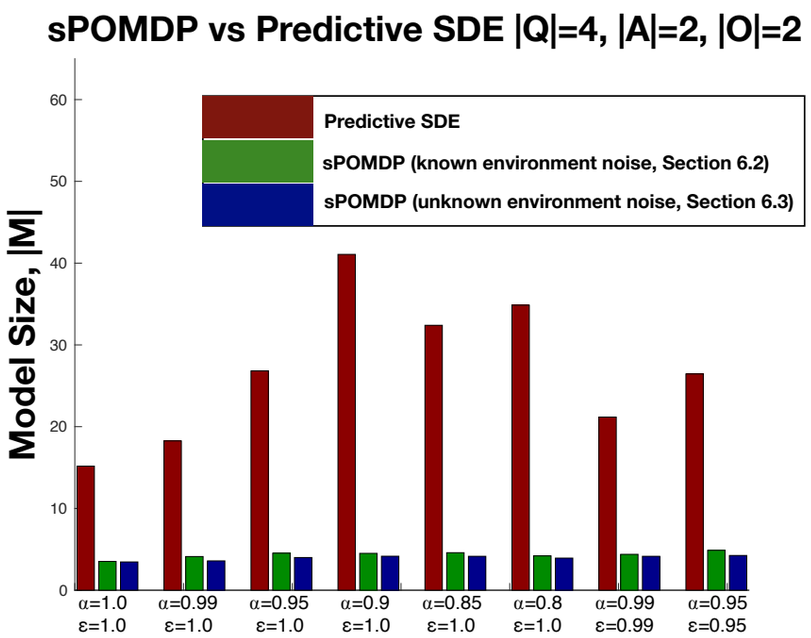


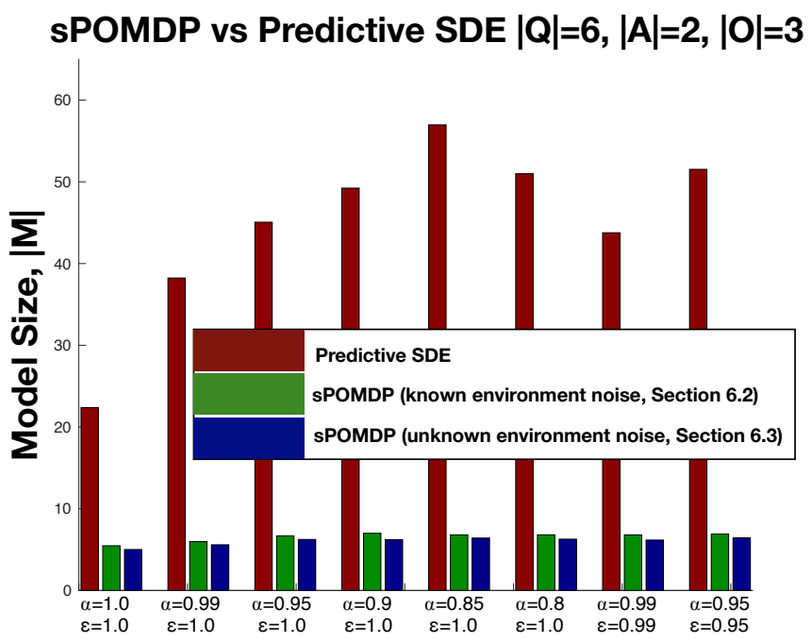


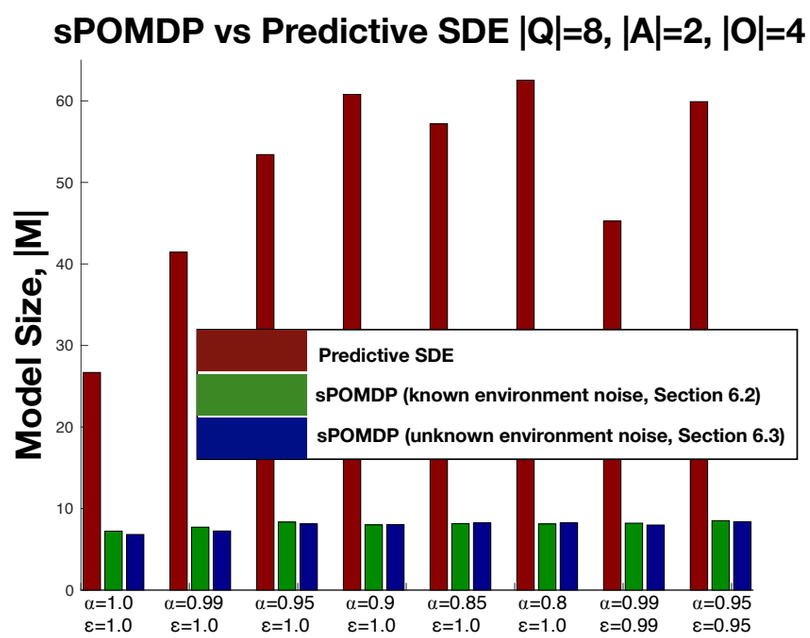


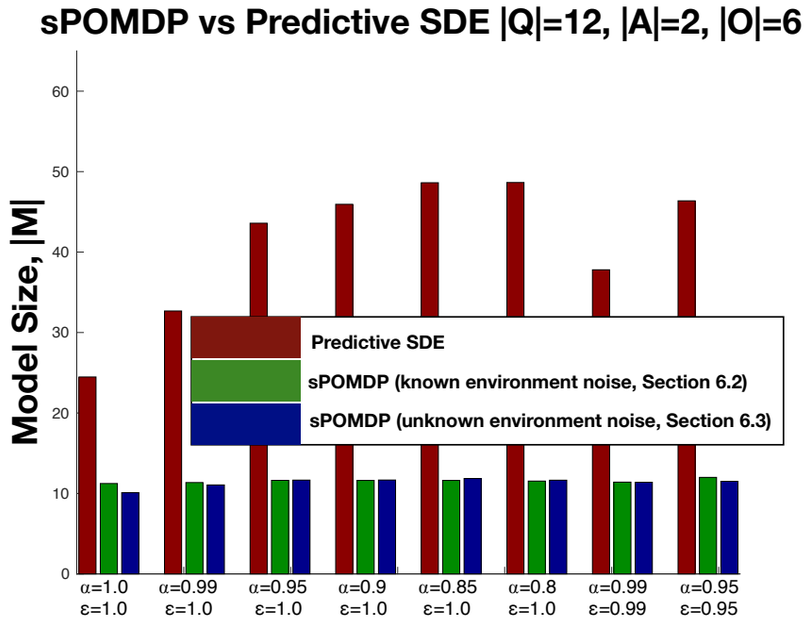
USC Viterbi
School of Engineering
Information Sciences

Experimental Results









USC Viterbi
School of Engineering
Information Sciences

Future Work

- Continuous observation spaces.
- Function approximation techniques for scalability and generalization to unexplored areas of the action/observation space.
  - Balance between **parametric** and **nonparametric** approaches.