Hierarchical POMDPs for Visual Processing Management

Prof. Mohan Sridharan
Chair in Robot Systems
University of Edinburgh, UK
https://homepages.inf.ed.ac.uk/msridhar/
m.sridharan@ed.ac.uk

Sensor Processing Management

- Large amount of raw data from multiple sensors.
- Several sophisticated processing algorithms.
- Processing can vary with the task and environment.
- *Autonomously tailor processing to the task at hand*.
- Pose sequencing of sensor processing operators as *probabilistic sequential decision making* (POMDPs).

POMDP overview

- Defined by the **tuple**: $<S,A,Z,T,O,R>$
- **Belief state (b_t)**: probability over states.
- **Actions**: sensing and information processing.
- **Observations**: noisy action outcomes.
- **Transition function**:

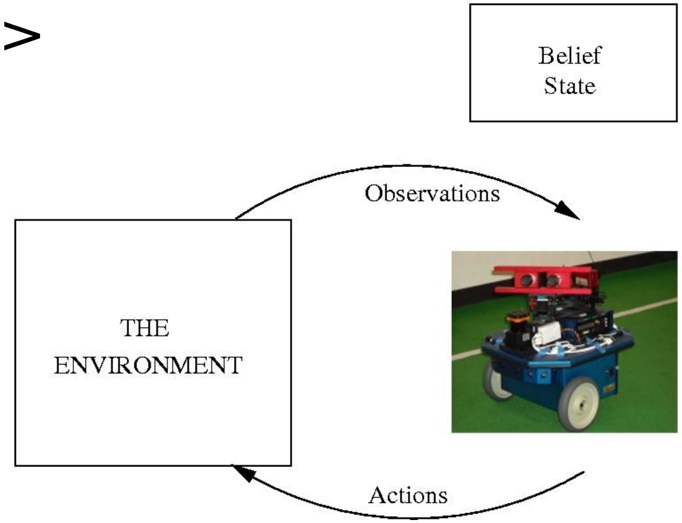
$$T : S \times A \times S' \rightarrow [0, 1]$$

- **Observation function**: $O : S \times A \times Z \rightarrow [0, 1]$
- **Reward specification**:

$$R : S \times A \rightarrow \Re$$

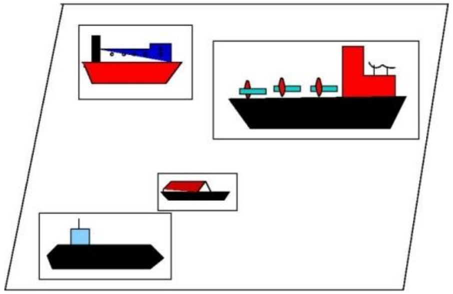
- **Policy** computation:

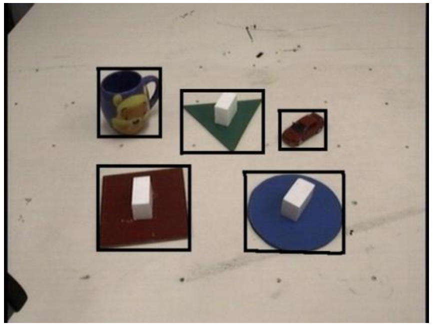
$$\pi : b_t \rightarrow a_{t+1}$$



Example: Visual Processing Management

- Robots and humans jointly reason about objects.
  - *Is there a blue submarine? Where is the red square?*
- **Features:**
  - State not observable, action modifies agent's ***belief***.
  - Non-deterministic actions: action effects not reliable.
  - Computational complexity.
- **Approach:**
  - plan visual processing: *where to look? How to process?*





Observed / Actual	Obj1	Obj2	Obj3
Obj1	0.75	0.25	0.0
Obj2	0.20	0.80	0.0
Obj3	0.15	0.0	0.85

POMDP for one ROI – $<S,A,Z,T,O,R>$

- States: Cartesian product of individual state vectors.

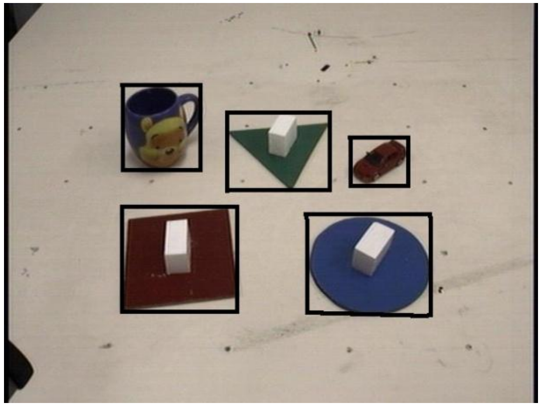
  $$S=S_c \times S_s,\quad S_c=\{\varphi_c^a, R_c^a, G_c^a, B_c^a, M_c^a\}\; S_s=\{\varphi_s^a, C_s^a, T_s^a, S_s^a, M_s^a\}$$
  $$S=(S_c \times S_s) \cup term$$

- Actions: visual+"special".

  $$A=\{Color, Shape, \ldots, yes, no, \ldots\}$$

- Observations: red, green, blue, circle, triangle, square, empty, unknown.

  $$Z_c=\{\varphi_c^o, R_c^o, G_c^o, B_c^o, U_c^o\}$$
  $$Z_s=\{\varphi_s^o, C_s^o, T_s^o, S_s^o, U_s^o\}$$
  $$Z=\cup_{a\in A} Z_a$$



POMDP for one ROI – $<S,A,Z,T,O,R>$

- Transition function. $T : S \times A \times S \rightarrow [0, 1]$
- Observation function. $O : S \times A \times Z \rightarrow [0, 1]$
- Reward specification. $R : S \times A \rightarrow \Re$

- ***Drawback:*** *Exponential state explosion with several ROIs and actions – $25^n + 1$ states for n ROIs with just two visual actions!!*
- **Approach:** *Exploit the existing structure.*

Solving a POMDP

- Updating beliefs:

$$b_{t+1}(s') = \frac{O(s', a_t, o_{t+1}) \sum_{s \in S} T(s, a_t, s')\, b_t(s)}{P(o_{t+1} | a_t, b_t)}$$

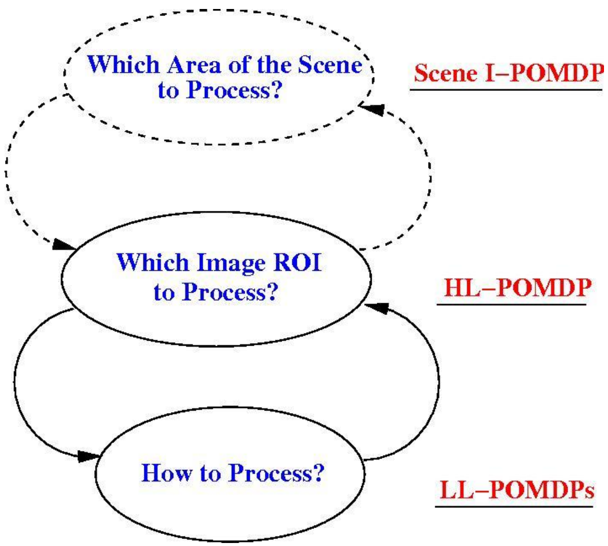
- Find hyperplanes/vectors that are most *aligned* with belief.

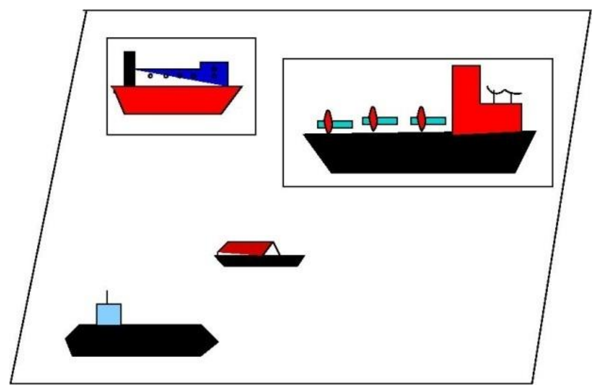
$$a_t = argmax_{a \in A} \left(\max_{v \in V(a)} b_t . v \right)$$

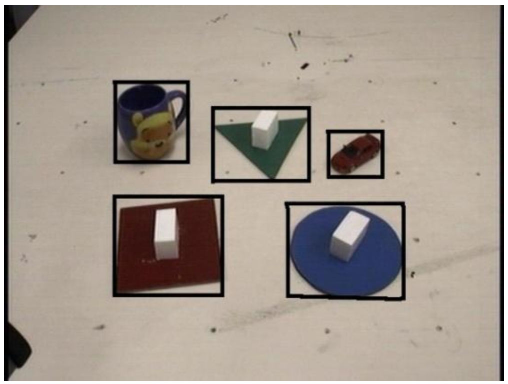
- Linear/dynamic programming – does not scale well ☹
- **Approach:**
  - *Exploit the existing structure!*
  - *Simplify using techniques from other disciplines!*

Hierarchical POMDP Formulation

- Proposed solution: Hierarchical Planning in POMDPs – HiPPo ☺
  - InfoMax POMDP to chose area of scene.
  - One POMDP for planning the processing actions on each ROI.
  - Higher-level POMDP to choose one of the LL-POMDPs at each step.







HiPPo – LL Formulation

- Operates on single ROI.

- Key points:
  - Observation functions learned.
  - Transition function is an identity matrix, *except for special actions and actions that change the state*.
  - Reward function learned. Relative costs and run-time complexity.

$$S = (S_c \times S_s) \cup term$$

$$A = \{Color, Shape\} \cup A_s$$

$$Z_c = \{\varphi_c^o, R_c^o, G_c^o, B_c^o, U_c^o\}$$

$$Z_s = \{\varphi_s^o, C_s^o, T_s^o, S_s^o, U_s^o\}$$

$$T_a = I, \quad \forall a \notin A_s \qquad Z = \cup_{a \in A} Z_a$$

$$O: S \times A \times Z \rightarrow [0, 1]$$

$$R: S \times A \rightarrow \Re$$

$$\forall s \in S$$

$$R(s, shape) = -1.25 \cdot f(ROIsize)$$

$$R(s, color) = -2.5 \cdot f(ROIsize)$$

$$R(s, special-actions) = \pm 100 \cdot \alpha$$

HiPPo – HL Formulation

- HL-POMDP:
  - State space: object presence in different combinations of regions.
  - Action u_i means process ROI R_i.
  - FR_i means desired object found in R_i.

$$S^H = \{R_1 \wedge R_2, R_1 \wedge \neg R_2, \neg R_1 \wedge R_2, \neg R_1 \wedge \neg R_2, term^H\}$$

$$A^H = \{u_1, u_2, A_s\} \quad A_s = \{sR_1 \wedge R_2, sR_1 \wedge \neg R_2, \ldots\}$$

$$Z^H = \{\neg FR_i, FR_i \mid R_i \in ROIs\}$$

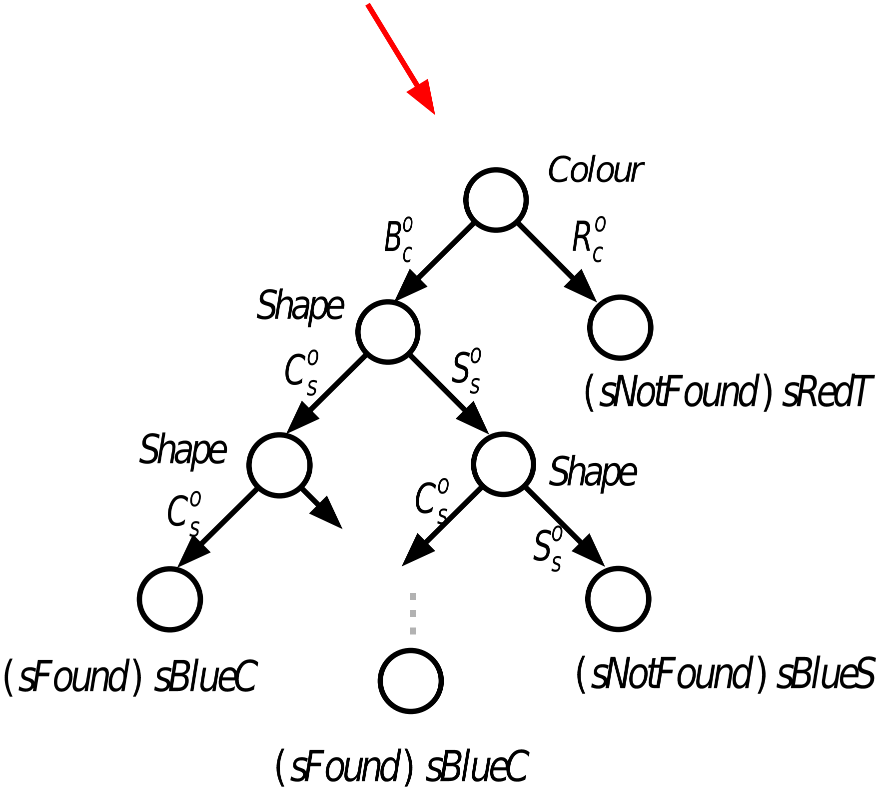
$$T_a = I, \quad \forall\, a \in \{u_1, u_2\}$$

- Key points:
  - Observation functions O^H and costs R^H derived from the policy trees of LL-POMDPs.
  - LL-POMDPs are black boxes that return definite labels (not belief densities).

Estimating O^H and R^H

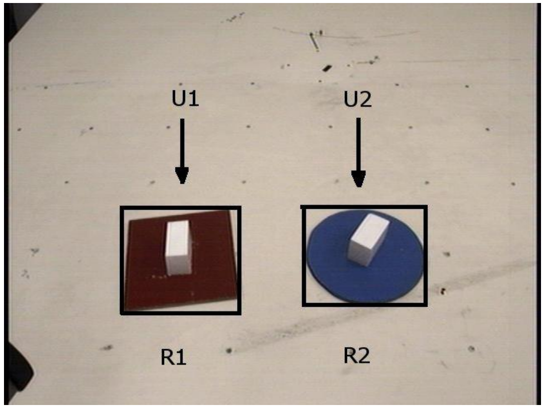
- Parse LL policy tree, conditioned on the HL state.

$$\Pr(FR_1 | R_1 = BlueC)$$



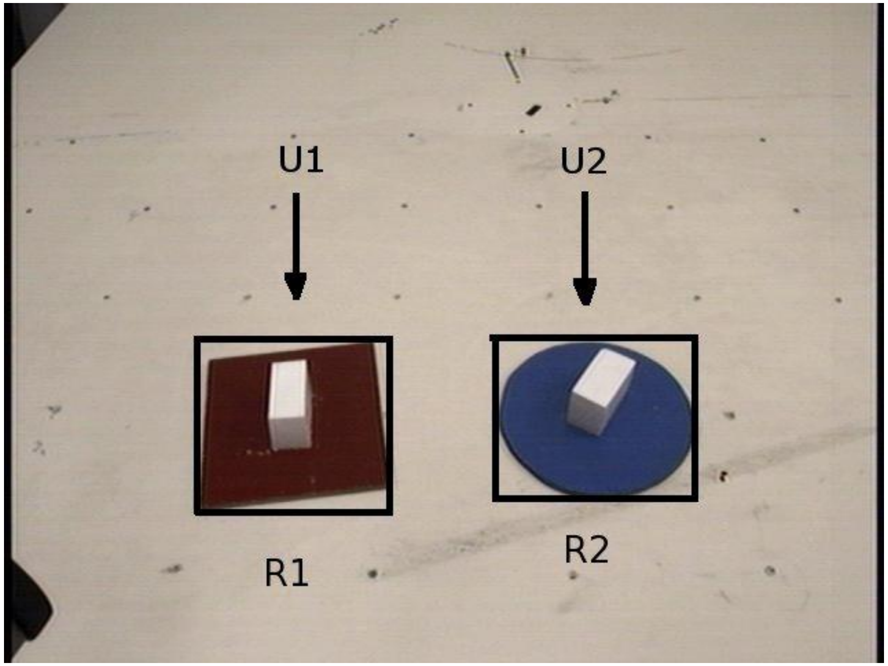
Illustrative Example

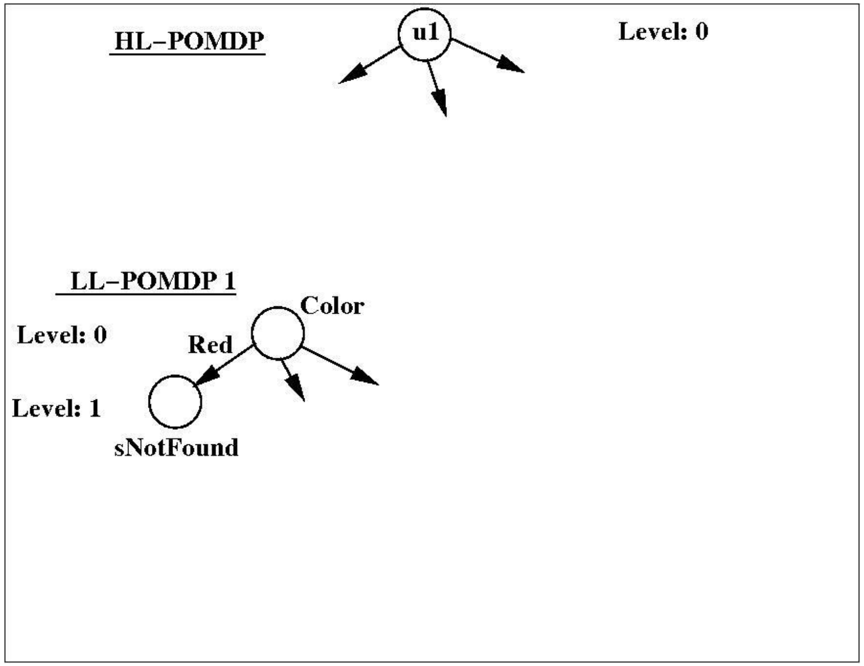
- Consider the scene with two ROIs extracted.



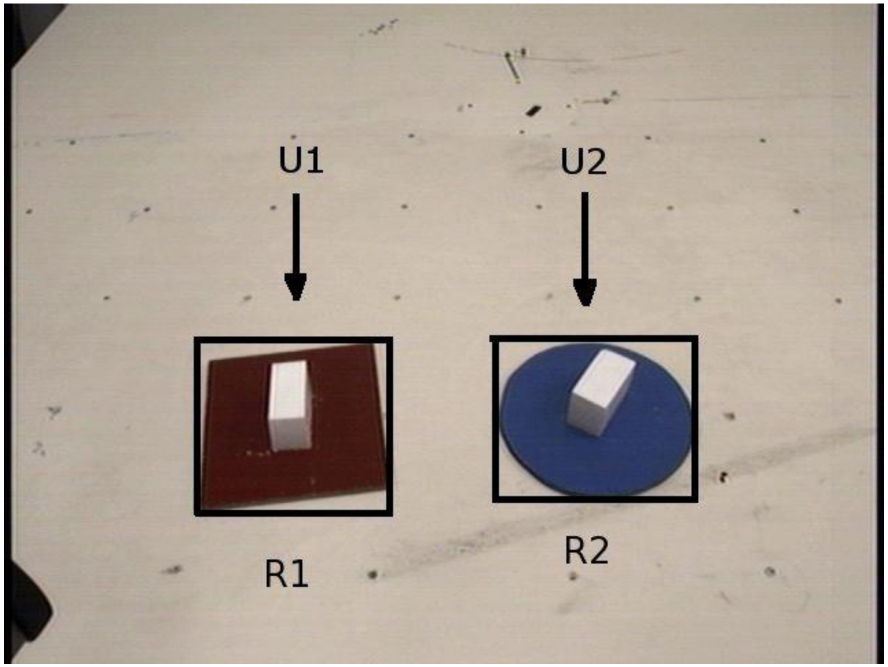
- Query: *Where are the blue circles?*
- Available operators: Color, Shape, SIFT.

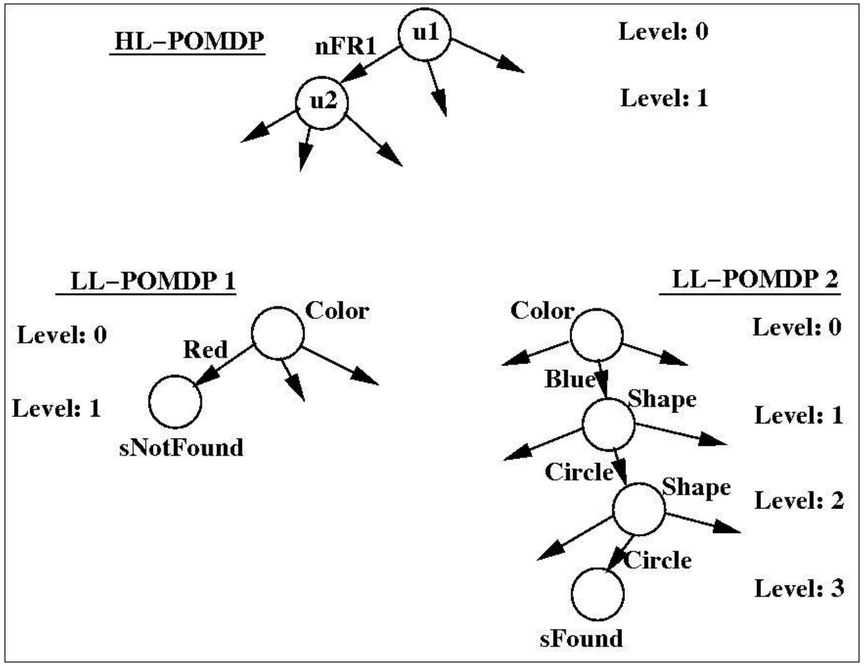
Example – Where are the Blue Circles?



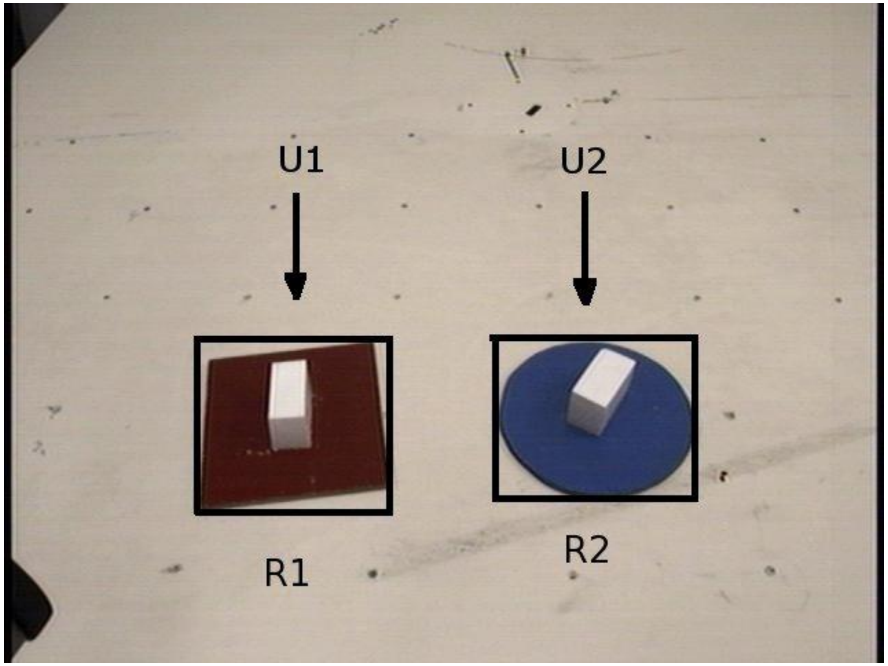


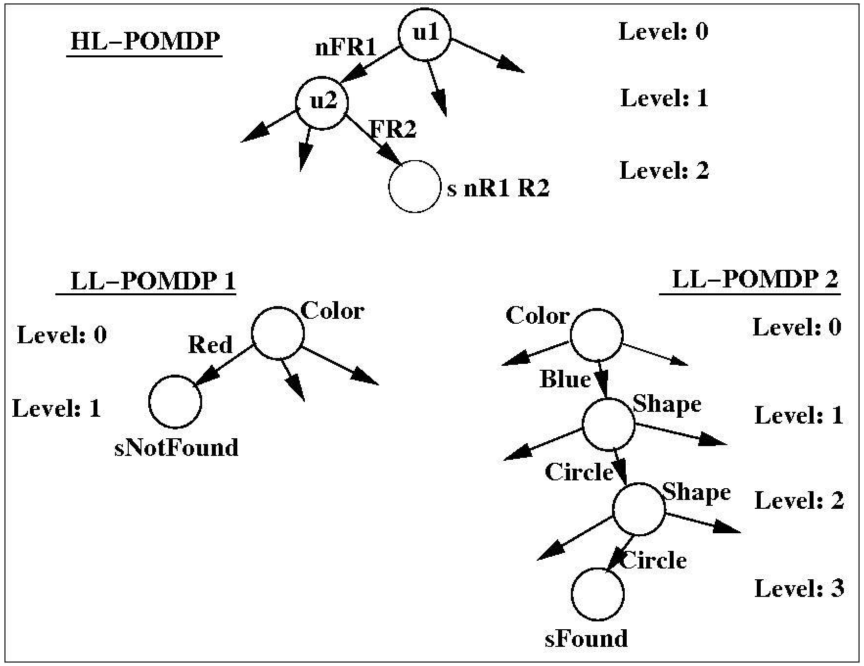
Example – Where are the Blue Circles?





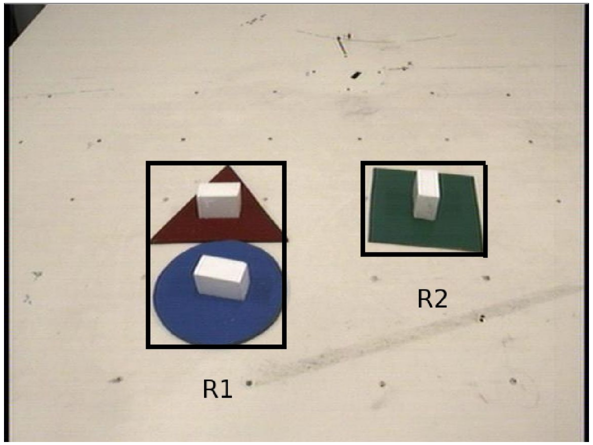
Example – Where are the Blue Circles?

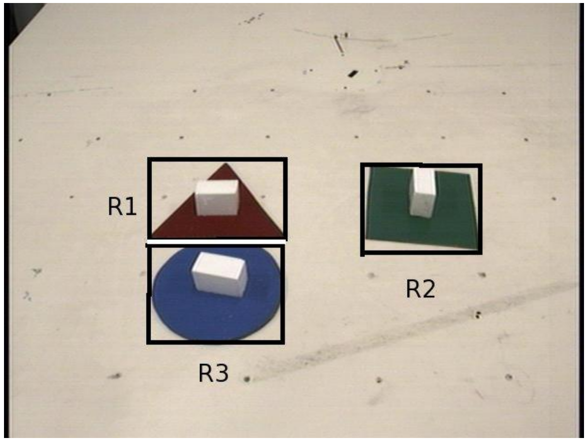




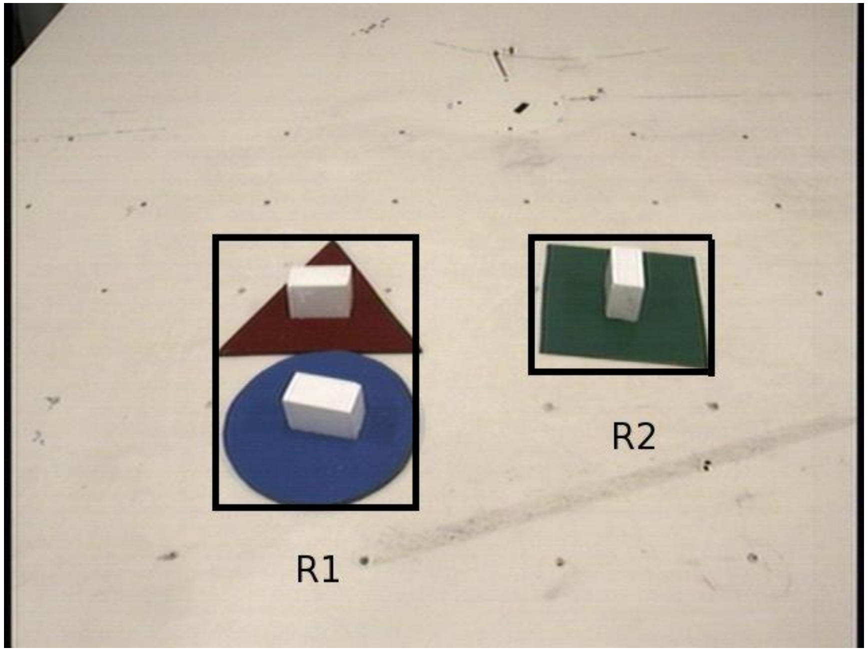
Overlapping ROIs

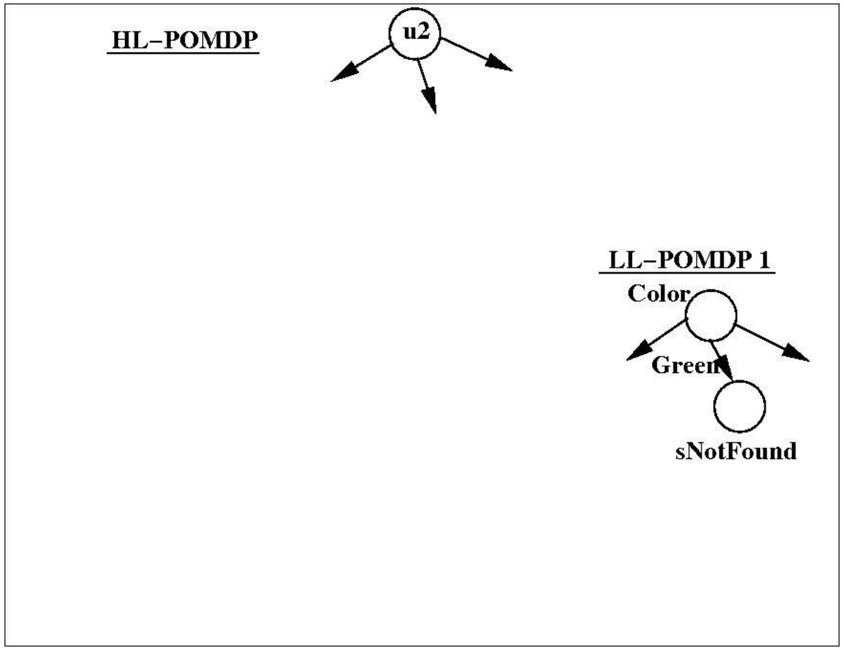
- ROIs often contain multiple objects:
  - Region-split is an useful operator.
- Planning through splits is problematic:
  - Splitting a region changes the state space – no longer the same POMDP!
- Different split action for each basic operator, e.g. split-color:
  - Each action is a split plus the operator applied to all new regions.



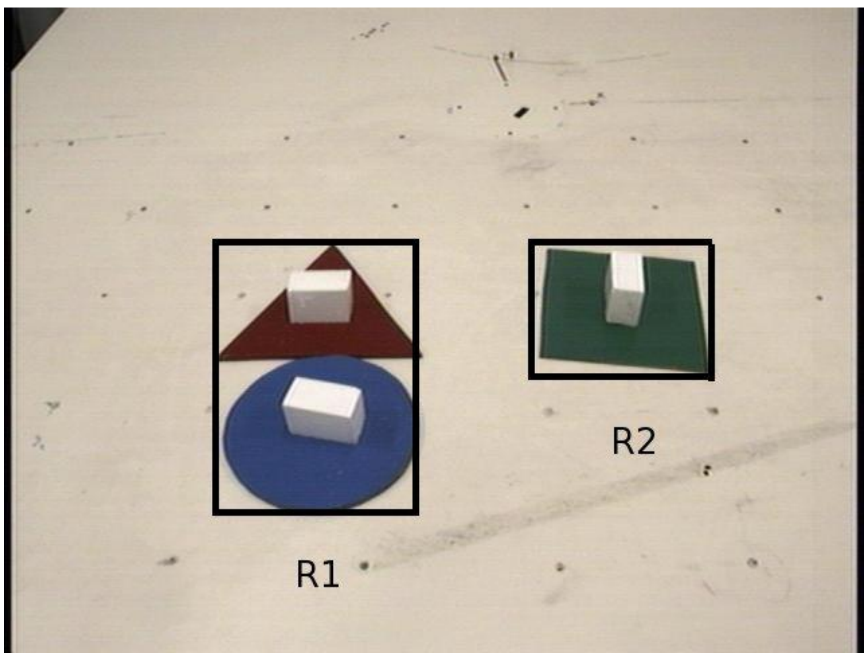


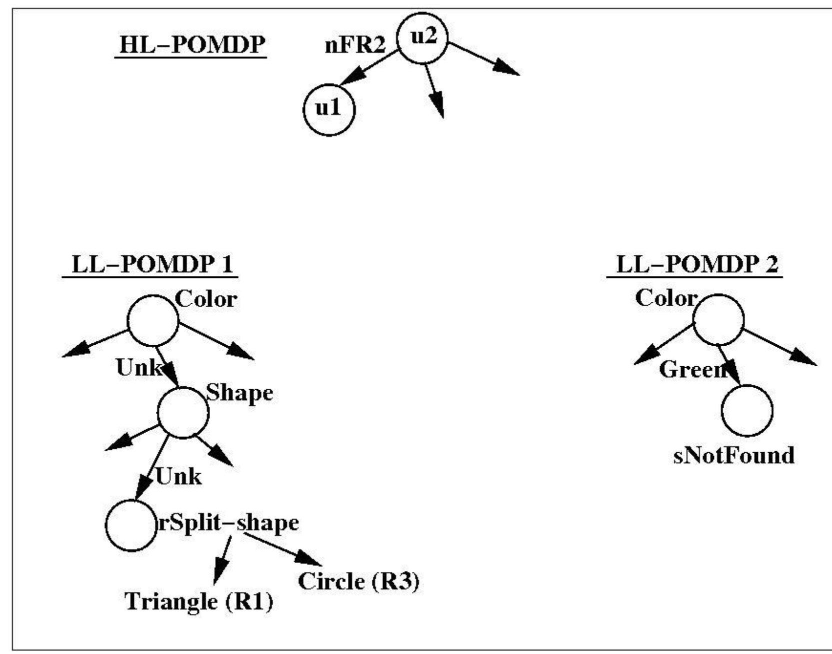
Region-split Example – Step 1



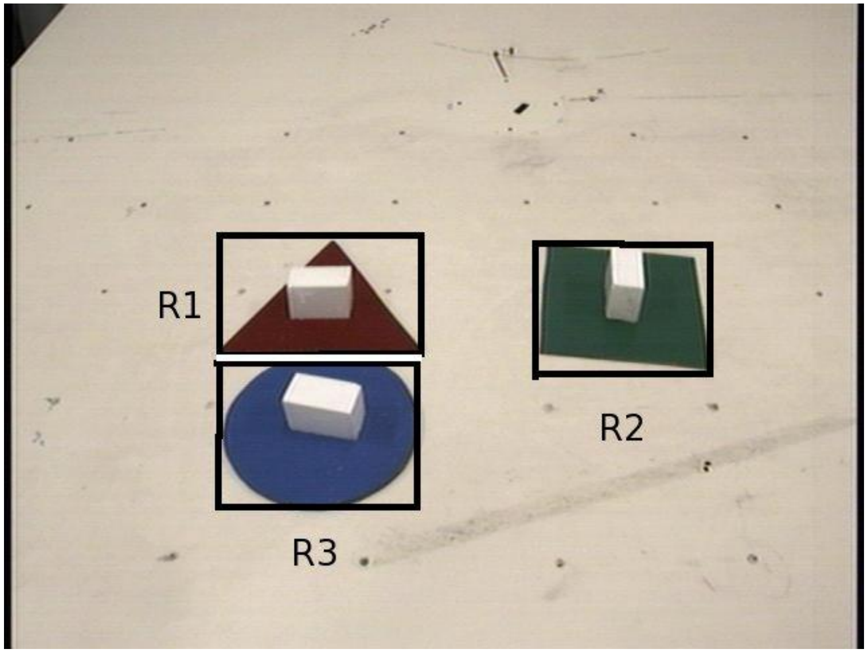


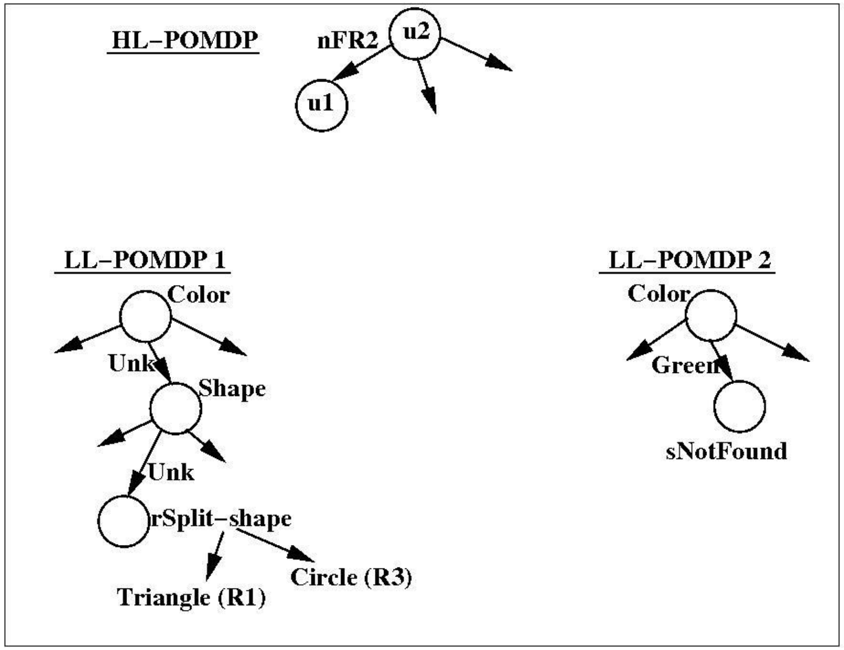
Region-split Example – Step 2



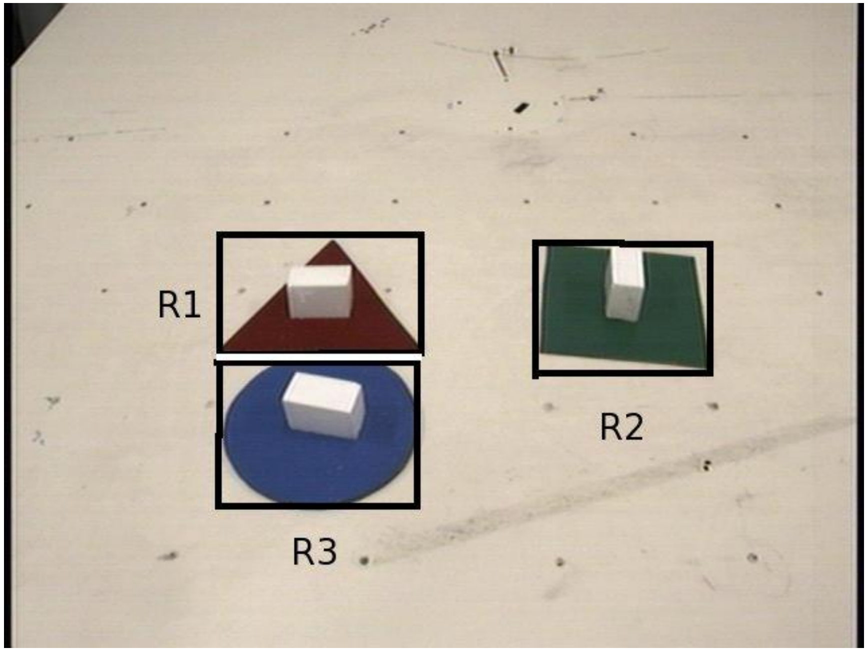


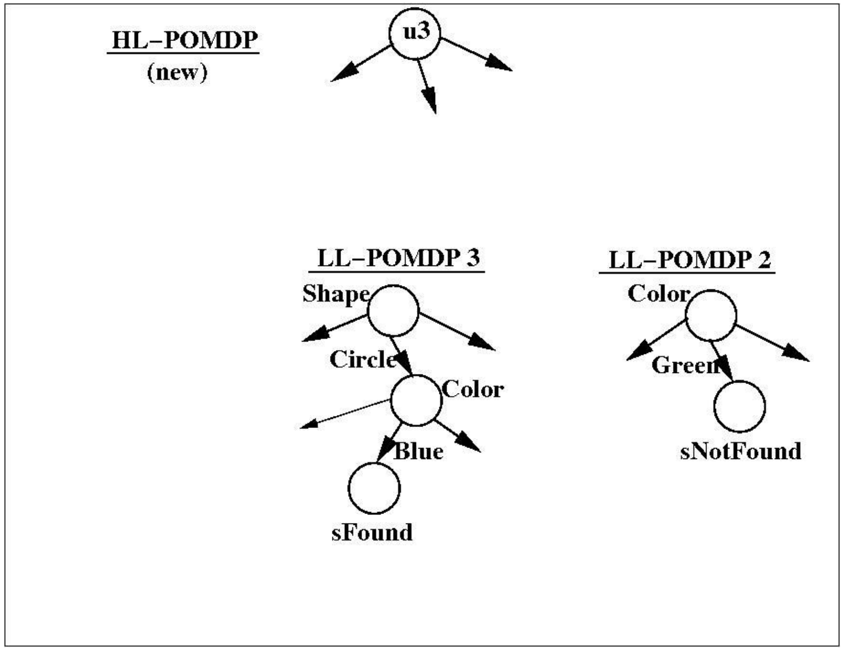
Region-split Example – Step 3



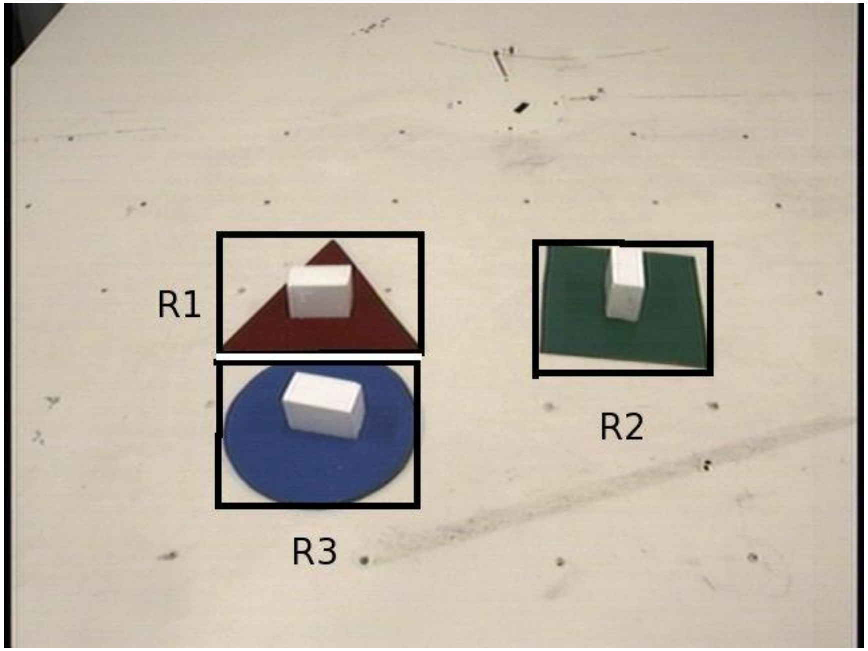


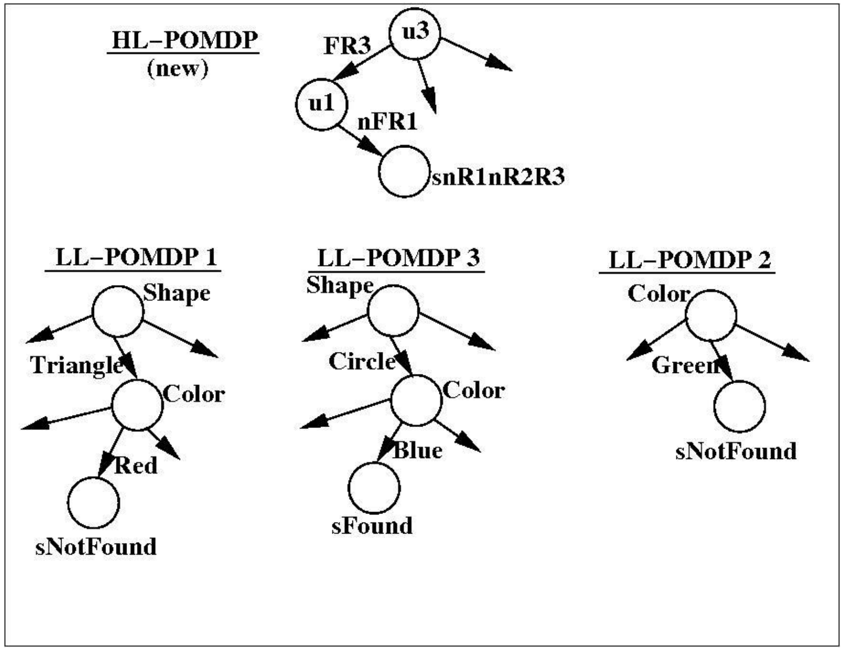
Region-split Example – Step 4





Region-split Example – Step 5





That’s all folks ☺