Interactive Learning of Mappings from Visual Percepts to Actions

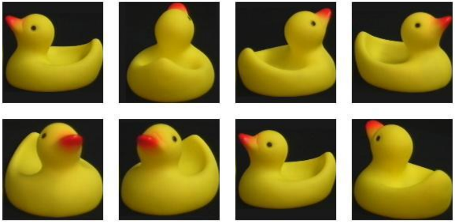
Sébastien Jodogne

Séminaires à Montefiore — 10 novembre 2005

Problem Definition

Data:

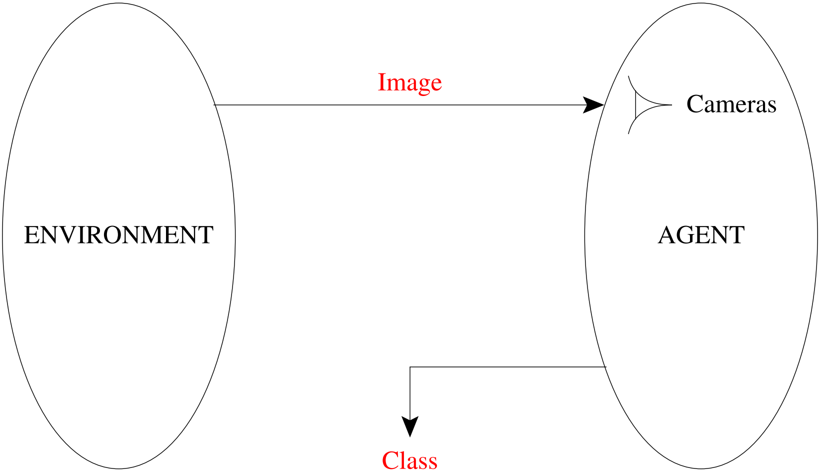
- A set of n classes of objects: {duck, green boat, hamburger, strawberry,...}.
- A set of pictures for each class:



Input: An image of an object: 

Output: The class of this object: "Strawberry".

Abstract View

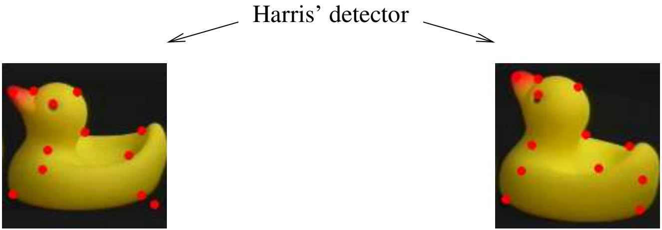


Local-Appearance Methods in Computer Vision

“Focus on robust and informative patterns in the visual signal.”

Local-Appearance Methods in Computer Vision

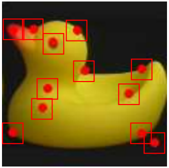
"Focus on robust and informative patterns in the visual signal."

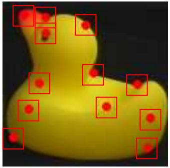


1. Locate robust, informative patterns: the interest points.

Local-Appearance Methods in Computer Vision

"Focus on robust and informative patterns in the visual signal."

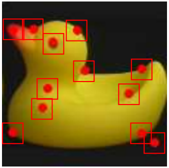


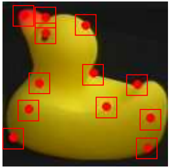


1. Locate robust, informative patterns: the interest points.
2. Compute a description of the patterns: the visual features.

Local-Appearance Methods in Computer Vision

"Focus on robust and informative patterns in the visual signal."

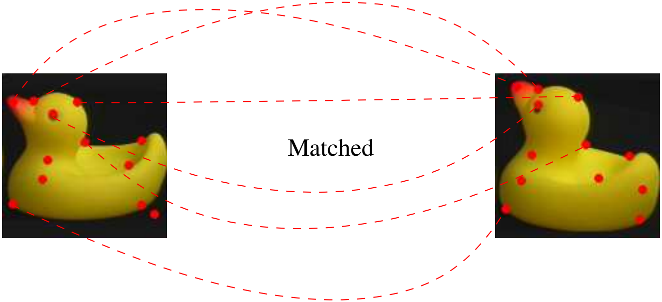




1. Locate robust, informative patterns: the interest points.
2. Compute a description of the patterns: the visual features.
3. Choose a distance on the features (Euclidean,. . .).

Local-Appearance Methods in Computer Vision

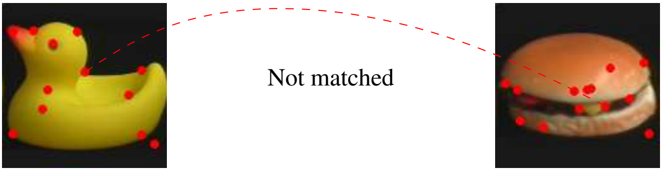
“Focus on robust and informative patterns in the visual signal.”



1. Locate robust, informative patterns: the interest points.
2. Compute a description of the patterns: the visual features.
3. Choose a distance on the features (Euclidean, . . .).
4. Match images if a sufficient number of features match.

Local-Appearance Methods in Computer Vision

"Focus on robust and informative patterns in the visual signal."



1. Locate robust, informative patterns: the interest points.
2. Compute a description of the patterns: the visual features.
3. Choose a distance on the features (Euclidean,. . .).
4. Match images if a sufficient number of features match.

Interest Point Detectors

- Harris,
- Harris-Laplace,
- Harris-affine,
- SIFT detector,
- Random (!) $\Leftarrow$ [Marée et al., 2005],...

Local Description Techniques

- Steerable filters,
- Differential invariants,
- SIFT keypoints,
- Raw pixels (!) $\Leftarrow$ [Marée et al., 2005],...

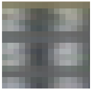
Algorithm

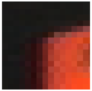
- Each visual feature votes for one class.
- An image is mapped to the class that has the most votes.


$\longrightarrow$ cup


$\longrightarrow$ tank


$\longrightarrow$ plate


$\longrightarrow$ phone


$\longrightarrow$ truck


$\longrightarrow$ frog

Advantages

- Flexible,
- Robust to partial occlusions,
- No need for segmentation,
- No need for 3D models of objects.

Improvements

Take spatial relationships into consideration:

- Semilocal constraints,
- Geometric model of a soccer player $\Leftarrow$ [Gabriel et al., 2005],
- Probabilistic graph-based model $\Leftarrow$ [Scalzo et al., 2005]

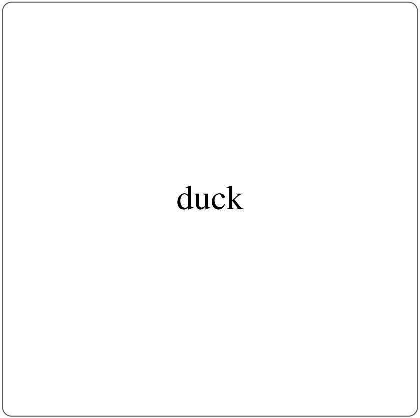
Informative Features

- We use all the visual features, even if they are not informative.
- Orthogonal point of view: Use only informative features.

Incremental Selection Process

- Build a binary decision tree:
  - Each internal tests the presence of one informative feature,
  - Each leaf outputs one visual class.
- Standard Machine Learning algorithms are applicable:
  - Maximize mutual information at each internal node.

Illustration

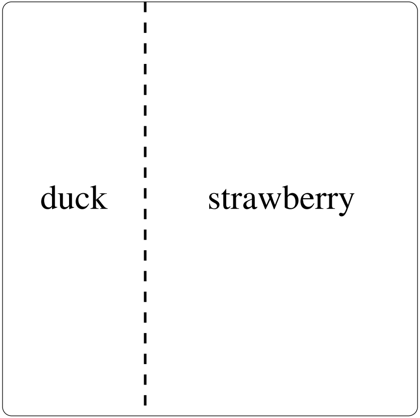


Visual space

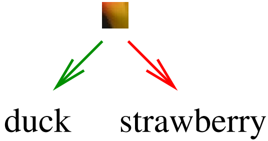
duck

Decision tree

Illustration

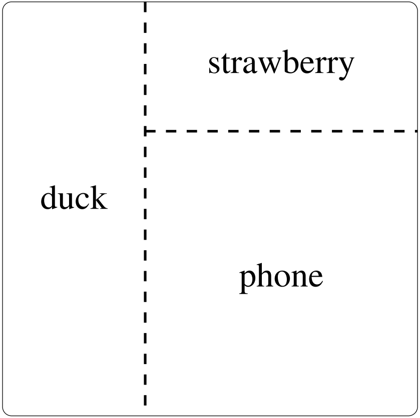


Visual space

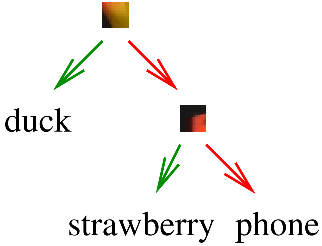


Decision tree

Illustration

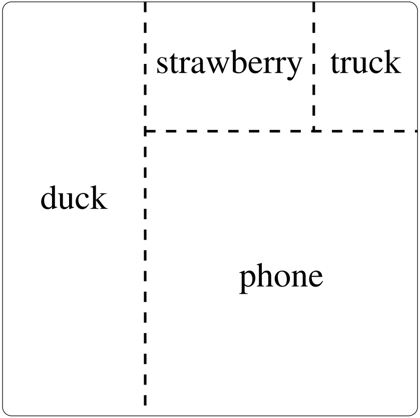


Visual space

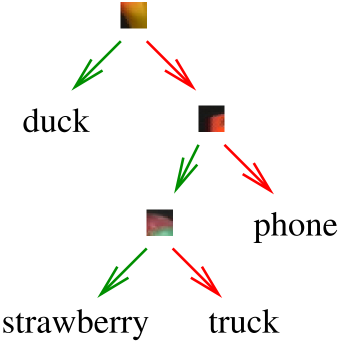


Decision tree

Illustration



Visual space



Decision tree

Image Classification

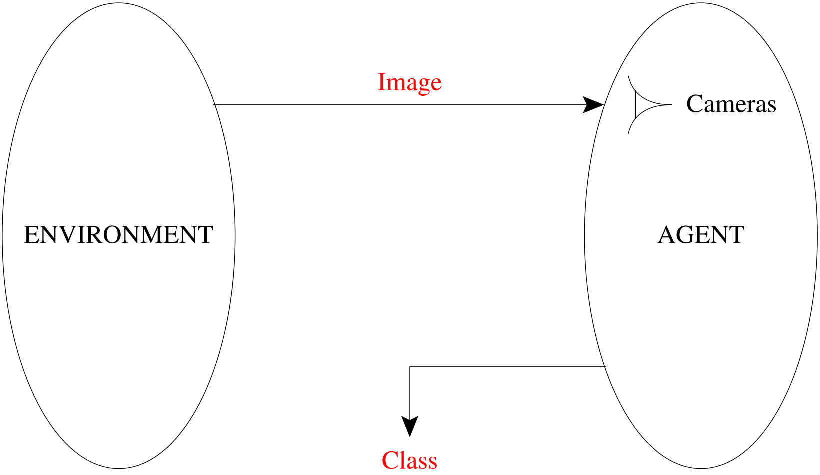
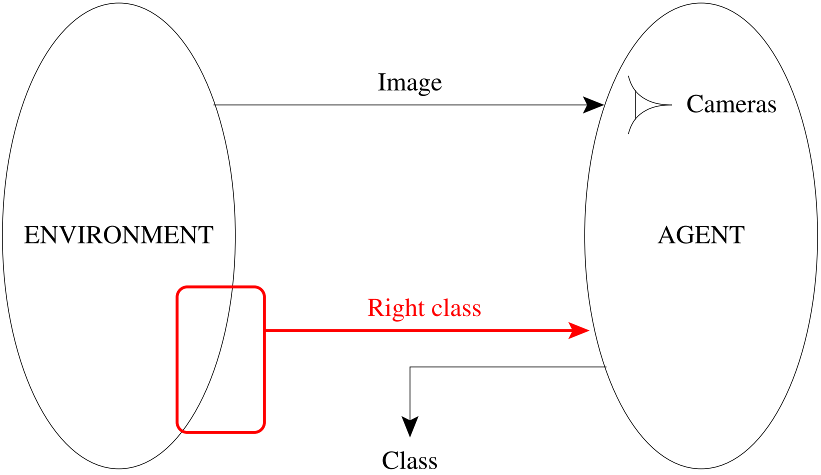
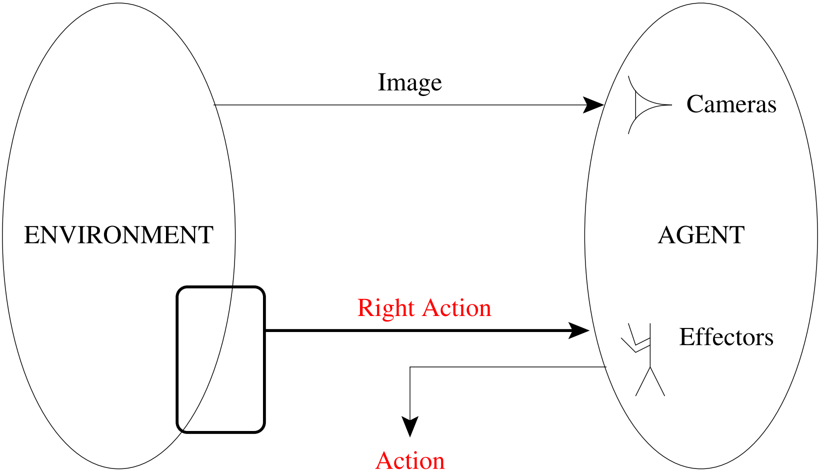


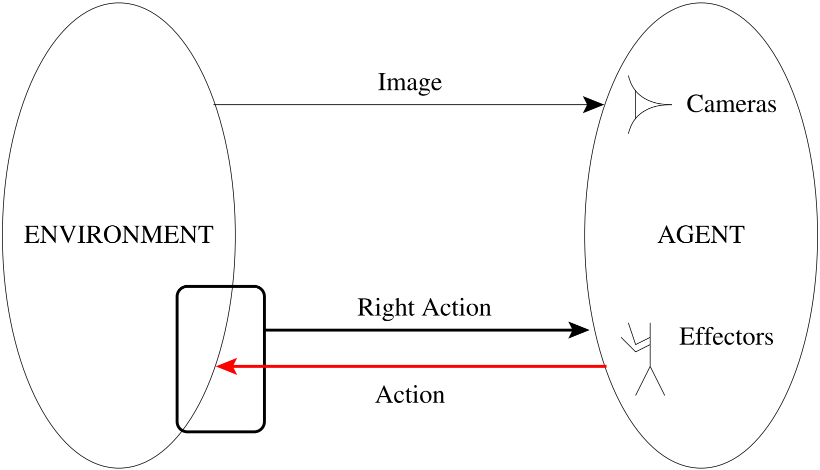
Image Classification during Learning



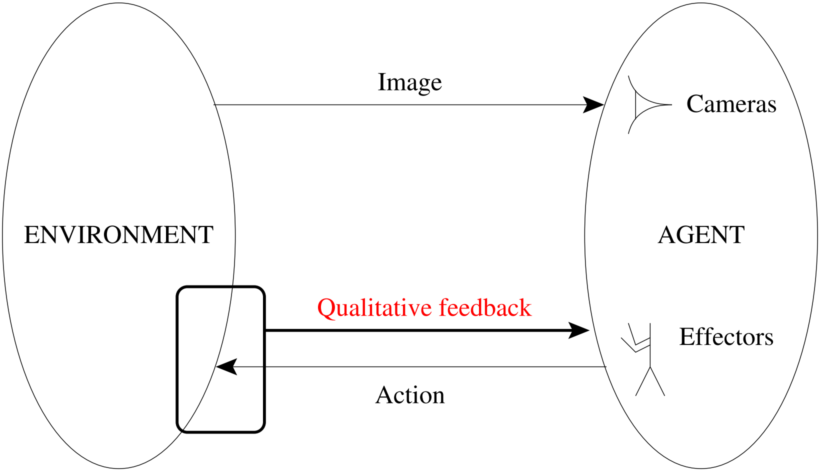
Vision-for-Action (Open-Loop)



Vision-for-Action (Closed-Loop)



Vision-for-Action (without Supervision)



Fact

Most everyday tasks can be solved by connecting images to the appropriate reactions (direct image-to-action mappings).

Kind of learning	Goal of the agent	Feedback
Supervised	"Do the right action, as told by my teacher"	The right action
Reinforcement	"Maximize my rewards"	Reward/punishment
Unsupervised	"Structure my percepts"	No hint

Reinforcement Learning

Historical question: How do animals learn?

RL Answer: Through their interactions with the environment, that give rise to a positive or negative feedback (trial-and-error).

More precisely: By learning a percept-to-action mapping that maximizes, over time, an evaluation of its performances given by the environment.

Examples :

- A dog learns to sit down by receiving sugars from its master.
- A robotic hand learns to grasp objects by receiving an information about the quality of the grasp from the physical world.

Reinforcement Learning Process

Basic principles :

- The agent knows nothing about its environment.
- It only knows about its percepts and actions.
- After each interaction, it receives a numerical feedback.
- It progressively improves its policy by trying new actions.

Advantages :

- No need of a physical model of the environment (while it can accelerate learning). Therefore :
  - General approach,
  - Simple design.
- Allows a dynamical adaptation when the environment changes.

Three Main Problems

1. The reinforcement is often delayed (e.g., in chess).

   $\Leftrightarrow$ Temporal credit assignment problem!

2. How to design a suitable reinforcement signal?

   $\Leftrightarrow$ Credit structuration problem!

3. Should the agent:
   - Innovate (i.e., randomize) to find new good actions to take?
   - Take advantage of its history to re-do fruitful actions?

   $\Leftrightarrow$ Exploitation vs. exploration dilemma!

Modeling the Environment

- Discrete time.
- Markovian probabilistic dynamics:

$$P\left(s_{t+1} = s' \mid \text{History}\right) = P\left(s_{t+1} = s' \mid s_t = s, a_t = a\right).$$

- Reinforcement function $r(s, a)$.

Markov Decision Process (MDP)

- S: finite set of states;
- A: finite set of actions;
- $\mathcal{T}(s, a, s')$: probabilistic transition function;
- $r(s, a)$: numerical reinforcement function.

Modeling the Agent

Sensors

- No direct access to s_t.
- Sensors convert a state $s_t \in S$ to a percept $p_t \in P$.

initial state

$$p_0 \xrightarrow{a_0} p_1 \xrightarrow{a_1} p_2 \cdots p_t \xrightarrow{a_t} p_{t+1} \longrightarrow \cdots$$

$$\downarrow r_1 \qquad \downarrow r_2 \qquad \downarrow r_{t+1}$$

Reinforcement Learning (RL) Process

Inputs: A database of interactions $\langle p_t, a_t, r_{t+1}, p_{t+1} \rangle$.

Output: An optimal control policy $\pi^* : P \mapsto A$.

Temporal Credit Assignment

We don't want to maximize *immediate rewards* (the sequence of r_t), but the *rewards over time*.

Return at Time t

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=1}^{\infty} \gamma^k r_{t+k},$$

where $\gamma \in [0, 1[$ is the discount factor giving the current value of the future rewards (i.e., a reward perceived k units of time later is only worth γ^k what it would represent currently).

- $\gamma = 0 \Leftrightarrow$ *short-sighted* agent : maximize immediate rewards.
- $\gamma \to 1 \Rightarrow$ agent with a more and more faraway horizon.

Goal of Reinforcement learning

Optimal Control Policy π^*

Policy that maximizes the expected return at any time!

Algorithms

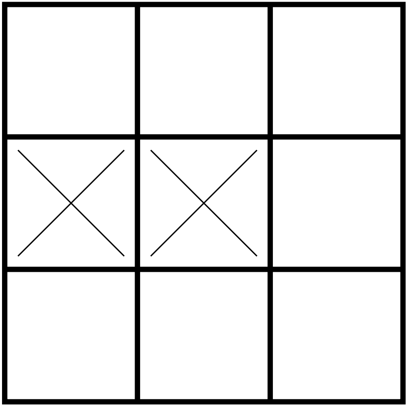
Model-based: Value Iteration, Policy Iteration,...

Model-free: Q Learning, SARSA, Actor-Critic,...

Sorry, but proving the existence of such a policy and the way to get it is far outside the scope of this talk!

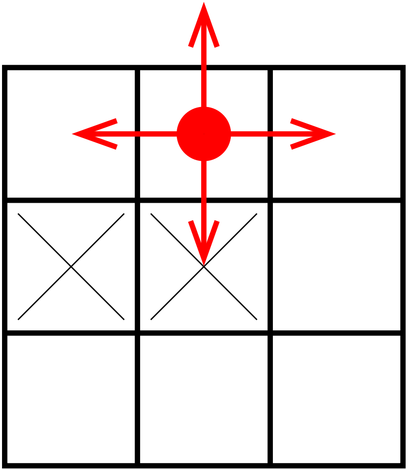
Example: A Visual Navigation Task

- Consider a discrete maze with walls.



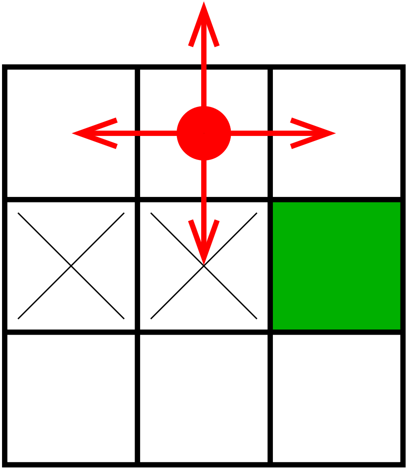
Example: A Visual Navigation Task

- Consider a discrete maze with walls.
- An agent moves in the maze (penalty of -1 by move).



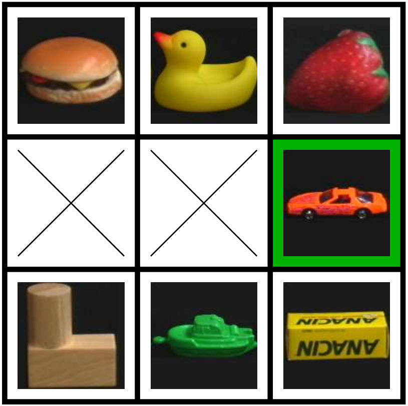
Example: A Visual Navigation Task

- Consider a discrete maze with walls.
- An agent moves in the maze (penalty of -1 by move).
- The agent much reach the exit as fast as possible (reward of $+100$).



Example: A Visual Navigation Task

- Consider a discrete maze with walls.
- An agent moves in the maze (penalty of -1 by move).
- The agent much reach the exit as fast as possible (reward of $+100$).
- The sensors return a picture of an object that depends on the cell.



Reinforcement Learning on Visual Tasks

RL: Pros

- Fully automatic;
- Flexible;
- Biologically plausible.

RL: Cons

Visual tasks are intractable, because of an extremely high-dimensional, noisy input space.

Our Research Interest

Apply RL on visual tasks!

Previous Work on Large, Discrete Input Spaces

- G Algorithm [Chapman & Kaelbling, 1991],
- "Selective Attention" in U Tree [McCallum, 1996].
- ...

Basic Idea

Build a decision tree that selects Boolean features, by iteratively removing perceptual aliasing $\Leftrightarrow$ Local-Appearance!

Similar Algorithms for Continuous Input Spaces

- Darling [Salganicoff, 1993],
- Continuous U Tree [Uther & Veloso, 1998],
- Variable Resolution Grids [Munos & Moore, 2002].
- ...

What Kind of Features could be Used?

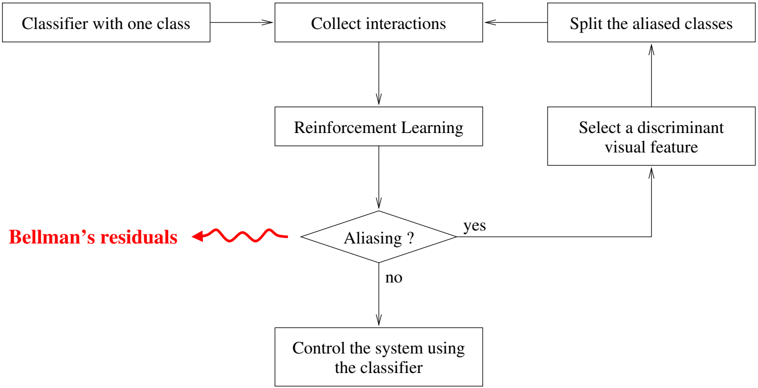
- Pixels? Very numerous, and not very informative.
- Patches? More informative, but still numerous.
- In general, robustness to noise, as well as to illumination and viewpoint changes is desirable.

Our Contributions

- Take advantage of the visual features used in local-appearance methods from Computer Vision.
- A state-splitting rule based upon Bellman's residuals and mutual information.

Reinforcement Learning of Visual Classes (RLVC)

“A feature is selected only once it has proved its relevance”



Summary

Visual Stimulus	
$\Downarrow$	Interest point detector
Informative Locations	
$\Downarrow$	Local description
Visual Features	
$\Downarrow$	Distance
Symbolic Features	
$\Downarrow$	Feature selection (decision tree)
Visual Classes	
$\Downarrow$	Reinforcement Learning
Image-to-action Mapping	

Discriminant Feature Selection

For each $(\mathcal{C}(p), a)$:

Discriminant Feature Selection

For each $(\mathcal{C}(p), a)$:

1. Compute Bellman's residuals from DB $\{\langle p_t, a_t, r_{t+1}, p_{t+1}\rangle\}$:

$$r_{t+1} + \gamma \max_{a' \in A} Q^*(\mathcal{C}(p_{t+1}), a') - Q^*(\mathcal{C}(p_t), a_t).$$

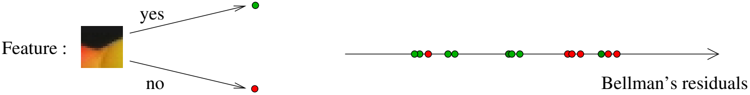
Discriminant Feature Selection

For each $(\mathcal{C}(p), a)$:

1. Compute Bellman's residuals from DB $\{\langle p_t, a_t, r_{t+1}, p_{t+1}\rangle\}$:

$$r_{t+1} + \gamma \max_{a' \in A} Q^*(\mathcal{C}(p_{t+1}), a') - Q^*(\mathcal{C}(p_t), a_t).$$

2. Sort them and apply the CART learning rule once (variance reduction).



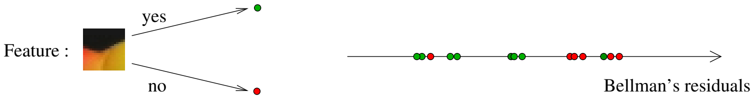
Discriminant Feature Selection

For each $(\mathcal{C}(p), a)$:

1. Compute Bellman's residuals from DB $\{\langle p_t, a_t, r_{t+1}, p_{t+1}\rangle\}$:

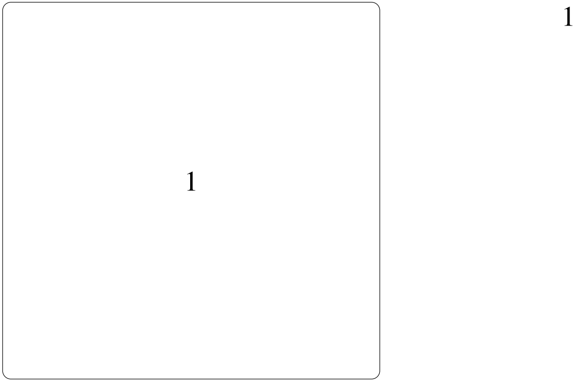
$$r_{t+1} + \gamma \max_{a' \in A} Q^*(\mathcal{C}(p_{t+1}), a') - Q^*(\mathcal{C}(p_t), a_t).$$

2. Sort them and apply the CART learning rule once (variance reduction).



3. This assumes a deterministic environment. In practice, it works also with non-determinism, if a suitable hypothesis test (e.g. Student's $t-$test) is applied.

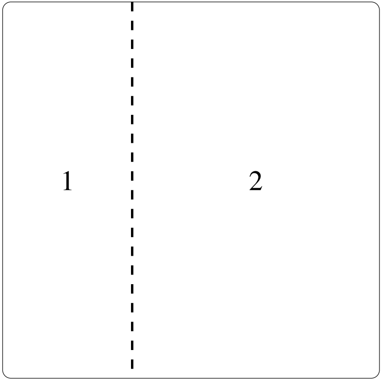
Illustration



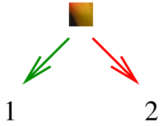
Visual space | Decision tree

$\Rightarrow$ Start with full aliasing

Illustration

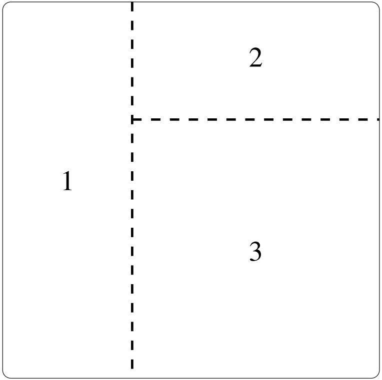


Visual space

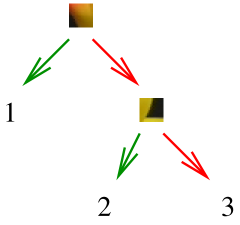


Decision tree

Illustration

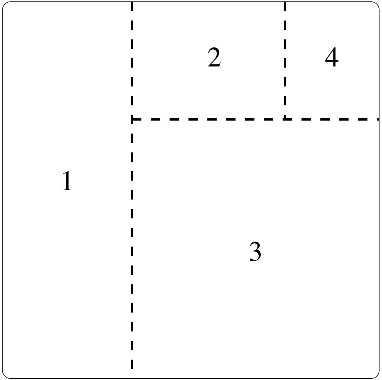


Visual space

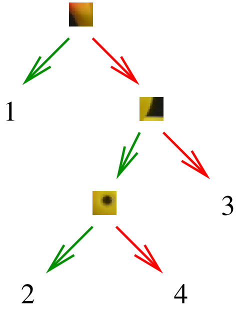


Decision tree

Illustration



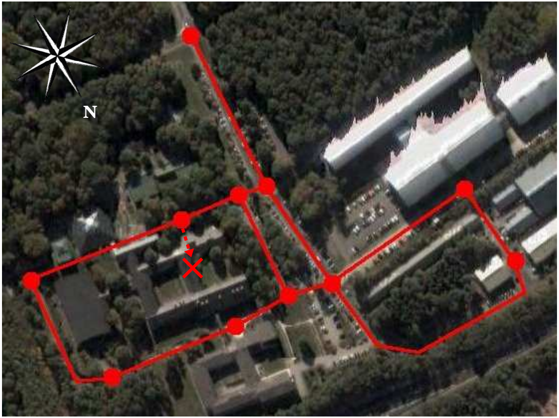
Visual space



Decision tree

⇒ Adaptive discretization (target zero Bellman's residuals)

Visual Navigation around Montefiore

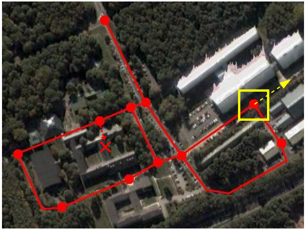


(c) GoogleMap

- State space: (p, d), i.e. $\{11 \text{ places}\} \times \{4 \text{ directions}\}$.
- Action space: {turn left, turn right, move forward}.
- Goal: Enter Montefiore Institute.

Optimal Control Policy

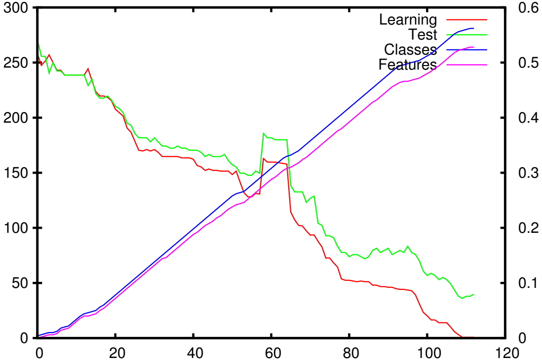
- The agent does not have direct access to (p, d).
- It perceives only a picture of the area ahead.
- Database of $11 \times 4 \times 24 = 1056$ images (1024×768 pixels).





RLVC Parameters

- SIFT keypoint detector [Lowe, 2004].
- Mahalanobis distance.
- Learning set: $11 \times 4 \times 18 = 792$ possible percepts.
- Test set: $11 \times 4 \times 6 = 264$ possible percepts.



Results of RLVC

- Visual classes: 281;
- Distinct SIFT features: 264;
- Policy error: 0.1% on LS, 8% on TS.

(i) Compacting Image-to-Action Mappings

Problem with RLVC

- Cannot undo splits that are subsequently proved useless.
- Can get stuck in local optima.
- In a word, greedy algorithm!

Possible Solution

- Periodically, aggregate visual classes that share similar properties, such as:
  - Optimal Value: $|V^*(c) - V^*(c')| \leq \varepsilon$;
  - Optimal Action: $\pi^*(c) = \pi^*(c')$;
  - Optimal State-Action Value: $||Q^*(c, \cdot) - Q^*(c', \cdot)|| \leq \varepsilon; \ldots$
- Do not do this too often, to allow exploration.

Potential Benefits

1. Discard useless features $\Rightarrow$ enhance generalization;
2. More samples per class $\Rightarrow$ better policies;
3. Re-initialize search for features $\Rightarrow$ escape from local optima.

Well, but...

- Original RLVC: Visual classes are conjunctions of features.
- Modified RLVC: Visual classes are the result of a sequence of:
  1. conjunctions (splitting), and
  2. disjunctions (aggregation).
- So, we must express arbitrary Boolean functions.
- Decision trees are not expressive enough!

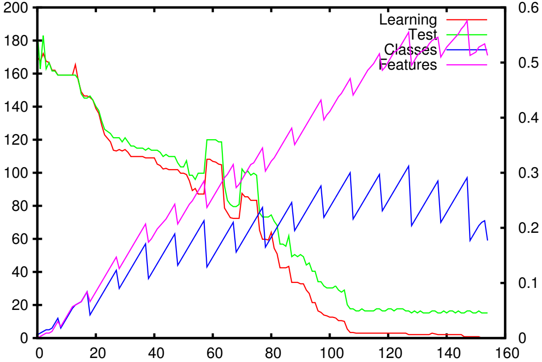
Binary Decision Diagrams (BDDs) [Bryant, 1992]

- Tree-based representation for encoding Boolean functions.
- Widely used in Computer-Aided Verification.
- Canonical if the order of the variables (visual features) is fixed.
- Reordering variables $\Rightarrow$ Discarding useless variables.
- Optimal reordering is NP-Complete, but good heuristics exist.

Summary

Replace the decision tree by a set of BDDs such that:

- Each BDD describes one visual class;
- The BDDs define a partition of the visual space.

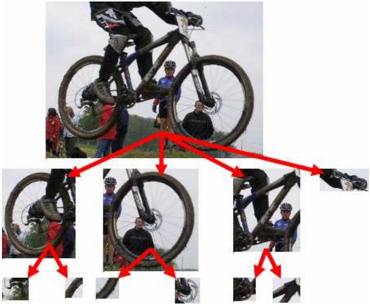


Influence of Compacting

- Visual classes: 281 → 59 ≈ 44 (number of states).
- Distinct SIFT features: 264 → 171.
- Policy error: 0.1% → 0% on LS, 8% → 4.5% on TS.

(ii) Hierarchy of Visual Features

- RLVC depends on the discriminative power of the features.
- Not enough power $\Rightarrow$ Sub-optimal image-to-action mapping.
- The physical structure imposes strong constraints on the spatial relationships between the visual features.



- Idea: Generate spatial combinations $\Rightarrow$ More discriminant.

[Jodogne, Scalzo & Piater, 2005]

(iii) Taking Advantage of Supervised Learning

Fitted Q Iteration

- Function Approximation is a successful technique for RL in continuous spaces (e.g., [Ernst et al., 2005]).
- Turn RL into a sequence of supervised regression problems.

Adaptation to Visual Tasks

Immediate:

- Use the same algorithms,
- Use supervised regression algorithms for discrete input spaces (notably Extra-Trees [Geurts et al., 2005]),
- Use the visual feature space as the input space.

[Work in progress. . .]

Conclusions

- Closed-loop learning of image-to-action mappings.
- Interactive, task-driven.
- Biological correlates.

Long-term Goal

Build a robotic system able to solve visual, reactive tasks.

Research Directions

- Continuous action spaces (discretization? Fitted Q?).
- Highly parallelizable $\Rightarrow$ Grid-ification.
- Structure a short-term memory [McCallum, 1995].
- Learning paradigms other than RL?

Thank you for your attention!