Autonomous Navigation using Active Perception

Report LiTH-ISY-R-2395

Per-Erik Forssén

Computer Vision Laboratory, Department of Electrical Engineering

Linköping University, SE-581 83 Linköping, Sweden

September 26, 2001

Abstract

This report starts with an introduction to the concepts *active perception*, *reactive systems*, and *state dependency*, and to fundamental aspects of perception such as the *perceptual aliasing* problem, and the number-of-percepts vs. number-of-states trade-off. We then introduce *finite state machines*, and extend them to accommodate active perception. Finally we demonstrate a state-transition mechanism that is applicable to autonomous navigation.

1 Introduction

Traditionally much effort in machine vision has been devoted to methods for finding detailed reconstructions of the external world. There is however no need for a system that interacts with the external world to perform such a reconstruction, since the world is continually "out there". This realisation is neatly summarised by the metaphor of "the world as an outside memory"[15]. By directing your sensors at something in the external world, instead of examining an internal model, you will not only save computational resources, you will probably get more accurate and up-to-date information as well. This observation is what lead to the emergence of the *active vision* [1, 3, 4] paradigm in computer vision.

Another recent realisation in system design is that surprisingly complex behaviours can be obtained without need for a central control mechanism. Complex behaviour can often be the reflection of a complicated environment on a few simple behaviours [2]. This realisation is the core of the behaviour oriented robotics field, initiated with the introduction of the *subsumption architecture* [5]. Subsumption is an early example of departure from the sense–plan–act paradigm, replacing it with a tighter coupling between sensors and actuators.

When actively extracting information from the real world, it is of crucial importance that we can put the information we extract into context. As we shall see, this can be accomplished by adding an *internal state* to the system.

1.1 Active Vision

Instead of generation of reconstructions of the real world, the goal in active vision is generation of actions.[1] The most common purpose of such actions will be to find out things about the real world, i.e. actively controlling the sensors.

One great advantage with active perception is that an actively obtained percept will give you more information than the corresponding passive percept. The reason for this is that you know that your action is what generated the percept. This observation has lead to suggestions of joint representations for percepts and actions, under names such as *sensorimotor contingencies* or *percept–action primitives*.

A percept–action representation is in some sense a minimal world model. However, only the aspects of the world that influence the actions of the system are encoded. Instead of being able to tell what the world looks like at a specific view, the model should aid the system at getting the desired view from the real world.

1.2 Dynamics

Active vision allows for use of a simple sensory apparatus with a much reduced dimensionality of the stimulus stream. For instance, instead of using a high resolution camera, we could use a low-resolution web camera, but move it around.

In this way, we are able to distribute the computations over time, and thus reduce the computational load. This distribution over time allows us to let preliminary results guide the gathering of new information.

If the sensory apparatus is simple, there will by necessity be several similar sensations that mean different things. The interpretation of stimulus will thus have to be highly dependent on knowledge of what the system expects to encounter. One way to incorporate such knowledge is to make the system *state dependent*.

1.3 Perceptual aliasing and states

Perceptual aliasing is the situation when identical stimuli should lead to different actions. Which action to take depends on some aspect of the world which is not apparent from the stimulus. There are two main ways to solve this problem.

- The first is to add more sensors to the system such that the states can be told apart. For simple situations this might be a quick fix, but it cannot possibly solve *all* problems without eventually sensing *all* aspects of the world.
- Another way to get rid of the perceptual aliasing problem is to add an internal state to the system. Which state the system is currently in can now be used to tell the sensations apart. In other words: **the purpose of a system state is to resolve perceptual aliasing.**

We can view these two approaches as movements along a curve (see figure 1). As we move away from the extreme case of an infinite amount of sensors where each percept is unique, we will have to successively increase the number of system states.

[1] This of course also applies to the more general concept of *active perception*.

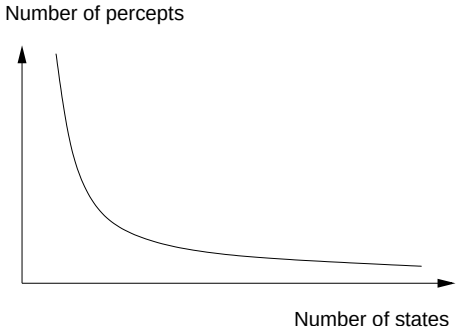

Figure 1: Percept–State curve.

The augmentation of the set of system states while reducing the complexity of the percepts is a promising way to improve response times of a vision system. In the following sections we will demonstrate how a system with very limited amount of percepts, and a very large number of system states can be designed.

2 Finite State Machines and Subsumption

A very simple system with states is a *finite state machine*, FSM (see figure 2). In a FSM, the state is determined as a mapping from the previous state and the current stimulus. The chosen action is a function of the current state. FSMs have binary inputs and states, and thus the state transitions are discrete. In order to minimise the size of the mappings the system state state is represented using *compact coding*[2].

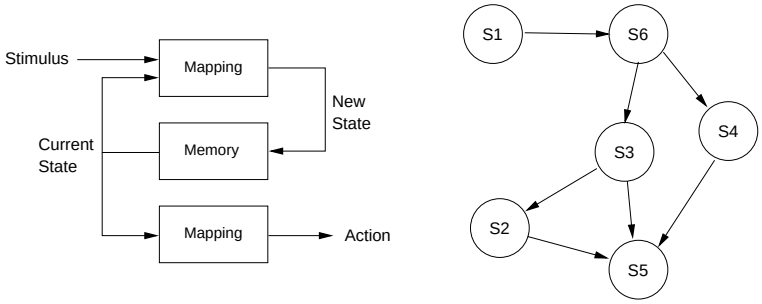

Figure 2: FSM circuit and a state diagram.
Left: FSM circuit (Moore type).
Right: State diagram: Nodes represent system states, links are state changes.

The *subsumption architecture* [5] in behaviour oriented robotics makes heavy use of FSMs. In the subsumption architecture, the robotic system is designed as a set of behavioural modules, where each module is an *augmented finite state machine* (AFSM). In an AFSM, each action is a function of the stimulus, the state, and a set of instance variables.[3] This is a way to replace the prevailing sense–plan–act paradigm, with a tighter

[2]State is typically represented as a binary number.
[3]See section 2.1 for the equivalence between continuous FSMs and AFSMs.

coupling between sensors and actuators (see figure 3).

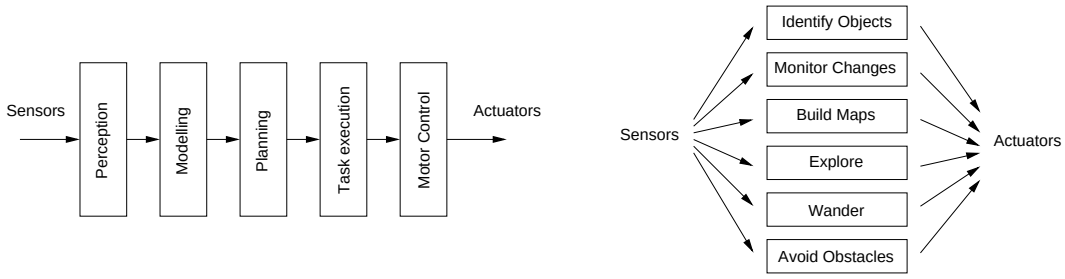

Figure 3: Sense–Plan–Act vs. Subsumption.
Left: Sense–Plan–Act type system.
Right: Subsumption architecture with behavioural modules.

Subsumption, the technique from which the architecture derives its name, is a robust way to implement layered control. More complex behaviours are allowed to subsume lower level behaviours (by suppressing them). This allows for a robust incremental design procedure, where new behaviours may be added without having to modify the existing ones.

The subsumption architecture is also an early example of a system without central control.

2.1 Notes on learning

It is of crucial concern that a vision architecture is well suited to associative learning [7]. There are some things that should be noted about the subsumption architecture in this context.

- The use of binary signalling in FSMs forces each state to be either on or off. We will instead use graded signals between 0 and 1, in the *channel representation* [8, 14]. This allows for several simultaneously active states, that are active to different degrees. The system behaviour will be an interpolation of the behaviours defined by the active states.

- The main difference between state and instance variables in AFSMs is that the state is discrete, while the instance variables may be continuous. If we allow for continuous state transitions as well, the instance variables can be seen as part of the state, and we arrive at a plain FSM with continuous states.

- Treating both state and instance variables as the system state is a more convenient model if we want the system to learn its behaviour, since both state and instance variables will be on the same side of the mapping to learn anyway.

- The compact coding of states in FSMs is undesirable if we want the system to learn a behaviour by stimulus–response association. Instead, the state should be represented using a *local* or *semi-local* coding scheme, since this will give us a linear

optimisation problem [6]. A local coding will also give us a *local metric* in state space (states with a large probability to be simultaneously active are close in state space).

- A semi-local coding allows for several concurrent hypotheses, provided that they are far enough apart (see discussion on *metamerism* in [6], pp 25). This is a strength, but also requires that the system should be able to reduce ambiguity through active perception. For this to be possible, each application of the state-transition mapping should produce a *narrowing* on the system state, instead of the *transition* that the mapping in FSMs generate.
- The use of a joint percept–action representation is not explicit in the subsumption architecture.

These observations lead us to the system design presented in the following section.

3 A state dependent active perception system

First we make the observation that a state dependent reactive system can conceptually be split into two subsystems: a *state transition* system, and a *motor program* system. We will also have an initial preprocessing subsystem that converts the stimulus into percepts through sparse coding (see figure 4). Basically this design consists of the FSM depicted in figure 2, with action as added input.

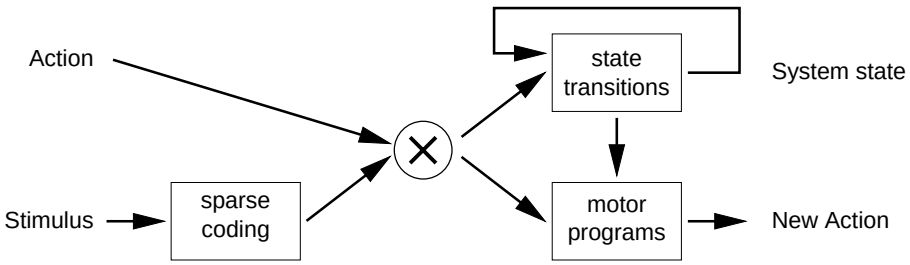

Figure 4: Information processing steps in active perception.

The idea of a state transition system is borrowed from the *cognitive maps* that biological systems seem to use for navigation. Our design is however not limited to navigation, it applies to dynamic learning agents in general. The term cognitive map usually refers to a map of *place cells* that fire when the system is at a particular location or place. A common way to cluster continuously varying sensory input autonomously is the *self organising map* (SOM) [12]. The original implementation of SOMs makes use of an *a priori* metric to guide the self organisation. An alternative that seems to be more useful, as well as more biologically plausible[4] is to instead let the *temporal correlation* of the learnt place cells define the metric driving the self organisation [10].

[4]SOM is not a biologically plausible model since there appears to be no correspondence between place-cell metric and place-cell locations in the mammalian cortex.

This has interesting parallels to the idea of using the system response as an organising mechanism for learning of behaviour [9]. Due to the physical constraints of the real world, system responses have to be continuous, both spatially and temporally.

Motor programs are closely related to the *sensorimotor primitives* of Morrow and Khosla [13]. They propose that a layer of sensorimotor primitives (SMP) as the lowest level of reactive behaviour. The next abstraction level, which they call *skills*, are FSMs which activate such sensorimotor primitives. This design reduces the dimensionality of connections between task space, and sensor and motor spaces, without sacrificing context sensitivity. This is accomplished by isolating contextual details to the SMPs. Importantly, each state in a skill does not necessarily point to a single SMP, instead several may be simultaneously activated.

For the time being we will set aside the issue of practical implementation of autonomous place cell learning, and concentrate on the desired behaviour of the state transition system.

4 A state transition mechanism for navigation

To demonstrate the principle of a state transition mechanism we will apply it on the problem shown in figure 5. The arrow in the figure symbolises an autonomous agent that is supposed to navigate through the labyrinth.

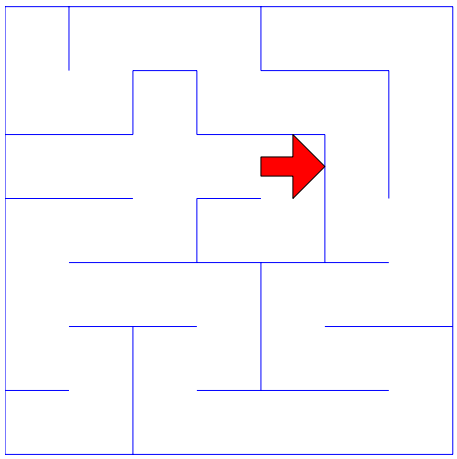

Figure 5: Illustration of the labyrinth navigation problem.

The stimulus constitutes a three element binary vector, which tells whether there are walls to the left, in front, or to the right of the agent. For the situation in the figure, this vector will look like this:

$$\boldsymbol{s} = \begin{pmatrix} 1 & 1 & 0 \end{pmatrix}^t$$

This vector is expanded by the sparse coding module to incorporate a flag that signals if the vector is all zero. The percept will thus look like this:

$$\boldsymbol{p} = \begin{pmatrix} 1 & 1 & 0 & 0 \end{pmatrix}^t$$

The only reason for this extra signal is that we want to train an associative network [8] to perform the state transitions, and without any input signal there is nothing to associate the response with.

The system has three possible actions `TURN LEFT`, `TURN RIGHT` and `MOVE FORWARD`. These are also represented as a three element binary vector, with only one non-zero element at a time. Thus `TURN RIGHT` would be represented like this:

$$\boldsymbol{a} = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}^t$$

Each action will either turn the agent 90° clockwise or anti clockwise, or move it forward to the next grid location (unless there is a wall in the way).

As noted in section 1.3, the purpose of the system state is to resolve perceptual aliasing. For the current problem this means that the system state has to describe both agent location and absolute orientation. This gives us the number of states as:

$$N_s = \text{rows} \times \text{cols} \times \text{orientations} \tag{1}$$

For the labyrinth in figure 5 this means $7 \times 7 \times 4 = 196$ different states.

4.1 Training of state transition mapping

The state transition mapping we want to make can formally be written as:

$$\boldsymbol{s}_n \otimes \boldsymbol{a} \otimes \boldsymbol{p} \mapsto \boldsymbol{s}_{n+1} \tag{2}$$

Where $\otimes$ is the *Kronecker product*, which generates a vector containing all product pairs of the elements in the involved vectors.

In order to find the state transition mapping, we supply the system with examples of each possible state transition in the form expressed in equation 2. This means that the number of training examples is given by the formula:

$$N_e = N_s \times N_a \tag{3}$$

Where N_s is the number of states and N_a is the number of actions. For the current situation this means $196 \times 3 = 588$ examples.

We now group the inputs into a vector $\boldsymbol{f} = \boldsymbol{s}_n \otimes \boldsymbol{a} \otimes \boldsymbol{p}$, and denote the output as $\boldsymbol{u} = \boldsymbol{s}_{n+1}$. The optimisation problem now looks like this:

$$\arg \min_{c_{ij}>0} \|\boldsymbol{u} - \boldsymbol{C}\boldsymbol{f}\|$$

See [11] for details of how the solution is found.

4.2 Performance

To evaluate the narrowing properties of the found mapping, we start with a system state consisting of all ones, and start to move the agent around in a semi random manner. Since the mapping of equation 2 is purely linear we also normalise the system state after each application of the mapping by division with the strongest response. One example of the narrowing behaviour is shown in figure 6.

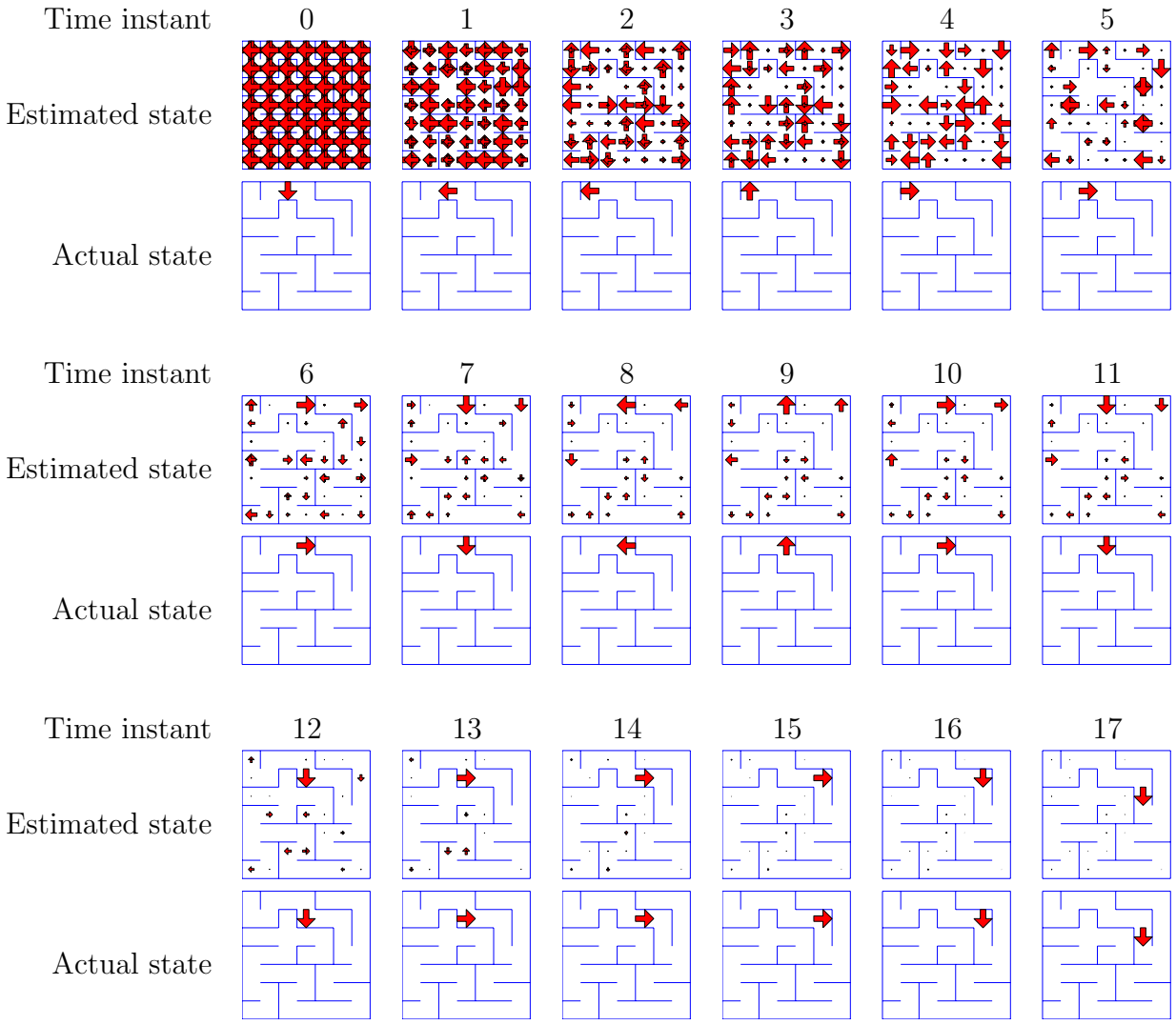

Figure 6: Illustration of the narrowing process.

The exploratory behaviour of the agent (the motor program box in figure 4) is at present defined by a *Markow process* with conditional probabilities of the actions dependent on the previous action and the current percept. Such a behaviour could be learnt by reinforcement by assigning credits depending on a measure of the narrowing performance.

5 Discussion

The purpose of this report is to outline which system properties are desirable in active perception, and not to present a useful application. The state transition presented in section 4 is intended to be useful in more complex active perception systems than the labyrinth example. Focusing on a system with very limited capacity with regard perception and action has however been fruitful for finding which general properties are desirable.

Future experiments will involve tests of the proposed architecture on navigation in aerial images. Actions will shift gaze between different landmarks, and the system state will tell which landmark the system is looking at.

Acknowledgements

The work presented in this report was supported by WITAS, the Wallenberg laboratory on Information Technology and Autonomous Systems, which is gratefully acknowledged.

References

[1] Y. Aloimonos, I. Weiss, and A. Bandopadhay. Active vision. *Int. Journal of Computer Vision*, 1(3):333–356, 1988.

[2] R. C. Arkin. *The Handbook of Brain Theory and Neural Networks*, chapter Reactive Robotic Systems. MIT Press, 1995. M. A. Arbib, Ed.

[3] R. Bajcsy. Active perception. *Proceedings of the IEEE*, 76(8):996–1005, August 1988.

[4] D. H. Ballard. Animate vision. In *Proc. Int. Joint Conf. on Artificial Intelligence*, pages 1635–1641, 1989.

[5] R. Brooks. A robust layered control system for a mobile robot. *IEEE Trans. Robot. Automat.*, (2):14–23, 1986.

[6] P.-E. Forssén. Sparse Representations for Medium Level Vision. Lic. Thesis LiU-Tek-Lic-2001:06, Dept. EE, Linköping University, SE-581 83 Linköping, Sweden, February 2001. Thesis No. 869, ISBN 91-7219-951-2.

[7] G. H. Granlund. The complexity of vision. *Signal Processing*, 74(1):101–126, April 1999. Invited paper.

[8] G. H. Granlund. An Associative Perception-Action Structure Using a Localized Space Variant Information Representation. In *Proceedings of Algebraic Frames for the Perception-Action Cycle (AFPAC)*, Kiel, Germany, September 2000.

[9] G. Granlund. Does Vision Inevitably Have to be Active? In *Proceedings of SCIA99, the 11th Scandinavian Conference on Image Analysis*, Kangerlussuaq, Greenland, June 7–11 1999. Also as Technical Report LiTH-ISY-R-2247.

[10] V. V. Hafner. Learning places in newly explored environments. In *Proceedings Supplement Book, Publication of the International Society for Adaptive Behavior*. SAB.

[11] B. Johansson. On Sparse Associative Networks: A Least Squares Formulation. Report LiTH-ISY-R-2368, Dept. EE, Linköping University, SE-581 83 Linköping, Sweden, June 2001.

[12] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[13] J. D. Morrow and P. K. Khosla. Sensorimotor primitives for robotic assembly skills. In *Proceedings of the 1995 IEEE Conference on Robotics and Automation*, Nagoya, Japan, May 1995.

[14] K. Nordberg, G. Granlund, and H. Knutsson. Representation and Learning of Invariance. In *Proceedings of IEEE International Conference on Image Processing*, Austin, Texas, November 1994. IEEE.

[15] J. K. O'Regan. Solving the 'real' mysteries of visual perception: The world as an outside memory. *Canadian Journal of Psychology*, 46:461–488, 1992.