Biologically-inspired visual-motor coordination model in a navigation problem

Jacek Jelonek Maciej Komosinski

Poznan University of Technology, Institute of Computing Science
Piotrowo 2, 60-965 Poznan, Poland
maciej.komosinski@cs.put.poznan.pl

Abstract. This work presents a biologically-inspired coordination model which associates motor actions with visual stimuli. The model is introduced and explained, and navigation experiments are reported that verify the implemented visual-motor system. Experiments demonstrate that the system can be trained to solve navigation problems consisting in moving around a 3D object to reach a specific location based on the visual information only. The model is flexible, as it is composed of an adjustable number of modules. It is also interpretable, i.e. it is possible to estimate the influence of visual features on the motor action.

1 Introduction

In the real world, creatures face a complex, changing environment and need to handle large amounts of information to survive and reproduce. Robots produced by humans should possess analogous qualities if we want them to autonomously make decisions and perform successfully in natural environments. However, this is not yet accomplished; there is still a huge difference between efficiency of performance between creatures and modern robots. If one could develop robots that are as robust, flexible and adaptive as living organisms, a lot of time and money would be saved that is spent on designing and developing robots that are highly specialized in performing a specific task in specific conditions.

One of the factors that play an important role in the success of living organisms is the way they acquire information from the environment. Their senses are interfaces between neural systems and the outer world. Living organisms exhibit a vast number of sensor types, including olfactory, tactile, auditory, visual, electric and magnetic ones. In this work we focus on visual sensing as the one that provides a lot of information about the environment and is therefore popular in natural systems and often used in artificial designs.

In the area of machine vision, problems that are considered are usually related to object recognition and classification. This work adds the aspect of active exploration of the environment based on information that is perceived. The information considered here is visual and much more complex than the information perceived by light-following robots that mimic simple organisms like *Paramecium*.

This paper focuses on a visual-motor model that facilitates stimulus–reaction performance, as it is the basic schema in functioning of living organisms. The stimulus is visual, and motor reaction is movement of an agent. The purpose of building this biologically-in-

The final version of this paper appeared in *Lecture Notes in Computer Science* 4253:341-348, 2006. http://dx.doi.org/10.1007/11893011_44

spired model is twofold. First, it helps in understanding cognitive processes in living organisms. Second, implementations of such models can cope with the complexity of real-world environments because these models are inspired by solutions that proved to be successful in nature.

The experiments with visual-motor model are performed using simulated, artificial agents. The software environment is the Framsticks simulator [1] equipped with a new *vector eye* sensor. We consider navigation and target approaching tasks [5] with an agent moving along a circular path around some scene, observing a three-dimensional object positioned in the center. The agent decides whether it wants to move left or right, and adjusts speed of its movement.

Analogously to natural environments, some locations around the object are advantageous ("life zones" that living organisms try to reach) while others are adverse (avoided "death zones"). The goal for the agent is to reach some optimal location (using motor actions) based only on the visual information that it perceives looking at the object. The visual-motor system inside the agent needs to be trained to accomplish this task.

The next section describes in detail the architecture of the visual-motor system, presents biological inspirations for the model, and introduces its three components: vector eye, visual cortex, and the motor area. Section 3 reports experiments that were performed, and Section 4 summarizes this article and points out directions of future research.

2 Architecture of the visual-motor system

The architecture of the proposed visual-motor system consists of three components –*vector eye*, *visual cortex* and *motor area*. Vector eye captures edges, basic visual elements of a scene, as observed by an agent. Each edge is characterized by four attributes – length, angle and coordinates of its center. The attributes of all edges (vector data) are transformed and aggregated by the visual cortex and fed to the *motor area* module. The motor area controls agent's movements in the virtual environment. The data flow is illustrated in Fig. 1.

2.1 Vector eye

Vector eye is a high-level sensor that provides a list of edges in a scene that are visible from some location in space. This information is accurate, i.e. it has no noise or imperfections which would exist if these edges were detected in a raster picture.

The sensor is implemented in Framsticks, a simulation environment that models three-dimensional bodies of agents and their neural control systems [1]. Framsticks allows users to perform predefined experiments, but the software also supports user-defined experiment definitions, fitness functions, and neuron types. In this work, only basic Framsticks functionality was used. Features like evolutionary optimization, neural control and embodiment can be utilized in future research.

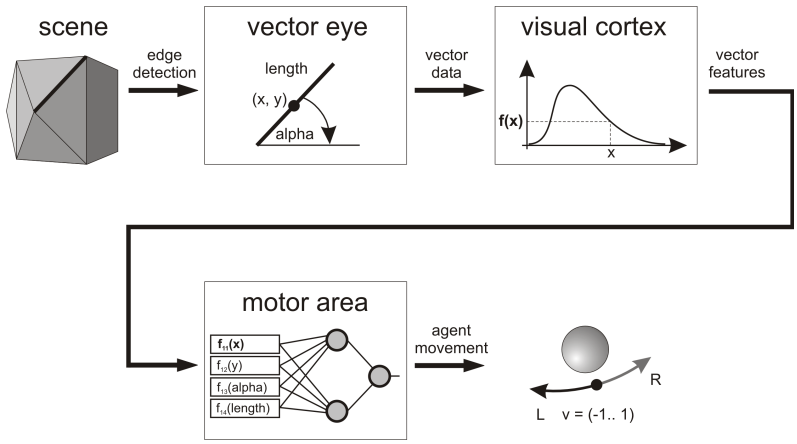

Fig. 1. Data flow in the model of visual-motor system

For the purposes of this work, the agent that travels around the object is considered as a point equipped with a single eye sensor that observes the centre of the scene and perceives a singe, three-dimensional shape, as shown in Fig. 2 on the left.

Many simple sensors that are commonly used in robotics (touch, proximity, 3D orientation) provide single-valued outputs, and therefore do not need special post-processing of this information in order to make it useful. Vector eye, on the other hand, is a complex sensor that provides a variable amount of information depending on what shape is perceived and what is the relative position and orientation of the sensor with respect to the shape (see Fig. 2, right).

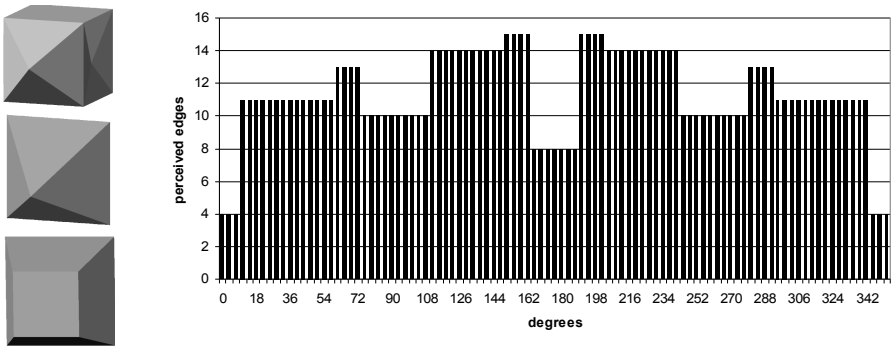

Fig. 2. Left: three-dimensional shapes used for experiments. Right: histogram of the number of edges perceived when observing a sample object from different angles

2.2 Visual cortex

The proposed model of visual cortex is inspired by its biological counterpart. It has been shown that the primary visual cortex consists of cells that are selectively responsive to different features of the visual stimulus [3, 4]. The cells are called feature detectors because they analyze the visual image to find specific features (such as a bar or line of a specific

orientation, length, etc.). For example, there are *simple cells* for which stimulus that maximally excites the cell is a line at a specific angle of orientation across the retina. Deviations from that preferred angle excite the cell less and less, until a line perpendicular to its receptive field has no effect. *Complex cells* show orientation specificity like simple cells, but additionally, they manifest highest sensitivity to stimuli that are lines moving in a specific direction across the visual field. Finally, *hypercomplex cells* are most sensitive to lines of specific length and specific angle of orientation that move in a specific direction.

The idea of feature detectors is employed in the proposed model of visual cortex. Data coming from the vector eye sensor contain a set of geometrical attributes. Four geometrical attributes are considered for each edge – angle, length and location (expressed by two coordinates, X and Y). Values of these attributes are transformed by parameterized Gaussian functions (see Fig. 3). Each function is defined by the following parameters: *modal* (preferred) value of associated attribute that excites the neuron most, *extreme* value of excitation (positive or negative), and *left* and *right standard deviation* that shape the function. The final excitation is a sum of excitations invoked by all edges provided by the vector eye. This solves the problem of aggregating variable numbers of attributes (depending on a number of edges) coming from the eye sensor.

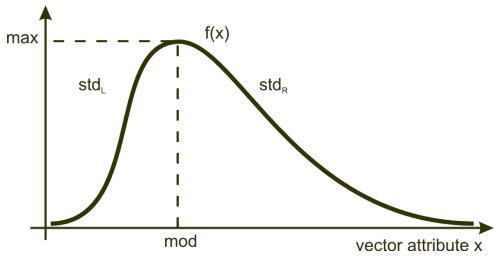

Fig. 3. Parameterized Gaussian function

2.3 Motor area

We started from a simple architecture of the motor component, employing the OWA (ordered weighted averaging) method proposed by Yager [6] to aggregate values of features from the visual cortex. However, this approach was not sufficient as such a component could not be trained well. Therefore, more OWA modules were added but it was still difficult to minimize error sufficiently. Finally, neural networks were introduced instead of the OWA operator, which let the system learn the navigation task successfully.

The motor area component is implemented as a set of motor modules. Each module consists of a two-layer feed-forward neural network. First layer neurons are fully connected to outputs of the visual cortex (parameterized feature detectors included in Fig. 4). Each neuron is defined by 3+*w* parameters, where *w* is the number of inputs (weights) and the other three parameters characterize the shape of the sigmoid transfer function. Motor output is computed as the total activation of all motor modules. A behavioral adaptation of an agent to virtual environment consists in adjusting all the parameters, and can be achieved by various optimization and/or learning techniques.

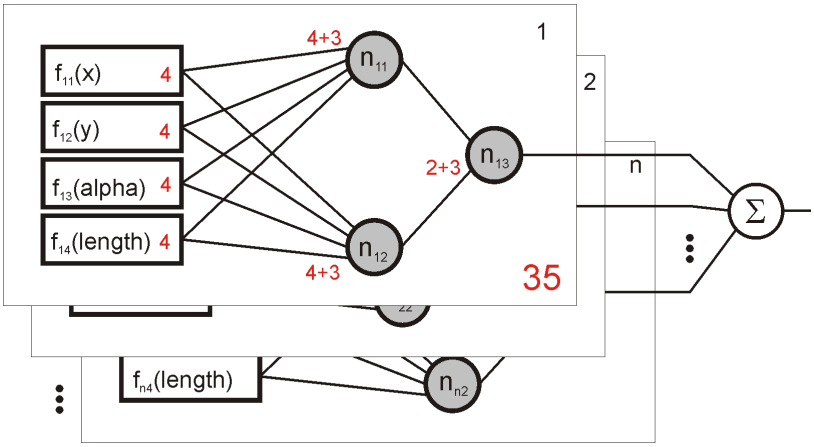

Fig. 4. The topology of the motor component. Number of parameters shown for each part of a module (total of 35 parameters per module)

3 Experiments: training and analyzing the visual-motor system

To train the visual-motor system and adjust its parameters, various training methods and regimes could be used, including evolutionary learning. In this work we employ direct, supervised learning, which is common in nature, where parents teach their children particular behaviors by rewarding or punishing them for specific actions.

A greedy gradient optimization algorithm [2] was selected as the training method, because it is relatively simple and quick. In the beginning, all system parameters are initialized randomly, and a small delta value is associated with each parameter. Then, for each parameter, its delta value is added, and if the new set of parameters is better than the old one, the new parameter value is accepted and the corresponding delta value is increased. Otherwise, if this change was not beneficial to the whole system, the old value for the parameter is retained and the corresponding delta value is negated and decreased. The process is repeated as long as a change in any parameter causes improvement of the whole system (i.e. the global error is decreased).

As we expect the agent to be able to navigate to some specified position around the object and stop there, its target speed should have positive values for one half and negative values for the other half of the circular path. The speed should be zero for the specified position where the agent should stop. There are many speed functions that satisfy these requirements, which causes problems for some learning algorithms, as there is no clear and continuous information on the expected direction of changes for system parameters.

The obvious error measure is the number of mistakes of the sign of speed value for all positions around the object. For example, if we assume positive speed as “move right” and negative speed as “move left”, the total error is the number of locations where the agent moves in a wrong direction, no matter how fast. Although such error formula is good for evaluating agent behavior, it is not very helpful during gradient optimization. It only yields a limited number of discrete values which does not provide sufficient (continuous) information to minimize the error, especially that there are many feasible speed functions. For this reason we decided to use a specific target speed function that the system is trained. It is a

sinus function, and the zero angle is adjusted to match the specified stop position, where the target speed is zero.

The first experiment described in this section concerns selecting the appropriate error function to be minimized, and the second experiment looks for the optimal complexity of the visual-motor system. Other experiments are mentioned in the last section of this article.

3.1 Error functions for the learning process

For the learning process, some error formula (or fitness function) is needed that will evaluate the visual-motor system configuration by observing the behavior of an agent. As agent's behavior is deterministic and fully determined by the visual sensor input, the error function can take into account a finite set of positions (angles) around the object. Let us introduce some variables:

x_i – target (optimal) speed for image viewed from angle i, i.e. $x_i = sin(i)$
y_i – speed that is generated by the visual-motor system (Fig. 4) for angle i
e_i – difference between target and actual speed for angle i; $e_i = y_i - x_i$
e – total error, $e = \sum_i |e_i|$

In the experiments, the full circle path around the central object was sampled with 100 angles, so $i \in \{0°, 3.6°, 7.2°, 10.8°, .., 356.4°\}$. The most obvious error function is e, the sum of individual errors for each angle. However, minimizing this error function resulted in a ragged speed characteristics (see the white line in Fig. 5). Therefore, another component of the error function was introduced, the standard deviation σ of individual errors e_i. Table 1 summarizes results obtained for minimizing three error functions, and Fig. 5 shows speed characteristics.

The results of this experiment show that minimizing both e and σ is advantageous, as it reduces raggedness of speed that is output by the visual-motor system (compare white and bold black lines in Fig. 5). Moreover, it helped the hill-climbing learning algorithm to minimize error e: the value of this error is actually smaller when minimizing $e+\sigma$ than when only minimizing e (see Table 1).

Table 1. Performance obtained for various error functions

Minimized error function	Trained performance	
	e	σ
e	7.63	0.12
σ	19.93	**0.05**
$e+\sigma$	**6.17**	0.08

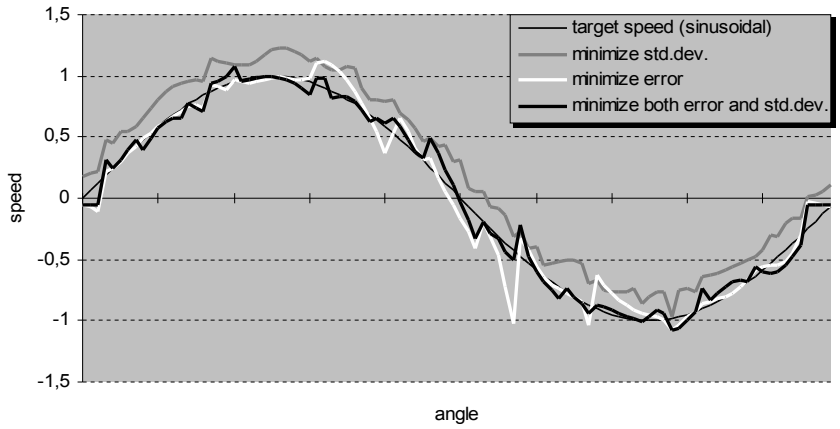

Fig. 5. Speed versus angle of observation

3.2 Adjusting the number of motor modules

The important advantage of the proposed model is its scalability: the number of motor modules can be easily adjusted to match the difficulty of the problem at hand. To investigate the influence of the number of modules (each with 35 parameters) on the training ability, we tested five visual-motor systems. For each system, 20 training experiments were performed, and the results are summarized in Fig. 6.

Although there is little improvement in the best-trained system among 20 trials, it can be clearly seen that both average errors and standard deviations are decreasing as subsequent modules are added. This means that the more modules there are in the system, the better it can cope with transforming visual stimuli into appropriate motor actions, and the stability of results increases.

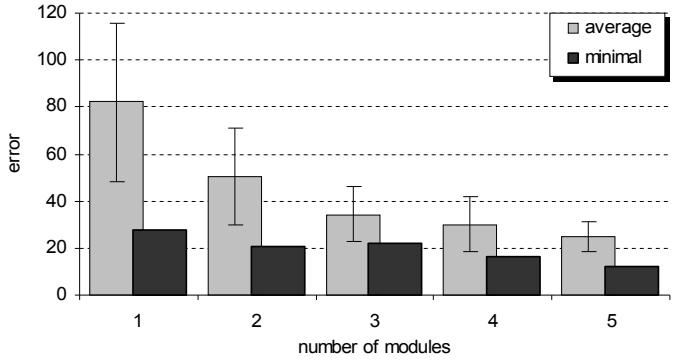

Fig. 6. Error values for visual-motor systems of increasing complexity

4 Summary and future work

This work presented a biologically-inspired visual-motor coordination model. The model has been verified in a number of navigation tasks and perceived shapes, and proved to be flexible and appropriate for such tasks.

Other interesting experiments with this model were performed as well. It was verified that minor changes in the shape of the 3D object do not deteriorate agent behavior. Changing the size of the object, and changing the distance of the agent from the object did not increase the error much, so the system proved to be robust to minor changes in the environment. The specific place where the agent should stop was also set in various locations around the central object, and training was always successful.

An interesting feature of the proposed model is that the system performance can be visualized and interpreted (explained). It is possible to estimate the influence of each edge on the output speed value, and to visualize it (e.g. edges that cause the agent to move right are red, and those causing the agent to move left are green). Moreover, it is possible to perform such analysis for individual sensory features (like edge angles, lengths, etc.).

Future works concern adding more degrees of freedom for the agent (i.e. moving in the 3D space, and near/far from the object), using evolutionary algorithms for more complex navigation tasks, introducing more complex shapes and real-world objects, using many eyes and visual-motor systems simultaneously, embodiment of such a system within a virtual body, and finally, perceiving realistic, raster camera images. Most of these experiments are work in progress.

Acknowledgment

This work has been supported by the State Committee for Scientific Research, from KBN research grant no. 3 T11C 050 26, and by the Foundation for Polish Science, from subsidy no. 38/2004.

References

1. Adamatzky, A., Komosinski, M. (eds): Artificial Life Models in Software, chapter 2. Springer-Verlag (2005).
2. Baldi, P.: Gradient descent learning algorithm overview: a general dynamical systems perspective. IEEE Transactions on Neural Networks 6:1 (1995) 182–195.
3. Hubel, D.H., Wiesel, T.N.: Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex. Journal of Physiology 160, London (1962) 106–154.
4. Hubel, D.H., Wiesel, T.N.: Receptive fields and functional architecture of the monkey striate cortex. Journal of Physiology 195, London (1968) 215–243.
5. Trullier, O., Wiener, S., Berthoz, A., Meyer, J.-A.: Biologically-based Artificial Navigation Systems: Review and Prospects. Progress in Neurobiology, Vol. 51 (1997) 483–544.
6. Yager, R., Rybalov, A.: Uninorm Aggregation Operators. Fuzzy Sets and Systems 80 (1996) 111–120.