In: M.H. Hamza (ed.), ***Proceedings of the 21st IASTED Conference on Applied Informatics,*** **pp. 1278-1283.**
Held February, 10-13, 2003, Insbruck, Austria

Evolving High-Dimensional, Adaptive Camera-Based Speed Sensors

Ralf Salomon
Department of Electrical Engineering and Information Technology
University of Rostock, 18051 Rostock Germany
email: ralf.salomon@etechnik.uni-rostock.de

ABSTRACT
This paper reviews some attempts that exploit a phenomenon, also known as motion parallax, to estimate the distance of closest approach of a moving object. Despite their success, the existing methods lack some desirable properties, such as reasonable scalability and online learning. To overcome these practically-relevant limitations, this paper proposes a new model that is based on Hebbian learning. Due to its scalability and online learning capabilities, this model is especially suited to mobile robots.
KEY WORDS
Intelligent Agents, Robotics, Computer Vision

1 Introduction

Since the beginning of the 90ies a new research area has emerged, for which Brooks has coined the term new artificial intelligence (new AI for short). New AI aims at understanding (natural) intelligence and its underlying mechanisms by building systems that exhibit "intelligent" behavior [1, 2, 7]. These systems are often realized as mobile robots, which are supposed to operate in dynamically changing, partially unknown environments without any human control (that is why they are attributed *autonomous*). New AI prefers a synthetic approach, i.e., understanding by building. In order to reach its research goals, new AI draws a significant amount of inspiration from natural systems. It therefore often investigates (biological) hypotheses and aims at validating them in simulation or on particularly-designed robots.

Even though the ultimate goal is to build physical robots, most attempts resort to simulation for obvious limitations of evolvable hardware available today (see, for example, the conference series *Simulation of Adaptive Behavior*), A very nice exception is the *eyebot* on which Lichtensteiger and Eggenberger have evolved simplified insect eyes [6]. Section 2 briefly explains how Lichtensteiger and Eggenberger used evolutionary algorithms to evolve the eye's morphology that allows the robot to consistently estimate the distance of closest approach. Despite its successes, previous research [6, 8] has indicated that this approach is practically limited to about 10-20 sensors, since the evaluation of a particular sensor distributions requires approximately one minute on a physical robot.

Furthermore, evolutionary algorithms, like backpropagation-type learning procedures, require a *preselected* training set for *offline* learning and thus lack any online adaptation capabilities. By way contrast, living creatures are inherently adaptive, scale very well over their visual system, and apparently do not suffer from overlearning effects (that much). Section 3 proposes a new, simple model that achieves these design goals to a large extent by using a simple Hebbian-based learning rule. Section 3 also discusses some of the model's basic properties.

Sections 4 and 5 discuss the methods and results of some representative experiments. The results show that the proposed learning method is able to train the network model such that it can consistently determine the robot's speed and thus can determine the distance of closest approach. Section 6 concludes with a brief discussion.

2 Background and Previous Research

2.1 The Eyebot

Inspired by biological evidence [3, 4, 5], Lichtensteiger and Eggenberger [6] constructed a robot, called *eyebot*, to model the eye of an insect, such as the house fly. The robot consists of a chassis, an on-board controller, and sixteen independently-controllable facet units, which are all mounted on a common vertical axis. A facet unit basically consists of the sensor, a thin tube, two cog-wheels, a motor, and a potentiometer. By means of the cog-wheels, the motor can position the facet within a range of about 200 degrees, and the potentiometer provides feedback about its actual position, i.e., its angle α_i. A thin opaque tube is used to reduce the sensor's aperture to about two degrees. These tubes are the primitive equivalent to the biological ommatidia [3, 4, 5]. It should be noted that such a low-cost construction is subject to several imprecisions and tolerances, which might be sensed as noise during operation.

2.2 Motion Parallax: A Mathematical Description

Figure 1 sketches how a phenomenon, also known as motion parallax, can be utilized to avoid obstacles. Let us assume that the compound eye is at a fixed position. If the obstacle X is moving at constant speed v, the eye views

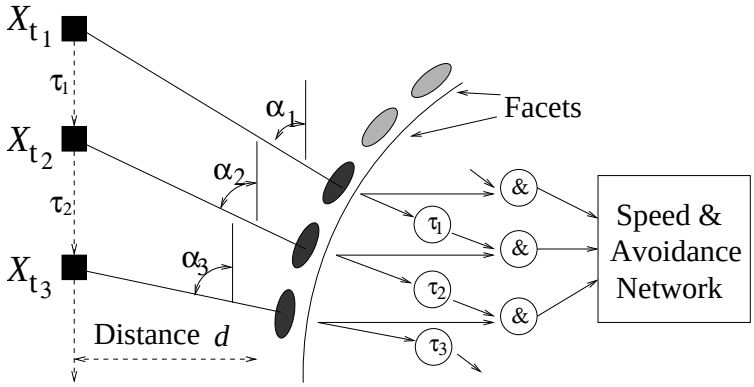

Figure 1. Due to their small aperture of about two degrees, the insect eye's facets recognize a moving object at different times $t_1, \ldots t_3$. If the time delays τ_i correspond to the object's traveling time t_{i+1}-t_i, the robot can consistently detect too small a distance of closest approach to an obstacle. For further details, see text.

the obstacle under different angles α_1, α_2, and α_3 at different time steps t_1, t_2, and t_3.

Let us assume that a facet views the obstacle under the angle α_i at time t_i. The distance y_i to the baseline (indicated by the dashed line labeled d) is given by $y_i = d/\tan\alpha_i$. At the next time step $t_{i+1} = t_i + \Delta t$, this distance reduces to $y_{i+1} = d/\tan\alpha_{i+1}$. With y_{i+1}-$y_i = v * \Delta t$ the following relation holds:

$$\frac{d}{v} = \frac{1}{\Delta t} \times \frac{\tan\alpha_i - \tan\alpha_{i+1}}{\tan\alpha_i \times \tan\alpha_{i+1}} \,. \qquad (1)$$

If the time difference Δt between two neighboring facets corresponds to the time delay τ_i, a neural network can estimate the distance d of closest approach by a simple and-operation. To this end, the robot has to assess its own speed v, what it can do by, for example, a flow sensor or its motor activation system. Depending on the assessed distance d, the robot might trigger an appropriate avoidance action.

2.3 Previous Results

Lichtensteiger and Eggenberger [6] used a simple neural network with all weights being constant and equal and applied evolutionary algorithms to evolve a suitable sensor distribution, i.e, the different angles α, such that a predefined fitness function $f = \sum_i (t_{i+1} - t_i - \tau)^2$ minimizes. Even with significantly improved algorithms [8], the results clearly indicate that this approach is practically limited to about 10-20 sensors due to the experimentation time and convergence problems.

Alternatively, the sensor distribution could be fixed and an algorithm like backpropagation could be applied to train the individual time delays τ_i. However, this approach as well as using evolutionary algorithms require the preselection of suitable training cases and offline learning, which immediately excludes any suitable online adaptation to changing conditions, such as broken sensors or a tilted camera.

3 The Model

As Fig. 2 shows, a high-resolution input device, such as a CCD camera, feeds its activation to a speed sensor module, which in turn communicates with a motor (sub-) system via reciprocal connections. The remainder of this section is devoted to some of the model's architectural details, the learning procedure, as well as some of the model's properties.

3.1 The Architecture

As has already been discussed in the background section, an object passing by the input device activates the input units i for a constant duration σ at different time steps t_i. Now, let I denote units of the input device and let S denote units of the speed sensor module. Every speed unit S_i directly connects to *exactly one* (randomly chosen) input unit I_j with the index j referencing to the speed unit's position (relative to the input device). Furthermore, all speed units also connect the neighborhood of their reference input unit. For example, speed unit S_i connects to the reference input unit I_r and also connects to $I_{r-m} \ldots I_{r+m}$ for some $m > 0$. Except for the reference connection, all signals are delayed by τ and have to pass a gate. The time delays τ may vary across all connections, and the gate has to be activated by the reference unit. That is, a speed sensor S_i receives delayed signals from the input device only during the time window defined by the reference unit I_r.

It should be noted here that all sensor units S_i are treated in the same way. For readability purposes, these neurons are arranged on a grid with columns and rows representing reference position r and speed, respectively. All speed sensors belonging to the same speed value, communicate with corresponding units in the motor system.

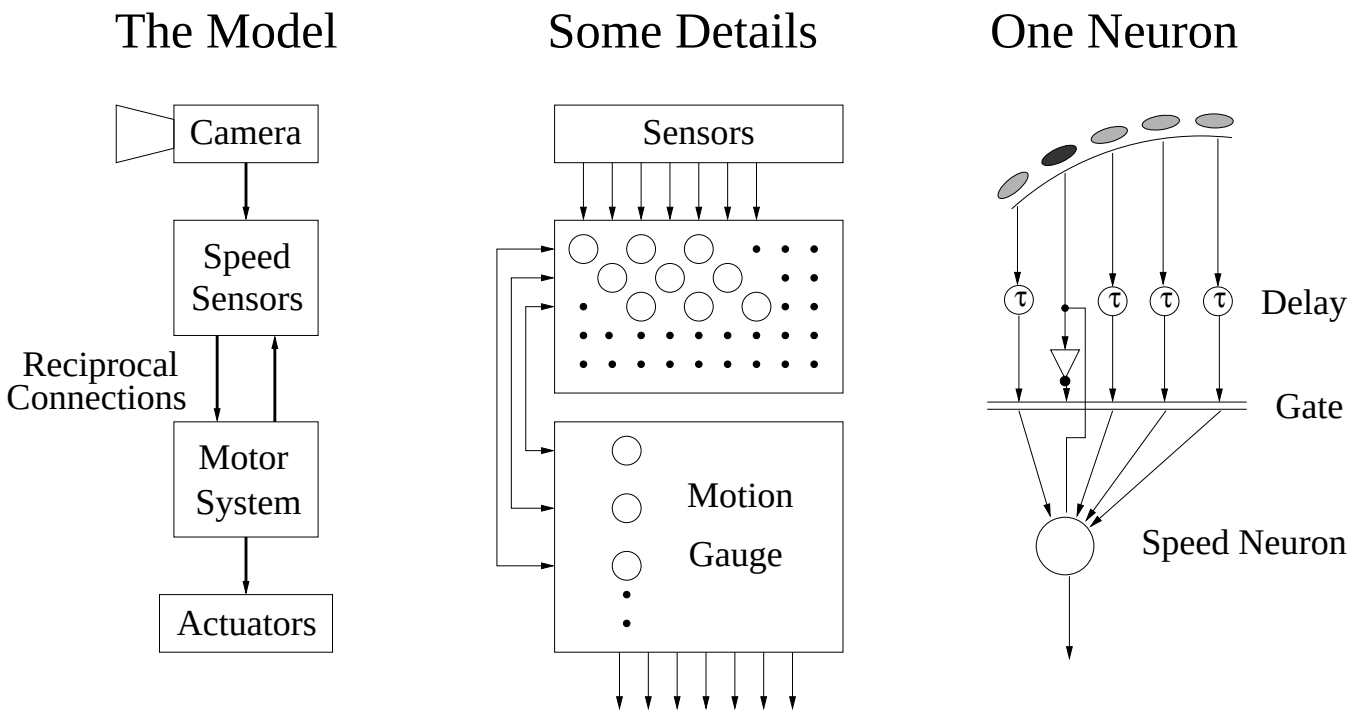

Figure 2. This figure sketches the model's architecture with increasingly more details when going from left to right. For further detail, see text.

3.2 The Learning Rule

Without loss of generality, the model makes the following assumptions: the input unit's activation I is either 0 or 1, the gate either passes the incoming activation or sets it to 0, if an input sensor views an object it sets its activation to 1 for the duration σ, the activation of the speed sensor units is bounded by $0 \leq S_i \leq 1$.

The operation of the speed sensor units is inspired by biological integrate-and-fire neurons. For each connection, S_i first calculates the overlap o_{ij} of its reference (I_r) and incoming delayed signals ($I_{j \neq r}$). More formally, the overlap is given by $o_{ij} = (1/\sigma) \int_t I_r(t) I_j(t) dt$. Due to the division by the signal duration constant σ, the overlap is normalized $0 \leq o_{ij} \leq 1$. To sharpen the edges, the overlap is modified by (with ϵ denoting an arbitrarily-chosen, small, positive constant):

$$o_{ij}^{'} \leftarrow \frac{\ln(o_{ij} + \epsilon) - \ln \epsilon}{\ln \epsilon} \,. \tag{2}$$

If activated by the reference signal I_r, the speed sensor unit S_i then applies the following, rather generic Hebbian-based learning rule (for some properties, see below):

$$w_{ij} \leftarrow w_{ij} + \eta o_{ij}^{'} - (\eta/\alpha) w_{ij} \,, \tag{3}$$

with w_{ij} denoting the connection (weight) connecting input unit I_j and speed sensor unit S_i, η denoting a learning constant, and α denoting a unlearning constant.

3.3 Model Properties and Behaviors

The learning rule, as defined by eq. (3) has the following properties: due to the unlearning constant α, the connection weight is bounded by $w_{ij} < \alpha o_{ij}^{'}$, and since $o_{ij}^{'} \leq 1$, the relation $w_{ij} < \alpha$ holds. Due to this bounding, the repetitive application of the learning rule eq. (3) lets the connections approach this bound, i.e., $\lim_{t \to \infty} w_{ij} = o_{ij}^{'}$. In other words, the network's connections assume values proportional to their incoming overlap values during the time window defined by the reference signal I_r. Without loss of generality, the unlearning constant can be set to α=1.

The learning constant η in eq. (3) determines the slope with which the final values are approached; it does not influence the final values. With smaller η values, the learning procedure consumes more time to saturate but is less sensitive to changing environmental conditions. Large η values have the opposite effect.

Like the high-resolution input device, the speed sensor module features very many units, to which learning applies as described above. The speed sensor module arranges all units on a grid with the x and y axes representing position r and speed, respectively. All units in one row are responsible for the same speed and are collectively coupled with corresponding units in the motor system. Within one row, the speed sensor units connect to different reference input units.

In the very beginning, all connections are initialized to some, randomly-chosen values. When the robot is moving with a particular speed v, the operating motor system

activates the appropriate row of the speed sensors. When these active speed sensor units receive their reference signal (what they do at different time steps), they update their connections w_{ij} by applying learning rule (3). During the initial leaning (exploration) phase, the robot repetitively moves with different speeds in order to train all speed sensor units.

After the initial learning phase, the speed sensor units may also by activated by the following activation rule:

$$S_i = \frac{o_{ij}^{'} w_{ij}}{\sum_j w_{ij}}\,, \tag{4}$$

with $o_{ij}^{'}$ denoting the modified overlap with the reference signal as described above. As cn be seen from eq. (4), the speed sensor unit integrates all incoming delayed signals during the time window given by the reference signal I_r. This integration is weighted by the w_{ij}'s and then normalized. This normalization has the effect that the parameter α has no effect and that only slightly varies with changing configurations (i.e., signal duration σ and number of connected inputs).

The main idea behind this model is as follows: Since all speed sensor units are treated in the same way and connected only to some local region of the input device, the model allows for simple scaling over very many input units of high-resolution sensor devices.

4 Methods

Unless otherwise stated, all experiments have used the following parameter settings: signal duration σ=2, learning rate η=0.01, unlearning rate α=1, overlap-modification parameter ϵ=0.001 (see, eq. (2)). During the initial training phase, the speed was set to values, such that the object travels about one sensor per time unit.

5 Results

This section presents some results, obtained with a computer simulation. In order to be comprehensible, this section presents figures with only a few units. Since learning of just an arbitrarily-chosen input is very simple (see also subsection 3.3), this section does not present any such figure due to space limitations.

Fig. 3 demonstrates the model's adaptation capability. After the network has learned an initial input distribution, which has been arbitrarily chosen for illustration purposes, the network can easily adapt to input changes. In the figure, the input values from position 0-15 have been changed, and the adaptation process takes about 140 epochs. It can be seen that the model just adapts the changed input.

As Section 3 has already discussed, the speed sensor module employs neural populations for different speed values v. Fig. 4 shows five different units that are particularly trained for varying speeds $v \in \{0.83, 0.91, 1.00, 1.10, 1.20\}$. In this test, an object passes by with speed $v = 1.0$, and the figure shows how the units' activations change over time (each tic marks a particular test point). It can be clearly seen that the units' responses decrease with the difference of the test and its inherent (trained) speed. In this figure, the x and y axes represent time and activation, respectively. It should be noted that the depicted units have been particularly chosen, such that a moving object activates them at different time steps (otherwise it would became impossible to read the figure).

Fig. 4 suggests that the speed sensor units, as has been expected, react quite sensitively to a passing object. The next experiment investigates the sensor module's behavior with respect to varying object speeds. To this end, an object is passing the input device several times each time with a different speed $v_1, \ldots, v_m$. For each pass, the systems monitors the activation of a set of selected speed sensor units S_i and remembers each unit's maximal activation. These maximal activations are then plotted in a graph. Fig. 5 presents such a graph with five selected units. It can be clearly seen that the speeds are v_1=0.77, v_2=0.88, v_3=1.00, v_4=1.14, and v_5=1.30 units per time step. All units exhibit their maximal activation (not necessarily 1 due to eq. (4)) at their particular training speed, and are less active elsewhere. It is obvious that a subsequent layer, e.g., another neural layer or a fuzzy controller, can easily reconstruct the object's/robot's speed.

6 Discussion

This paper has proposed a new model to overcome existing deficiencies in the area of autonomous agents. The proposed model allows the utilization of a high-resolution input device, such as a CCD camera, by featuring a simple Hebbian-based learning rule. This learning procedure has the following advantages over currently used methods: entire online learning without any need to preselect any training pattern set, straight forward scaling property by using only local information among units, and no overlearning deficiencies and thus constant applicability with inherent adaptation properties.

The model's design is largely inspired by biological observations. However, no claim is made that in turn, any part is biological plausible; it is a just a technical application.

When setting up the system, the speed sensor module may be equipped with as many units as desired, and then each has to connected to a reference input sensor as well as the reference unit's neighborhood. In the initial training phase, the speed sensor units may be grouped to populations with each population being responsible for a particular speed.

Future research will be devoted to the following steps: The first step will be concentrating on implementing this model on a physical robot that features a two-dimensional CCD camera. The second step will be trying to substitute the gate functionality that is activated by the reference sig-

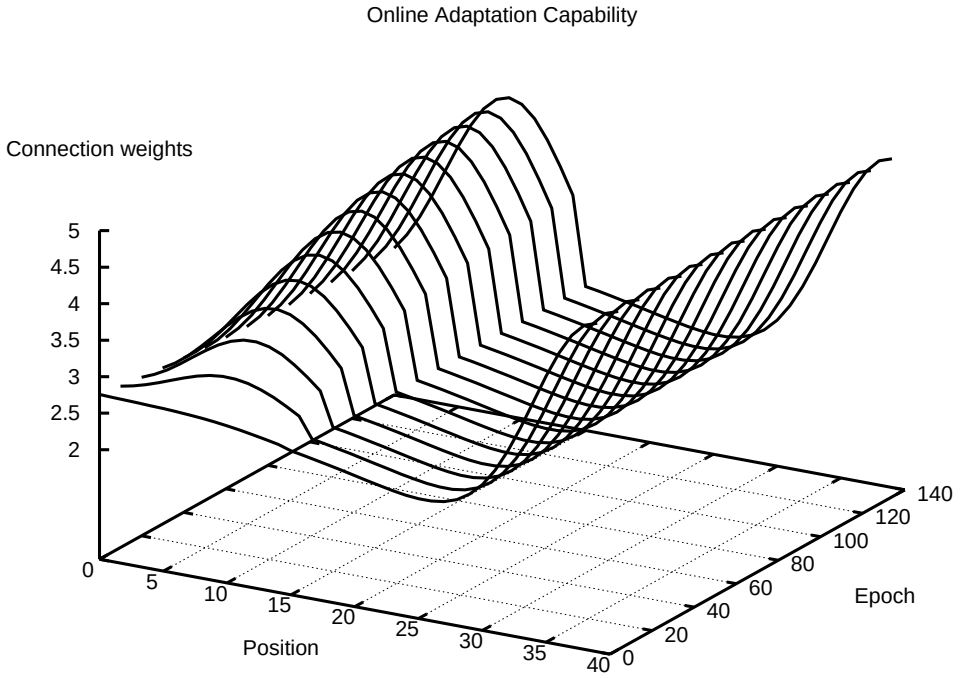

Figure 3. After the network has learned a certain input distribution (arbitrarily chosen for illustration purposes), it can adapt to a new distribution (positions 0-15) within 140 epochs.

nal by more natural mechanism; this reference signal is currently required, but substituting it would probably simplify the model.

Acknowledgements

The author gratefully thanks Lukas Lichtensteiger for pointing me to his research and all the experimental details, including the data, the robot, as well as the background literature. The author is also very grateful to the anonymous reviewers for their feedback.

References

[1] R. A. Brooks, Intelligence Without Reason. *Proceedings of the 12th Intl. Conference on Artificial Intelligence (IJCAI-91),* 1991, 569-595.

[2] R. A. Brooks, Intelligence without representation. *Artificial Intelligence,* 47, 1991, 139-159.

[3] T. S. Collett, Peering – a locust behavior pattern for obtaining motion parallax. *Journal of Experimental Biology,* 76, 1978, 237-241.

[4] N. Franceschini, J. Pichon, and C. Blanes, From insect vision to robot vision. *Philosophical Transactions of the Royal Society of London B,* 337, 1992, 283-294.

[5] G. A. Horridge, Insects which turn and look. *Endeavour* 1, 1978, 7-17.

[6] L. Lichtensteiger and P. Eggenberger, Evolving the Morphology of a Compound Eye on a Robot. *Proceedings of the Third European Workshop on Advanced Mobile Robots (Eurobot '99),* 1999.

[7] R. Pfeifer and C. Scheier, *Understanding Intelligence* (MIT Press, Cambridge, MA, 1999).

[8] R. Salomon and L. Lichtensteiger, Exploring different Coding Schemes for the Evolution of an Artifi cial Insect Eye. *Proceedings of The First IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks,* 2000, 10-16.

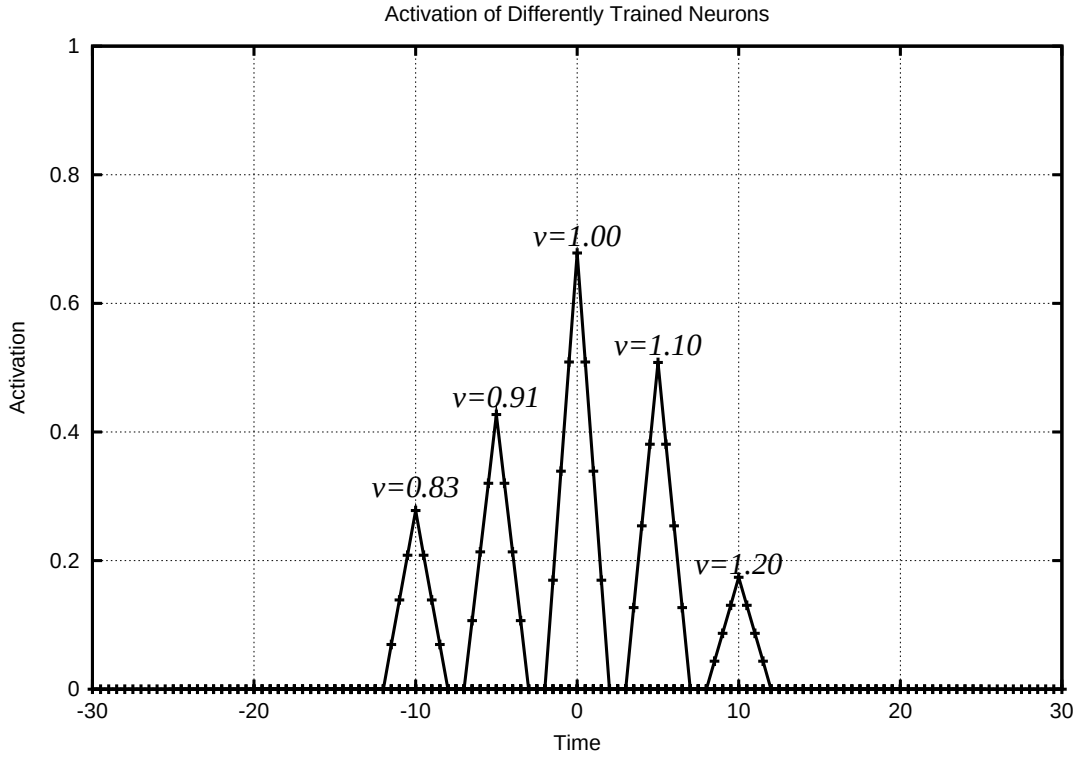

Figure 4. This figure shows the activation of five speed sensor units, each trained with a different speed v, when an object is passing by with v=1 unit per time step.

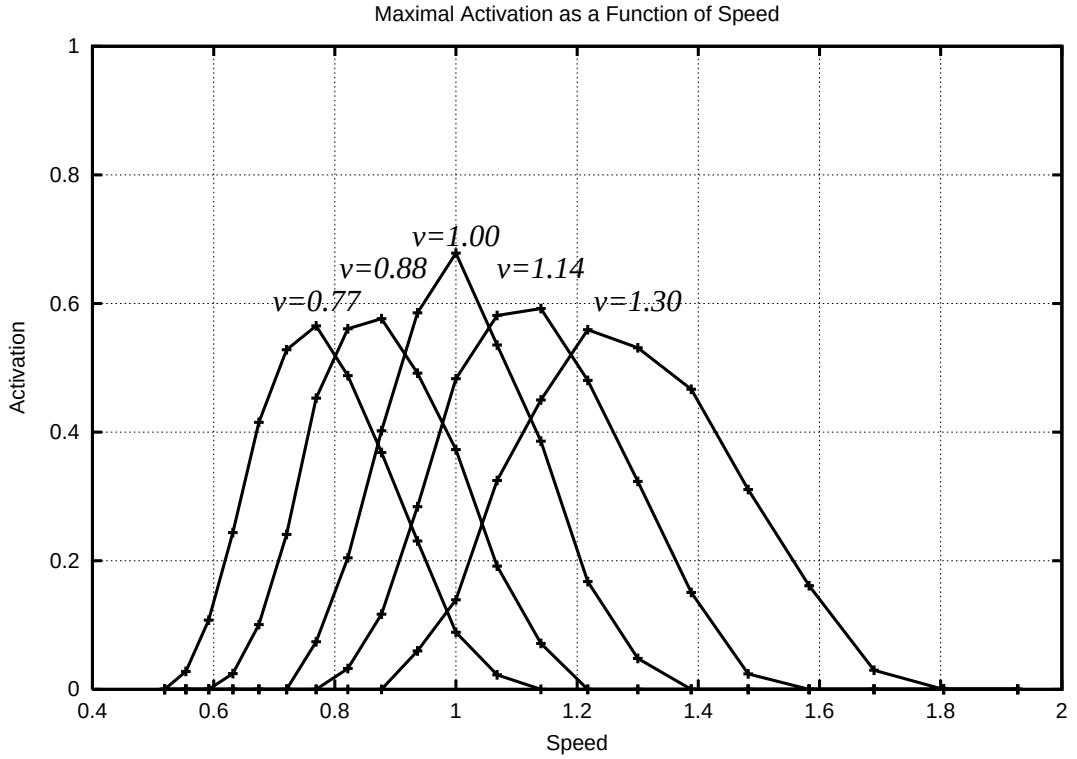

Figure 5. This figure shows how the speed sensor units I react to varying speeds. Each figure point represent the maximal activation of a unit during a complete pass of an object passing the robot.