Leveraging Saccades to Learn Smooth Pursuit: A Self-Organizing Motion Tracking Model Using Restricted Boltzmann Machines

Arjun Yogeswaran, Pierre Payeur
Department of Electrical Engineering and Computer Science
University of Ottawa
800 King Edward Ave, Ottawa, Canada
ayoge099@uottawa.ca, ppayeur@uottawa.ca

Abstract

In this paper, we propose a biologically-plausible model to explain the emergence of motion tracking behaviour in early development using unsupervised learning. The model's training is biased by a concept called retinal constancy, which measures how similar visual contents are between successive frames. This biasing is similar to a reward in reinforcement learning, but is less explicit, as it modulates the model's learning rate instead of being a learning signal itself. The model is a two-layer deep network. The first layer learns to encode visual motion, and the second layer learns to relate that motion to gaze movements, which it perceives and creates through bi-directional nodes. By randomly generating gaze movements to traverse the local visual space, desirable correlations are developed between visual motion and the appropriate gaze to nullify that motion such that maximal retinal constancy is achieved. Biologically, this is similar to using saccades to look around and learning from moments where a target and the saccade move together such that the image stays the same on the retina, and developing smooth pursuit behaviour to perform this action in the future. Restricted Boltzmann machines are used to implement this model because they can form a deep belief network, perform online learning, and act generatively. These properties all have biological equivalents and coincide with the biological plausibility of using saccades as leverage to learn smooth pursuit. This method is unique because it uses general machine learning algorithms, and their inherent generative properties, to learn from real-world data. It also implements a biological theory, uses motion instead of recognition via local searches, without temporal filtering, and learns in a fully unsupervised manner. Its tracking performance after being trained on real-world images with simulated motion is compared to its tracking performance after being trained on natural video. Results show that this model is able to successfully follow targets in natural video, despite partial occlusions, scale changes, and nonlinear motion.

Introduction

Smooth pursuit is the process by which the eye tracks a moving object in its field of view (Thier and Ilg 2005), and is driven by the direction and speed of a stimulus, resulting in continual shifts of the eye such that the moving target always falls approximately in the center of the field of view. How this mechanism develops is still under considerable debate. This work proposes a computational model to illustrate the theory by which smooth pursuit behaviour can emerge through exposure to real-world data, and demonstrates it as a motion tracking method which is tested on natural video. General-purpose machine learning algorithms are used to highlight the theory's robustness by operating without customized models.

Newborns cannot track motion, but can observe targets by redirecting their eyes through saccades (Aslin 1988). Saccades are sharp movements of the eye to change the fixation point, or gaze, and allow the brain to gather more information and learn better sensory-motor control (Canfield and Kirkham 2001). Smooth pursuit is non-existent at infants under 8 weeks of age (Aslin 1981) and improves with age as they mature (Richards and Holley 1999; von Hofsten and Rosander 1997). As it improves, tracking involves alternating between saccadic and smooth pursuit movements. This work posits that, when an infant uses a saccade to look at something and finds that the object has moved by the time it gets there, the brain must learn to produce a better prediction of how to move in the future; getting better at predicting trajectories facilitates learning to control its gaze to follow that trajectory. Controlling the gaze, instead of only receiving data, allows an additional level of interaction with the visual data available in the world. Smooth pursuit behavior can emerge via self-organization based on real-world stimuli and through feedback generated by saccades; the proposed model will illustrate this process.

Adams et al. (2012) use active inference, which builds on action and perception to minimize surprise through online learning, to model smooth pursuit. On simulations, when the network is presented with a 1-dimensional sinusoidally-moving black target on a white background, it learns to predict the trajectory using gaze movements such that pursuit behaviour emerges. Active inference uses the generative model to represent sensory information regarding position and uses those same variables to generate action. This is a principle that is also implemented by the proposed method. However, the proposed method does not hardcode prior beliefs and learns them from the data. Also, the proposed simulations occur on more complex data. Denil et al. (2012) use a deep network and particle filtering to estimate the trajectory of a target through recognition at each time step by accumulating information from local gaze searches (Larochelle and Hinton 2010). Wang and Yeung (2013) also perform tracking through classification using features learned online followed by particle filtering. The proposed method differs from the above-mentioned methods since it uses only a deep belief network, matches the trajectory through action created generatively, and does not require recognition since it models only local motion.

The recent move towards unsupervised learning, learning from large amounts of unlabeled data, has provided excellent results in a number of domains due to the data-driven approach reducing the bias - and surpassing the insight - of human designers. The concept that natural visual data can cause the emergence of physiologically-consistent results in an unsupervised learning system (Lee et al. 2007) gives evidence that real-world stimuli might be sufficient for the self-organization of a visual processing system. Reinforcement learning (Botvinick et al. 2015) has been used to great effect in recent years, by learning to do complex tasks with a basic reward mechanism, such as outperforming humans in games when learning from only raw visual data (Mnih et al. 2015; Silver et al. 2016). Given the discriminative and generative properties of deep belief networks, they are suitable to perceive action as well as create it through bi-directional nodes. Furthermore, their biological similarities via the connectionist paradigm make them an ideal candidate to show that this behaviour can be learned through only the knowledge of local connections in intermediate stacked representations.

The proposed unsupervised approach learns how to shift its gaze such that change in its receptive field is minimized between timesteps, through a model which learns to perceive and generate action. Retinal constancy is the simple mechanism required to create the association between motion and desired gaze movements. Given no prior understanding of motion information being carried from frame to frame, the most predictable event to occur is that there is no motion relative to the retina; there is retinal constancy between frames. As the system's best guess is that there is no motion relative to the retina, its behaviour is to arrive at a state such that this guess is fulfilled. When its gaze is on a stationary object, it needs to do nothing to fulfill this guess. When its gaze is on a moving object, it needs to move its gaze at the same velocity as the target to fulfill its guess. By using gaze shifts, analogous to saccades, as a method to traverse the local visual space, retinal constancy drives the network to learn correlations between visual motion and gaze movements. In a sense, this is similar to reinforcement learning, where retinal constancy is rewarded.

Motion tracking in machine vision typically consists of finding an appropriate representation for the target, a method to detect the target and match it from frame to frame, spatiotemporal constraints, and filtering to compensate for error. Biological counterparts to these concepts exist, but are less precisely-defined and the mechanisms are not as well-understood. The proposed model offers novel perspectives on many of those concepts.

Background

Restricted Boltzmann Machine

The restricted Boltzmann machine (RBM) (Hinton 2002; Smolensky 1986) is an undirected bipartite network which uses its hidden layer to represent input data from the visible layer. The RBM is an energy-based model, and calculates the energy of the joint configuration of visible nodes and hidden nodes by (1).

$$E(v,h) = -b'v - c'h - h'Wv \tag{1}$$

where v and h are the visible and hidden node states, respectively, b and c are the visible and hidden biases, respectively, and W are the symmetric weights connecting the hidden and visible nodes. The equation to determine the probability that a hidden node is on, given the visible vector, is given by (2).

$$P(h_j = 1|v) = sigmoid(c_j + \sum_i W_{ij}v_i) \tag{2}$$

where h_j is the j^{th} hidden node, v is the visible nodes vector, c_j is the bias of the j^{th} hidden node, w_{ij} is the weight connecting the i^{th} visible node, v_i, and h_j. The sum is over all visible nodes. The equation to determine the probability that a visible node is on, given the hidden vector, is given by (3).

$$P(v_i = 1|h) = sigmoid(b_i + \sum_j W_{ij}h_j) \tag{3}$$

where v_i is the i^{th} visible node, h is the hidden nodes vector, b_i is the bias of the i^{th} visible node. The sum is over all hidden nodes.

Training is accomplished using contrastive divergence (CD), and involves lowering the energy for preferred configurations of hidden nodes and visible nodes, and raising the energy for undesirable configurations. The training alternates between the positive phase and negative phase, where the positive phase samples the hidden state, h^+, and the visible state, v^+, from the data while the negative phase produces the reconstructions of the hidden state, h^-, and the visible state, v^-.

$$\Delta w_{ij} = \gamma \cdot [< v_i^+ h_j^+ > - < v_i^- h_j^- >] \tag{4}$$

where γ is the learning rate, and $< \cdot >$ is the average over a number of samples.

Gated Factored Restricted Boltzmann Machine

The gated factored restricted Boltzmann machine (GRBM) (Memisevic and Hinton 2010), an extension of the RBM model to learn higher-order relationships in the data, models the joint distribution between inputs, outputs, and hidden nodes, using intermediate factors. Also trained using CD, this generative model tries to predict the output, y, from the input, x, allowing the hidden nodes to develop efficient codes for the observed transformations between x and y. With visual data, this technique can be applied to learning spatial transformations. The energy of the joint configuration of the nodes is defined in (5).

$$E(x,y,h) = -\sum_{f=1}^{F}\sum_{ijk} x_i y_j h_k w_{if}^x w_{jf}^y w_{kf}^h + \sum_k w_k^h h_k + \sum_j w_j^y y_j \tag{5}$$

where w_{if}, w_{jf}, and w_{kf} are the weights connecting factor f to the input, output, and hidden nodes which are labeled x, y, and h, respectively. The probability that hidden node k is on, given the input and output vector, x and y, respectively, is calculated by (6). The probability that the output node j is on, given the input and hidden vector, x and h, is calculated by (7).

$$P(h_k = 1|x,y) = sigmoid(c_k + \sum_f w_{kf}^h + \sum_i x_i w_{if}^x + \sum_j y_j w_{jf}^y) \tag{6}$$

$$P(y_j = 1|x,h) = sigmoid(b_j + \sum_f w_{jf}^y + \sum_i x_i w_{if}^x + \sum_k h_k w_{kf}^h) \tag{7}$$

Deep Belief Network

It is possible to layer several RBMs on top of each other, where the hidden nodes of one layer behave as the visible nodes of the next, forming a more powerful representation of the data with increasing levels. Each RBM is trained in a greedy manner and only learns correlations in the data from the layer below it. This type of architecture is considered a deep architecture, which can also be used as a generative model. By presenting input data, activations propagate upwards until the top most layer, which then behaves as an autoassociative memory that generates activations that propagate downwards until they generate examples at the input layer. This is known as a deep belief network (DBN) and can be used for classification as well as generation. Hinton (2007) uses it to act discriminatively by classifying hand-written digits from the MNIST dataset (Lecun et al. 1998) through generation of associated labels at the visible nodes.

Model

The proposed model contains a motion encoding layer, a motion tracking layer, and the receptive field. The receptive field is the visual content of a region of interest, and its position is controlled by gaze movements generated by the motion tracking layer. The motion encoding layer learns to represent motion in the receptive field, while the motion tracking layer learns gaze control from the output of the motion encoding layer. The network has some general parameters as follows:

$$gaze \in \Gamma \tag{8}$$

$$h_1 = f_1(x^{(t)}, x^{(t+\Delta t_m)}) \tag{9}$$

where gaze controls the receptive field and belongs to the set of all allowable transformations, Γ. h_1 is the encoded motion vector, and f_1 is the motion encoding output function, where $x^{(t)}$ is the receptive field image at time t, and Δt_m is the sampling delay between frames such that motion can be perceived. There are slight differences in the formalization between training and execution. The training is as follows:

$$h_2^{train} = f_2(h_1, gaze^{train}, r(x^{(t)}, x^{(t+\Delta t_m+\Delta t_g)}) \tag{10}$$

where f_2 is the motion tracking output function, based on the motion encoding output, a random gaze, and the retinal constancy value, r. Δt_g is the delay between the time the motion is perceived to the time that the gaze catches up to the target. $t + \Delta t_m + \Delta t_g$ is also the time at which retinal constancy is evaluated during training. Effectively, the target's motion between t and $t + \Delta t_m$ is used to estimate the target's position at $t + \Delta t_m + \Delta t_g$. During execution, the motion tracking output function relies solely on the motion vector. The gaze is calculated by $f_2{}'$, an inversion of the motion tracking output function that behaves generatively. This is formalized as follows:

$$h_2^{execution} = f_2(h_1) \tag{11}$$

$$gaze^{execution} = f_2{}'(h_2^{execution}) \tag{12}$$

When using RBMs to implement these functions, the system can be modeled as a two-layer DBN. The receptive field and motion encoding layer propagates data upwards, while the motion tracking layer generatively produces gaze movements as feedback downwards. The circular dependencies between them, as well as the details of training and execution, are illustrated in Figure 1. The timing relationships to the input, output, and the processing are shown in Figure 2.

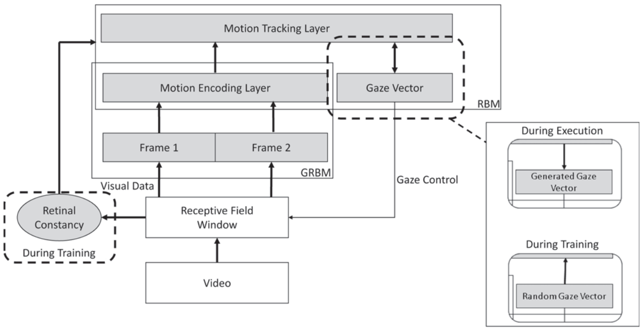

Figure 1: Complete system description, where the gaze vector is generated by the motion tracking layer during execution, and created randomly during training. The hidden activations of the motion encoding layer and the gaze vector are concatenated to form the input vector of the motion tracking layer.

From the machine vision perspective, motion encoding consists of representing the target, and estimating possible target matches from frame to frame. Motion tracking consists of selecting the best match and moving the region-of-interest to match that target. The proposed model exhibits a

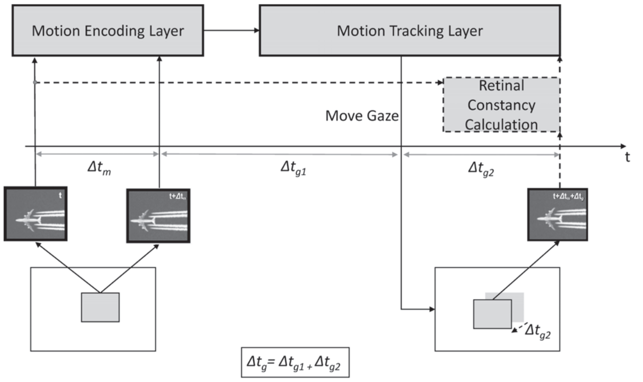

Figure 2: Timing and input/output diagram. Items using dotted lines occur only during training.

biologically-plausible self-organizing method for those two components.

Motion Encoding

The motion encoding layer calculates motion information from two image frames separated by a finite time interval, Δt_m, as in (9). This encoding is trivial when hardcoded, such as in traditional machine vision methods, but a primary directive of this work is to allow such properties to be learned.

The implementation uses the GRBM to learn transformations between frames, with the learned transformations encoded by the hidden nodes. The GRBM's hidden layer nodes represent translations and skews when trained on real-world video data (Memisevic and Hinton 2010), making it ideal for this task. Also, to meet the other goals of this work, it is a general machine learning algorithm, and can be a part of the DBN since it is an extension of the RBM.

Motion Tracking

The goal of the motion tracking layer is to associate the motion vector produced by the motion encoding layer with the appropriate gaze movements, such that those gaze movements can be created as defined by (12). This is implemented by training a bi-directional autoassociative memory with the learning rate modulated by the retinal constancy value. The RBM is chosen because of its generative properties, it being a general machine learning algorithm, and because it can be part of the DBN.

This association is learned by concatenating the motion vector and gaze movement vector into a single training example, implementing (10). During training, it learns associations when they have a high retinal constancy. During execution, it generates the gaze through the execution of one Gibbs step, initialized with the motion vector, implementing (11) and (12).

The RBM is suitable to learn of deeper neural functions from associations in a lower layer, as evidenced by the progressively more complex features developed in the 2-layer deep network proposed by Lee et al. (2007), similar to those found at increasing layers of the visual cortex. Hinton shows that the RBM can also be used for classification generatively, demonstrated by MNIST label generation in a deep belief network (Hinton 2007). Computationally, the network behaves similar to that work, where gaze movements are generated in a manner similar to the labels through top-down generation created by bottom-up stimulation. Biologically, this layer is similar to the middle superior temporal area in the brain, since it has been shown to produce eye movements relating to pursuit.

Gaze Movement Vector

The gaze movement vector both perceives the gaze movement and creates it. In the case of tracking in video, it both captures and controls the position of the virtual receptive field in the image. The gaze movement vector encodes translation, as in (13). This is close to the biological eye, which rotates about two axes.

$$\Gamma = (r, s)|(r, s) \in ([-i, +i], [-j, +j]) \tag{13}$$

where (r, s) is a translation vector within [-i,+i] horizontal range and [-j,+j] vertical range. The vector is implemented by dividing it into a half for left-right movement and a half for up-down movement. Each element represents a velocity in that axis. For example, in a range of -2 to +2, a vector of 10000 00010 indicates a -2 horizontal and a +1 vertical shift. During training, gaze movements are randomized to simulate saccades, with exactly one element in each axis set to 1. During execution, this vector is generated by the motion tracking RBM, and the position in each axis with the highest value is chosen as the shift in that direction.

Retinal Constancy

In this model, retinal constancy is how desirable saccadic movements are linked to detected motion. It is a non-trivial solution, because perception relates to action solely through comparison snapshots of the receptive field. The comparison does not describe how the visual contents change, and occurs despite temporal delays. Retinal constancy is calculated as a function of the difference between images. In this implementation, raw pixel data is used, using (14), and is applied in the training of the motion tracking layer from (10).

$$r(x^{(t_1)}, x^{(t_2)}) = max(0, 1 - \alpha \sum_{i=1}^{N} \frac{(x_i^{t_1} - x_i^{t_2})^2}{N}) \tag{14}$$

where N is the number of pixels in image patch x. The coefficient, α, is the selectivity of the comparison. Higher selectivity causes the network to have fewer samples from which to learn, yet the accuracy of those samples will be higher. Lower selectivity allows the network to see more samples, even if they are not necessarily good ones.

Retinal constancy is used to bias the motion tracking layer to learn favorable patterns; this is achieved by throttling the network's learning rate with retinal constancy as a multiplying coefficient during training. Equation (15) shows the CD weight update equation for the motion tracking layer, updated from (4).

$$\Delta w_{ij} = \gamma \cdot r(x^{(t)}, x^{(t+\Delta t_m+\Delta t_g)}) \cdot \\ [< v_i^+ h_j^+ > - < v_i^- h_j^- >] \qquad (15)$$

Biologically, memory storage and recall can be affected by factors such as saliency and emotion (Sandberg et al. 2001), therefore, this is biologically-plausible since this layer is an autoassociative memory.

Experiments

Experimental Parameters

The simulation enforces practical limits on (13) by limiting the magnitude of the gaze and, correspondingly, the magnitude of possible transformations to encode by:

$$\Gamma = (r, s) | (r, s) \in ([-2, +2], [-2, +2]) \qquad (16)$$

This corresponds to limiting gaze control to -2 to +2 pixels both horizontally and vertically, where $gaze \in \Gamma$. Image patches from the van Hateren natural image database (van Hateren and van der Schaaf 1998) are used as training data for both the motion encoding layer and the motion tracking layer, with motion simulated by applying transformations corresponding to the ranges defined by Γ. Experiments are also performed by training the motion encoding layer with unconstrained real-world video taken from the Hollywood2 dataset (Marszalek 2009). Random gaze movements are also selected from Γ with a probability of $p = 0.5$. Each transformation has an equal probability of being chosen for both the random gaze movement and the motion simulation. Receptive field sizes were fixed to 14x14. Raw grayscale frames are used with local contrast normalization and thresholded to leave only areas of positive contrast. Retinal constancy uses a selectivity coefficient, α, of 100. CD was used for the training of all RBMs, with one weight update per training pattern for the motion tracking RBM, and one weight update per batch of 100 for the motion encoding GRBM. The GRBM has 200 factors, 100 hidden nodes, and 196 visible nodes for each frame. For the purposes of computational motion tracking, $\Delta t_m = 1$ frame, meaning two consecutive image frames are used to estimate motion. Δt_g is set to 1, meaning that the saccade is driven to the target in the frame after when the motion is perceived. The motion tracking RBM's hidden layer contains only 25 nodes such that it must be efficient about the correlations it learns. Since there is no filtering built into the model, a convolution procedure is used to increase stability. The motion encoding is executed on the contents of the receptive field as well as the nearest non-overlapping windows surrounding it, and a mean pooling operation is performed on the resulting vectors to produce the final motion vector, h_1.

Results

The network is first tested on patches taken from the van Hateren natural image database undergoing the limited subset of allowable transformations in Γ. The tracking accuracy was calculated by how frequently the translated image pairs fed to the network generate the correct gaze control to nullify the translation. The mean results of tracking accuracy relative to increasing amounts of training samples over 5 trials are shown in Figure 3 where the motion encoding layer is trained with simulated and real-world motion and the motion tracking layer is trained with varying simulated motion ranges.

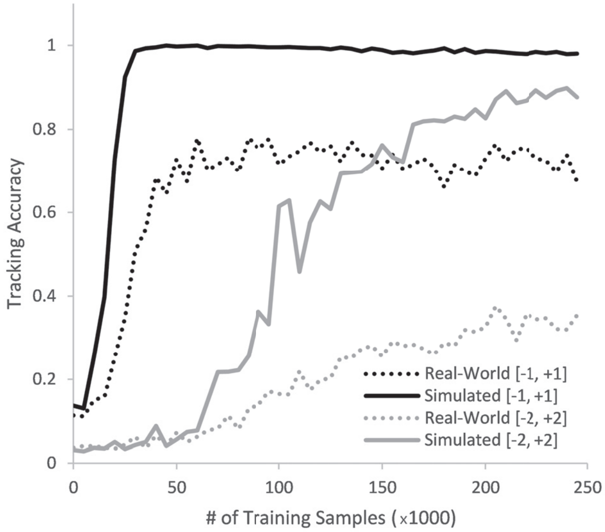

Figure 3: Comparison of tracking accuracy vs number of training samples seen. Model configurations have the motion encoding layer trained on simulated motion and natural video, and the motion tracking trained on different shift ranges.

It can be seen that training the motion encoding layer with simulated motion produces very good results, regardless of the motion range. The results decrease when training with real-world video; though it is still acceptable with the [-1,+1] motion range, the motion range of [-2,+2] suffers a performance decrease. This discrepancy is due to the diversity, balance, and cleanliness of the simulated motion training data compared to the complex real-world data. Having a smaller motion range produces better tracking accuracy, as evidenced by the performance of the motion tracking layer operating in the [-1,+1] pixel range both horizontally and vertically. Yet the simulated [-2,+2] pixel range still produces robust results. To evaluate this robustness in a more challenging task, and for a more realistic execution, the system was tested on real-world video footage.

Figure 4 shows selected frames from real-world videos where the motion tracking system, trained on simulated motion with a [-2,+2] range, follows a manually-selected target from beginning to end. A video of the network tracking a truck on the highway with relatively smooth linear motion, over the course of 140 frames, is shown in the upper row. The second example provides a more complex motion with an object that changes scale by showing a perspective of a soccer ball being shot. This shows the network's ability to follow the target on a more complex motion, including a changing pattern due to rotation, a changing scale due to distance, and illumination changes. This tracking occurs over the course of 150 frames. The third video is that of a flock of birds,

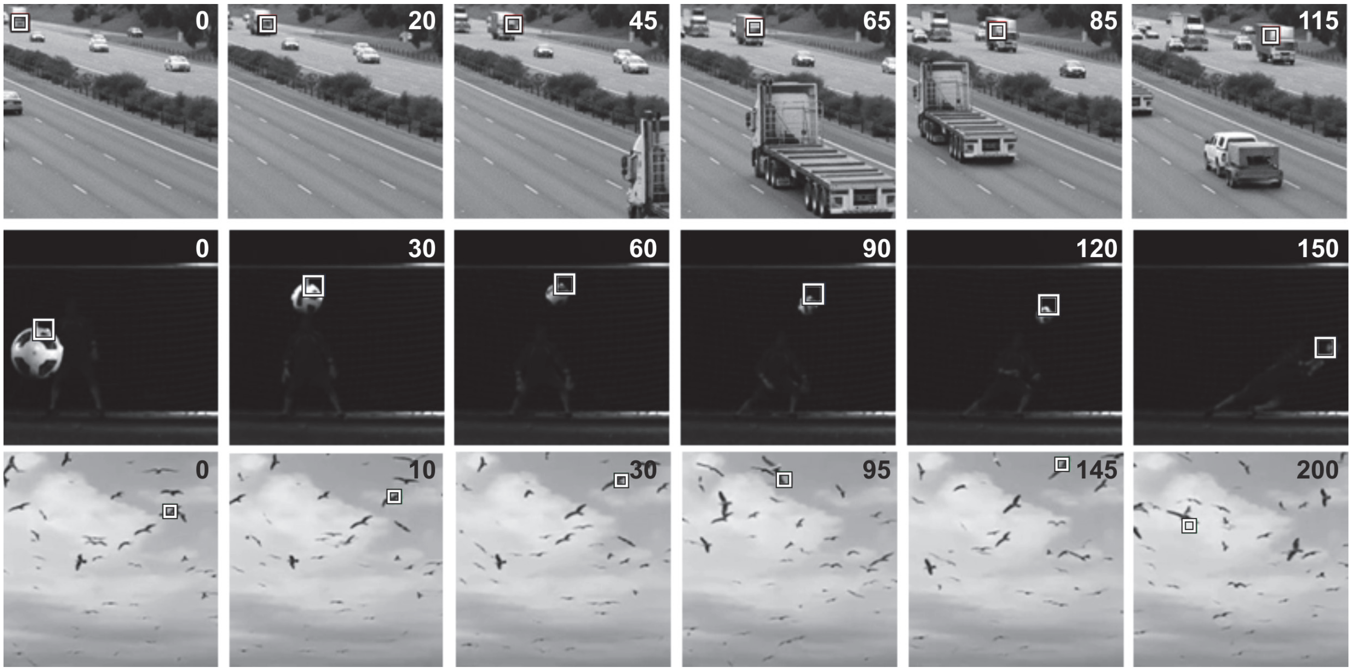

Figure 4: Selected frames of the network tracking objects in real-world videos. The three videos show the network tracking a truck on a highway (top), a soccer ball in flight (middle), and a bird in a flock (bottom). The rectangle shows the network's receptive field and gaze position. The frame number is listed in the top-right corner of each frame.

which shows the network's ability to track an object which changes shape and its ability to tolerate overlapping motion, performed over 240 frames. The network follows a single bird through a curved trajectory.

It can be seen from the results that the proposed network is very effective at tracking the motion of a particular image pattern once it is in the receptive field. Even with unconstrained real-world video, it is accurate in its motion range, and is capable of tracking a wide variety of targets over a variety of motion through a behaviour learned exclusively by self-organization.

Conclusion

As a whole, the concept of motion tracking is well-researched in machine vision, and its counterpart, smooth pursuit, is well-researched in biology. There is less focus on computational models of smooth pursuit, and most of the emphasis is found in specific implementations to simulate certain aspects of it. This work aims to promote the idea of unifying perception and action through the representational and generative capabilities of a biologically-plausible model, as well as postulating that exposure to natural video is sufficient for tracking behaviour to emerge. The results show that it performs well on real-world data both for training and execution, which is a significant leap forward from simulations that are typically performed on simplistic synthetic data. The proposed motion tracking model is positioned well within the realm of computational models of biological smooth pursuit, machine vision, and unsupervised learning, and serves to contribute novel ideas to each of those fields.

References

Adams, R. A.; Perrinet, L.; and Friston, K. 2012. Smooth pursuit and visual occlusion: active inference and oculomotor control in schizophrenia. *PLoS ONE* 10: e47502.

Aslin, R. N. 1981. Development of smooth pursuit in human infants. *Eye Movements: Cognition and Visual Perception* 31-51. Hillsdale, NJ: Erlbaum

Aslin, R. N. 1988. Anatomical Constraints on Oculomotor Development: Implications for Infant Perception. *Perceptual Development in Infancy: The Minnesota Symposia on Child Psychology* 20. Hillsdale, NJ: Erlbaum.

Botvinick, M.; Weinstein, A.; Solway, A.; and Barto, A. 2015. Reinforcement learning, efficient coding, and the statistics of natural tasks, *Current Opinion in Behavioral Sciences* 5: 71-77.

Canfield, R. L.; and Kirkham, N. Z. 2001. Infant cortical development and the prospective control of saccadic eye movements. *Infancy* 2: 197-211.

Denil, M.; Bazzani, L.; Larochelle, H.; and de Freitas, N. 2012. Learning where to attend with deep architectures for image tracking. *Neural Computation* 24: 2151-2184.

Hinton, G. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation* 14: 1771-1800.

Hinton, G. 2007. Learning multiple layers of representation. *Trends in Cognitive Sciences* 11: 428-434.

Larochelle, H.; and Hinton, G. 2010. Learning to combine foveal glimpses with a third-order Boltzmann machine. *Ad-*

vances in Neural Information Processing Systems 1243-1251.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86: 2278-2324.

Lee, H.; Ekanadham C.; and Andrew, N. Y. 2007. Sparse deep belief net models for visual area V2. *Advances in Neural Information Processing Systems* 873-880

Marszalek, M.; Laptev, I.; and Schmid, C. 2009. Actions in Context. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2929-2936.

Memisevic, R.; and Hinton, G. 2010. Learning to represent spatial transformations with factored higher-order Boltzmann machines. *Neural Computation* 22: 1473-1492.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.; Veness, J.; Bellemare, M. et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518: 529-533.

Richards, J. E.; and Holley, F. B. 1999. Infant attention and the development of smooth pursuit tracking. *Developmental Psychology* 35: 856-867.

Sandberg, A.; Lansner, A.; and Petersson, K. M.; 2001. Selective enhancement of recall through plasticity modulation in an autoassociative memory. *Neurocomputing* 38: 867-873.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; van den Driessche, G. et al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529: 484-489.

Smolensky, P. 1986. Information Processing in Dynamical Systems: Foundations of Harmony Theory. *Parallel Distributed Processing: explorations in microstructure of cognition* 1: 194-281. D. Rumelhart and J. McClelland, MIT Press

Thier, P.; and Ilg, U. J. 2005. The Neural Basis of Smooth-Pursuit Eye Movements. *Current Opinion in Neurobiology* 15: 645-652.

van Hateren, L.; and van der Schaaf, A. 1998. Independent component filters of natural images compared with simple cells in primary visual cortex. *Proceedings of the Royal Society of London B: Biological Sciences* 265: 359-366.

Von Hofsten, C.; and Rosander, K. 1997. Development of smooth pursuit tracking in young infants. *Vision Research* 37: 1799-1810.

Wang, N.; and Yeung, D. 2013. Learning a Deep Compact Image Representation for Visual Tracking. *Advances in Neural Information Processing Systems* 809-817.