SAND2020-9979R

LABORATORY DIRECTED RESEARCH & DEVELOPMENT
WHERE INNOVATION BEGINS

SAND20XX-XXXXR
LDRD PROJECT NUMBER: 218973
LDRD PROJECT TITLE: Neuromorphic Processing and Sensing for Interception
PROJECT TEAM MEMBERS: Frances S. Chance

ABSTRACT

Interception of a moving and potentially evading target can be a challenging problem, in particular for conditions in which the target may be moving at high speeds and difficult to detect. We have proposed to merge two Sandia LDRD efforts, the SPARR Spiking/Processing Array (neuromorphic event-driven sensing) and the Dragonfly-Inspired Algorithms for Intercept-Trajectory Planning (neural-inspired algorithms for interception) toward a unified system with direct application to national security. Neuromorphic systems demonstrate the most potential for speed and efficiency gains when communication is event-driven and computations are simple but parallelizable. Accordingly, we anticipate fully realizing potential benefits from a neuromorphic interception system if event-driven sensing is combined with processing and acting also implemented on event-driven (spiking) systems. We have successfully translated a neural-inspired interception algorithm to a neural network architecture for evaluation on neuromorphic hardware. Preliminary implementations of the neural network designed for implementation on the Loihi chip are still too immature for conclusive evaluation, but the results of this effort have demonstrated a viable path for a previously developed dragonfly-inspired interception algorithm to be implemented on neuromorphic hardware.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

Sandia National Laboratories

U.S. DEPARTMENT OF ENERGY

INTRODUCTION AND EXECUTIVE SUMMARY OF RESULTS

Interception of an evading target requires simultaneous sensing of and acting on observations of a target. Interception in hypersonics is particularly challenging because the extreme speeds of both the target and interceptor require extremely fast sensing and processing. The objective of this effort was to combine the SPARR Spiking/Processing Array (neuromorphic event-driven sensing) with the Dragonfly-Inspired Algorithms for Intercept-Trajectory Planning (a neural-inspired algorithm for interception based upon biological dragonflies) to demonstrate the potential of processing on neuromorphic hardware specifically for interception systems (see Addendum B, end-of-year slides). Our hypothesis is that the spiking sensing modality of SPARR will combine synergistically with a spiking neural network (SNN) version of the dragonfly algorithm implemented on neuromorphic hardware, revealing potential performance advantages to a next-generation interception system.

Spiking neuromorphic platforms offer a trade-off between increased numbers of processors with decreased computational complexity per processor. Our expectation was that the highest speed and efficiency gains would arise when supporting algorithms were highly parallelizable. Moreover, to take full advantage of the speed and efficiency gains offered by SPARR, a focal plane that outputs spikes, our expectation was that best performance would arise from processing on spiking hardware that could receive information in the native format of the sensor (spiking). Before performance of SPARR and dragonfly operating together could be evaluated, the dragonfly model (see Chance, 2019) needed to be converted to an architecture that could receive spikes from SPARR. Accordingly, the first steps to this end were to 1) convert the dragonfly-inspired interception algorithm to a neural network (NN) architecture and 2) implementing the model on a spiking neuromorphic platform for evaluation.

We have successfully developed a continuous-valued NN version of the dragonfly-inspired interception algorithm. The NN model is fully described in Chance, 2020 (also included as Addendum A to this report). We have also successfully used Nengo (Bekolay et al., 2014) to develop a SNN version for deployment on the Intel Loihi chip (Davies et al., 2018). As of the writing of this report, the Nengo-developed version of the dragonfly NN runs on the Nengo Loihi-emulator, but currently requires additional trouble-shooting before evaluation on Loihi hardware can be performed (individual components of the network have been run on the hardware but not a full implementation of the model). Future work will also consider alternate implementations of the dragonfly algorithm as a SNN for evaluation on the Loihi platform.

DETAILED DESCRIPTION OF RESEARCH AND DEVELOPMENT AND METHODOLOGY

This effort leveraged a previously developed interception algorithm based upon biological dragonflies. Full details of the dragonly-inspired interception algorithm are provided in Chance, 2019. The objective of this project was to convert the dragonfly model (Chance, 2019) to a form that could be implemented on neuromorphic hardware for further evaluation (e.g. for size, weight and power requirements). While "best practices" for neuromorphic implementations of algorithms is still very much an active and open area of research, it is generally assumed that maximum gains in efficiency and speed are realized when computations are relatively simple and parallelizable. Accordingly, a spiking neural network (SNN) implementation was identified as an attractive option for fully realizing potential advantages offered by emerging neuromorphic hardware.

As a first step towards a neuromorphic implementation, the dragonfly model was first translated to a continuous-valued neural network, fully described in Chance 2020 (also see Addendum A). The general architecture (layers, connectivity, synaptic weight values) of this model was very influential when developing the spiking implementations of the dragonfly model. The Intel Loihi chip (Davies et al., 2018) is a recently developed neuromorphic chip that implements asynchronous spiking neural networks. This particular platform was of interest to this project because it is one of the first neuromorphic platforms to incorporate synaptic plasticity and thus could accommodate future potential neural-inspired modifications to the dragonfly model.

<u>Nengo implementation</u>

Multiple options exist for implementing neural networks on Loihi ranging from software packages to hand-crafted spiking neural networks. Nengo (Bekolay et al., 2014) is a relatively mature software package specifically developed for constructing, testing, and deploying neural networks. The software has previously been used (Blouw et al., 2018) to demonstrate energetic advantages for deep learning algorithms implemented on Loihi compared to other computing devices (ranging from traditional CPUs to alternate neuromorphic hardwares) and therefore presented a logical path for implementing a "first-pass" of the dragonfly algorithm on Loihi.

The Nengo implementation of the dragonfly neural network relies heavily on the use of "ensembles", a population of neurons whose firing rates encode a particular value or function. For the Nengo implementation of dragonfly, each neuron of the continuous-valued NN version was approximated by a population of 256 neurons. For example, in the continuous-valued NN, the responses of individual neurons from the prey-image and the fovea-position representations (see Chance, 2020 and Addendum A) were determined by Gaussian functions of the distance between the preferred position of a neuron and the input position (prey-image position or fovea position). Each of these neurons was replaced by an ensemble of neurons whose firing rates approximated the same Gaussian function. Likewise, individual neurons in the motor output population of the continuous-valued NN were replaced by ensembles of neurons in the Nengo implementation. Connection weights between ensembles of neurons in the Nengo implementation were determined by the synaptic weights between analogous neurons in the continuous-valued NN.

The advantage to this approach was that many functions are built-in to Nengo. For example, multiplication of inputs from two neurons of the continuous-valued NN is achieved using a dot product between two ensembles. The disadvantage is that significantly more neurons are required. The continuous-valued NN presented in Chance (2020) comprised of 441 (21x21) neurons for the prey-image position, fovea position, and motor output populations, and 194,481 (21^4) neurons in the sensory representation population. The Nengo implementation requires 256 times the number of neurons in the continuous-valued NN, suggesting that experiments with the physical hardware might be required before definitive comparisons of energy-requirements and processing speed between the two implementations can be made. While the Nengo implementation has been successfully tested on the Nengo Loihi chip emulator, additional troubleshooting is required to run the Nengo implementation on the Loihi chip (while individual components, for example the prey-image population, can be run on the chip, the full implementation has not been run successfully to date).

Hand-crafted implementation

In parallel to the Nengo implementation, a simplified version of the dragonfly model (specifically, one that only does classical pursuit – see Chance 2019) was converted to a spiking NN (developed in Matlab) by hand. The intent was not to develop a spiking implementation for direct comparison against the Nengo implementation, but instead to develop a smaller spiking network that could be used as a "pilot" network for evaluating what network specifications most significantly impact efficiency and speed of the network. This approach to developing a dragonfly SNN also allowed us to identify certain functions (e.g. normalization) to identify in future development efforts.

Briefly, each neuron in the continuous-valued neural network was replaced by a single spiking neuron. In this implementation, individual neurons represent a single continuous-valued variable, V. If $V \geq V_{\text{th}}$, a spike is fired and V is set to V_{reset}. If $V < V_{\text{th}}$, V is multiplied by $\exp(-dt/\tau)$, causing V to exponentially decay to zero. Input to the prey-image representation is calculated as in Chance, 2020. Input to motor output neuron j is $\sum_{i,j} W_{ij} s_i$, where $s_i = 1$ if prey-image neuron i spiked in the previous time step and 0 otherwise. The synaptic weight between prey-image neuron i and motor output neuron j is

$$W_{ij} = \iint dx_1 dx_2 \exp\left(-\frac{(a_{i1}-x_1)^2+(a_{i2}-x_2)^2}{2\sigma_r^2}\right)\exp\left(-\frac{(c_{j1}-x_1)^2+(c_{j2}-x_2)^2}{2\sigma_m^2}\right),$$

where (a_{i1}, a_{i2}) is the preferred prey-image location of prey-image neuron i and $\left(c_{j1}, c_{j2}\right)$ is the preferred goal location of motor output neuron j, and (x_1, x_2) is the location of the prey in eye coordinates.

This implementation was intended to be a simplified version of the dragonfly algorithm and therefore did not include a fovea-position population or a sensory-representation population. The fovea remained fixed at the center of the eye. As a result, the model dragonfly driven by this SNN only exhibits classical pursuit.

RESULTS AND DISCUSSION

Dragonfly Neural Network

As previously indicated, the NN version of the dragonfly-inspired interception algorithm is described in Addendum A (also Chance, 2020). For the purposes of this effort, it is important to note that the NN version generates trajectories that are almost identical to those generated by the original model. In Figure 1, a trajectory generated by the original dragonfly-inspired interception model (open black symbols) is compared with a trajectory generated by the NN version of the dragonfly (filled blue symbols). Close inspection of the figure reveals slight differences between the trajectories generated by the two versions of the model, most likely arising from differences in the resolution with which the motor output is encoded. The similarity between the two models has been reproduced for other target trajectories (including ones in which the target changes direction, not shown) and allowed us to conclude that the NN version produces equivalent trajectories to the original version of the model.

The original dragonfly model was described in Chance (2019). The NN version (filled blue symbols) is described Addendum A (also see Chance, 2020). For the NN version, the prey-image position, fovea position, and motor output populations consisted of 441 (21x21) neurons each. The sensory representation population consisted of 194,481 (21^4) neurons. Prey-image position, fovea position, and motor output neuron preferences were an evenly spaced grid ranging from [-$l\tan(\theta)$, [$l\tan(\theta)$], where l is the distance from the center of the dragonfly's head to the center of the eye, and θ is $\pi/2.1$ for the prey-image position and fovea position populations and $\pi/4$ for the motor output population. Motor output activity threshold was 16.

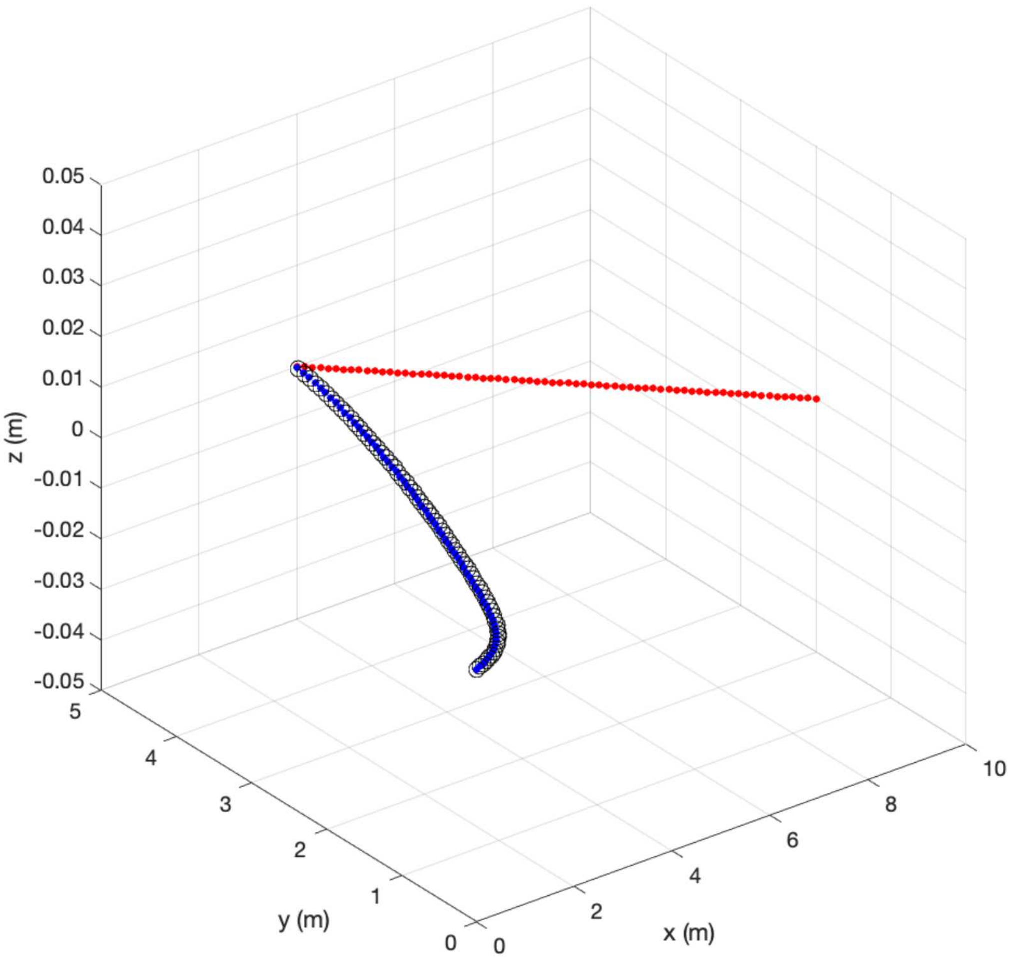


Figure 1: comparison of an interception trajectory generated by the original dragonfly-inspired interception algorithm (open black symbols) with a trajectory generated by the continuous-valued neural network version of the dragonfly model (filled blue symbols). Target trajectory is indicated by the red symbols.

Nengo Implementation

Additional troubleshooting and debugging will be required to fully understand the advantages and disadvantages to the Nengo implementation of the dragonfly model. While the ensemble approach has demonstrated that the full model can, in principle, be implemented on the Loihi hardware, we have encountered certain issues that must be addressed before any statements about the benefits of using this implementation on Loihi hardware.

Hand-crafted Implementation
The "hand-crafted" implementation will likely need further development before it can be extended to incorporate the full dragonfly model. The current network exhibits sensitivity to the number of prey-image position neurons that are activated by the stimulus. Our current assessment suggests that a mechanism for normalizing the input to a population of neurons will do much to enhance the stability of the activity patterns in each population. This mechanism will likely be tailored to the specific hardware platform (in this case the Loihi chip).

ANTICIPATED OUTCOMES AND IMPACTS

Our results demonstrate that there is a viable path forward to an implementation of the dragonfly-inspired interception algorithm on a neuromorphic platform, but additional work is required before definitive statements can be made regarding any potential speed, SWaP, or efficiency advantages. It is clear from our results that "best practices" for neuromorphic implementations are not established. An ideal outcome of this effort will be for development of the dragonfly algorithm for neuromorphic implementation to continue. The expertise gained from this LDRD, in particular familiarity with the Loihi chip, will be very beneficial to this effort going forward.

As described in Addendum B, this LDRD has supported one journal publication (with two additional publications submitted), one conference publication, three invited presentations internal to SNL and one conference presentation. The PI received an invitation to speak at a special conference session this past March, but the conference has been postponed due to the COVID-19 pandemic. If we are able to find support, we plan to submit at least one additional publication describing the neuromorphic implementation of dragonfly (once complete) and at least one white paper describing the advantages and disadvantages of implementing the dragonfly algorithm on the Loihi chip.

This LDRD has been synergistic with other projects to evaluate the potential value of a neuromorphic interception system to SNL customers and mission areas. For example, the PI (in collaboration with members of 6530 and additional members from 1421) has just completed a project evaluating the feasibility of implementing the dragonfly algorithm on a physical vehicle and identifying the major hurdles to interfacing the neural-inspired algorithm with a "real-world" system. This LDRD has also supported development of expertise with neural-inspired algorithmic development and neuromorphic implementations. These increased capabilities will impact the broader vision to leverage emerging neuromorphic platforms and neuroscience knowledge to benefit SNL's national security missions.

CONCLUSION

Interception of a moving and potentially evading target remains an unsolved problem. The goal of this effort was to evaluate potential efficiency and speed advantages to be gained from combining neuromorphic event-driven sensing (e.g. SPARR) with a neural-inspired interception algorithm implemented on neuromorphic hardware. We have demonstrated that a previously developed neural-inspired interception algorithm (from "Dragonfly-Inspired Algorithms for Intercept-Trajectory Planning", see Chance 2019) can be translated into a neural network without significantly impacting performance. We have also demonstrated that a spiking implementation of this neural network can be run on a neuromorphic emulator (specifically the Nengo emulator of the Intel Loihi chip). Future work will maturing neuromorphic implementation(s) of the algorithm for benchmarking on the Loihi chip.

ACKNOWLEDGMENTS

Luke Parker developed the Nengo implementation of the dragonfly neural network with guidance from Dr. Suma Cardwell. Park Hays and David Karelitz provided beneficial consultation describing the outputs of the SPARR platform.

REFERENCES

(includes references contained in Addendums)

Bekolay, T., Bergstra, J., Hunsberger, E., DeWolf, T., Steward, T.C., Rasmussen, D., Choo, X., Voelker, A.R. and Eliasmith, C. (2014) Nengo: a Python tool for building large-scale functional brain models. Front. Neuroinform. doi: 10.3389/fninf.2013.00048

Blouw, P., Choo, X., Hunsberger, E., and Eliasmith C. (2018) Benchmarking keyword spotting efficiency on neuromorphic hardware. arXiv:1812.01739v2 **[cs.LG]**

Brighton, C.H., and Taylor, G.K. (2019) Hawks steer attacks using a guidance system tuned for close pursuit of erratically manoeuvring targets. Nature Communications 10: 2462.

Brighton, C.H., Thomas, A.L.R., and Taylor, G.K. (2017) Terminal attack trajectories of peregrine falcons are described by the proportional navigation guidance law of missiles. Proc. Natl. Acad. Sci. USA 114: 13495-13500.

Chance, F.S. (2019) Dragonfly-inspired algorithms for intercept trajectory planning. Sandia Technical Report SAND2019-11695.

Chance F.S. (2020) Interception from a Dragonfly Neural Network Model. ICONS 2020: International Conference on Neuromorphic Systems 2020.

Combes, S.A. (2015) Dragonflies predict and plan their hunts. Nature 517: 279-280.

Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S.H., Dimou, G., Joshi, P., Inmam, N. et al. (2018) Loihi: a neuromorphic manycore processor with on-chip learning. IEEE Micro 38: 82-99.

Fabian, S.T., Sumner, M.E., Wardill, T.J., Rossoni, S., and Gonzalez-Bellido, P.T. (2018) Interception by two predatory fly species is explained by a proportional navigation feedback controller. J. R. Soc. Interface 15: p.20180466.

Frye, M.A., and Olberg, R.M. (1995) Visual receptive field properties of feature detecting neurons in the dragonfly. J. Comp. Physiol. A 177: 569-576.

Gonzalez-Bellido, P.T., Peng, H., Yang, J., Georgopolous, A.P. and Olberg, R.M. (2013) Eight pairs of descending visual neurons in the dragonfly give wing motor centers accurate population vector of prey direction. Proc. Natl. Acad. Sci. 110: 696-701.

Mischiati, M., Lin, H-T., Herole, O., Imler, E., Olberg, R., and Leonardo, A., (2015) Internal models direct dragonfly interception steering. Nature 517: 333-338.

Olberg, R.M. (1986) Identified target-selective visual interneurons descending from the dragonfly brain. J. Comp. Physiol. A Neuroethol. Sens. Neural. Behav. Physiol. 159: 827-840.

Olberg, R.M., Seaman, R.C., Coats, M.I., and Henry, A.F., (2007) Eye movements and target fixation during dragonfly prey-interception flights. J. Comp. Physiol. A. 193: 685-693.

Olberg, R.M., Worthington, A.H., Fox, J.L., Bessette, C.E., and Loosemore, M.P. (2005) Prey size selection and distance estimation in foraging adult dragonflies. Journal of comparative physiology A 191: 791-797.

Olberg, R.M., Worthington, A.H., and Venator, K.R., (2000) Prey pursuit and interception in dragonflies. J. Comp. Physiol. A. 186: 155-162.

Salinas, E., and Abbott, L.F. (1995) Transfer of coded information from sensory to motor networks. Journal of Neuroscience 15: 6461-6474.

Zipser, D. and Andersen, R.A. (1988) A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons. Nature 331: 679-684.

ADDENDUM A "INTERCEPTION FROM A DRAGONFLY NEURAL NETWORK MODEL"

Full reference: Chance F.S. (2020) Interception from a Dragonfly Neural Network Model. ICONS 2020: International Conference on Neuromorphic Systems 2020. doi: 10.1145/3407197.3407218

Interception from a Dragonfly Neural Network Model

Frances S. Chance
fschanc@sandia.gov
Sandia National Laboratories
Albuquerque, NM

ABSTRACT

While dragonflies are well-known for their high success rates when hunting prey, how the underlying neural circuitry generates the prey-interception trajectories used by dragonflies to hunt remains an open question. I present a model of dragonfly prey interception that uses a neural network to calculate motor commands for prey-interception. The model uses the motor outputs of the neural network to internally generate a forward model of prey-image translation resulting from the dragonfly's own turning that can then serve as a feedback guidance signal, resulting in trajectories with final approaches very similar to proportional navigation. The neural network is biologically-plausible and can therefore can be compared against *in vivo* neural responses in the biological dragonfly, yet parsimonious enough that the algorithm can be implemented without requiring specialized hardware.

CCS CONCEPTS

• **Computing methodologies → Motion path planning**.

KEYWORDS

guidance, interception, insect vision, neuromorphic

ACM Reference Format:
Frances S. Chance. 2020. Interception from a Dragonfly Neural Network Model. In *International Conference on Neuromorphic Systems 2020 (ICONS 2020), July 28–30, 2020, Oak Ridge, TN, USA*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3407197.3407218

1 INTRODUCTION

The field of neuromorphic computing is founded on the assumption that better understanding of neural systems and how they function can be leveraged to create more advanced computing systems. This study focused on a highly specialized nervous system, the neural circuitry underlying prey interception in the dragonfly. In nature dragonflies are highly successful hunters (with a 90-95% success rate [3, 11]). What key computations underlie the robustness of dragonfly hunting and how easily can the dragonfly system be translated to a man-made platform? This study seeks to contribute to the advancement of neuromorphic computing by constructing a computational model of the dragonfly nervous system that is framed at a level amenable for translation to a neuromorphic platform.

ICONS 2020, July 28–30, 2020, Oak Ridge, TN, USA

ACM ISBN 978-1-4503-8851-1/20/07…$15.00
https://doi.org/10.1145/3407197.3407218

Dragonflies were of particular interest for this study not only because of their high success rate, but also because they are known to use proportional navigation guidance as they approach their prey. Proportional navigation is a guidance law that results in the geometrically shortest path to interception. Also, dragonflies demonstrate a remarkably short latency when responding to prey maneuvers – on the order of 50 ms [7, 9], quite remarkable given that the response time constant of a single neuron is on the order of tens of ms. While a number of animal species (including dragonflies) are known to use proportional navigation (see [1, 4, 7] for reviews), there are certain advantages to studying an insect system, including the assumption that the underlying circuitry is likely to be 'light' (and therefore the validated model could be translated to a manmade system with relative ease).

As dragonflies approach their prey, they adjust their head position to maintain the image of its prey (referred to here as the 'prey image') on a specific part of the eye [9] (referred to as the fovea) through behavior known as foveation. While simply maintaining a constant angle between the dragonfly's direction of movement and its line-of-sight to the prey will result in behavior known as 'classical pursuit' (during which the dragonfly will head directly at its prey at all times) or a variant known as 'deviated pursuit' (in which a constant but non-zero angle will be maintained between the dragonfly's direction of flight and its line-of-sight to the prey) and therefore is not sufficient to produce proportional navigation. I have developed a model of dragonfly prey interception that executes proportional navigation solely based upon prey-image translation across the eye. While prey-image slippage away from the fovea has been suggested as the signal used by dragonflies for interception [6, 11], this is the first model (to the author's knowledge) of how that signal is used. This model is in the form of a neural network and incorporates certain simplifications intended to facilitate translation to a man-made system. Nevertheless, I will discuss model predictions that can be directly tested in the nervous system of the biological dragonfly.

2 MODELING APPROACH

While dragonflies have two eyes, the neural circuits thought to process moving targets and underlie dragonfly tracking of prey [5, 6, 8] largely do not have binocular receptive fields (although see [10], suggesting that dragonflies do not use depth perception to capture prey. Accordingly, the 'eyes' of the dragonfly model presented here are simplified as a flat two-dimensional screen (referred to here as the model dragonfly's eye). During each simulation time step, the movement of the prey relative to the dragonfly and the resulting translation of the image of the prey on the dragonfly eye (the prey image) are calculated. The dragonfly then adjusts its pitch and yaw angles (for simplicity, this study did not include roll), to maintain the prey image directly on the fovea. If the dragonfly approaches to within a minimum distance of the prey (specifically the distance that that the dragonfly can move within one simulation time step), a successful capture is declared and the engagement ends.

One significant difference between the model dragonfly and the biological dragonfly is that the fovea of the model dragonfly eye is moveable – its location on the eye is a function of the previous turns required to maintain the position of the prey image on the fovea. As described below, the model adjusts the position of the fovea to execute proportional navigation. By comparison, the fovea of the biological dragonfly eye is immoveable, although it is likely that proprioceptive information from the neck (encoding the angle of the head relative to the body) performs an analogous function.

Simulating the motor system (e.g. wings, muscles) was outside of the scope of this project, as was detailed simulation of the dragonfly eyes. For the results presented here, it is assumed that the dragonfly and its prey fly at the same speed (10 m/s) and have the same maneuverability.

2.1 Calculation of model dragonfly turning

The turning required to maintain the prey image on the fovea is calculated by a neural network of continuous-valued (non-spiking) neurons. Neurons in the 'prey-image representation' population (denoted by open circles in Figure 1) encode the position of the image of the prey on the eye (in 'eye coordinates'). The response f_i of each neuron i from this population is determined by a Gaussian tuning curve:

$$f_i(x_1, x_2) = \exp\left(-\frac{(\mathrm{a_{i1}} - \mathrm{x_1})^2 + (\mathrm{a_{i2}} - \mathrm{x_2})^2)}{2\sigma_\mathrm{r}^2}\right),$$

where (x_1, x_2) is the location of the prey image on the eye, (a_{i1}, a_{i2}) is the preferred position of the prey-image for neuron i, and σ_r determines the width of the tuning curve.

Neurons in the 'fovea-position representation' (indicated by filled blue circles in Figure 1) encode the position of the fovea in eye-coordinates. The response (g_j) of neuron j within this population is also determined by a Gaussian tuning curve:

$$g_j(y_1, y_2) = \exp\left(-\frac{(\mathrm{b_{j1}} - \mathrm{y_1})^2 + (\mathrm{b_{j2}} - \mathrm{y_2})^2)}{2\sigma_\mathrm{g}^2}\right),$$

where (b_{j1}, b_{j2}) is the preferred fovea location for the neuron, (y_1, y_2) is the fovea location, and σ_g describes the width of the tuning curve. These two inputs are combined in the sensory representation (red circles in Figure 1) such that the response S_{ij} of a sensory representation neuron multiplicatively combines input from one prey-image neuron (i) and one fovea-position neuron (j) :

$$S_{ij} = f_i(x_1, x_2) g_j(y_1, y_2).$$

The sensory representation is designed such that all possible combinations of prey-image position and fovea position neurons are included.

Neurons in the motor output population (green circles) represent the goal direction, in eye coordinates, to which the dragonfly should turn. The response of neuron i in the motor output population, R_i, is determined by summing over all inputs in the sensory representation, weighted by an appropriate factor (W_{ij}):

$$R_i = \sum_j W_{ij} S_j.$$

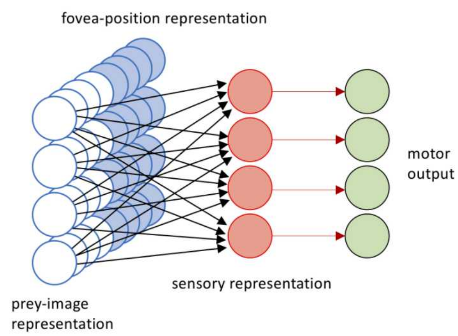


Figure 1: Schematic of the model dragonfly neural network. Open circles are prey-image neurons, and filled blue circles are fovea-position neurons. The responses of neurons in the sensory representation (filled red circles) arise from multiplicative interactions between neurons in the prey-image and fovea-position representations. The motor output population (green circles) encodes the direction that the dragonfly should turn (see text). For clarity, some neurons and connections between neurons are not drawn.

All network weights are calculated based upon the prey-image position, fovea position, and goal-direction preferences of the presynaptic and postsynaptic neurons (the neural network is not trained).

It is assumed that the motor output neurons are characterized by some "inherent" response tuning that determines the preferred goal location c of each motor output neuron. The inherent response m_i of neuron i of the motor output representation is:

$$m_i(z_1, z_2) = \exp\left(-\frac{(\mathrm{c_{i1}} - \mathrm{z_1})^2 + (\mathrm{c_{i2}} - \mathrm{z_2})^2)}{2\sigma_\mathrm{m}^2}\right),$$

where (c_{i1}, c_{i2}) is the preferred goal direction, and (z_1, z_2) is the direction of turn. It should be noted that, while there is some assumed inherent tuning of the motor output neuron σ_m, in practice σ_r and σ_g play a dominant role in determining the specificity of the motor output neurons.

The weight from sensory representation neuron j to motor output neuron i is given by:

$$W_{ij} \propto \int\int\int\int dy_1 dy_2 dz_1 dz_2 f_j(a_{j1} - (z_1 + y_1), a_{j2} - (z_2 + y_2)) \\ g_j(b_{j1} - y_1, b_{j2} - y_2) m_i(c_{i1} - z_1, c_{i2} - z_2)$$

where f_j and g_j are the prey-image representation and fovea-position representation tuning curves. Goal direction are expressed in eye coordinates, and because the eye is fixed relative to the body, this is equivalent to expressing goal direction in body-coordinates, relative to the reference frame of the dragonfly's body. The biological motivation to the pattern of connectivity is inspired by published models of coordinate transformations in parietal cortex [12, 13].

To determine dragonfly turning, the motor output representation is decoded through a neural-activity weighted average of the preferred directions of the motor population. The motor output activity is first thresholded (activity below a certain threshold is set to zero), then the direction and magnitude of turn is decoded as:

$$d_1 = \frac{\sum_i c_{i1} R_i}{\sum_i R_i} \text{ and } \mathrm{d}_2 = \frac{\sum_\mathrm{i} \mathrm{c}_{\mathrm{i}2} \mathrm{R}_\mathrm{i}}{\sum_\mathrm{i} \mathrm{R}_\mathrm{i}},$$

where (d_1, d_2) is the change in direction (in eye-coordinates) that the dragonfly executes. Expressed using terms more typically used to describe turns by airborne vehicles, the change in yaw is $\Delta\theta = \tan^{-1} \frac{d_1}{\epsilon}$, and the change in pitch is $\Delta\phi = \tan^{-1} \frac{d_2}{\epsilon}$, where ϵ is the distance from the dragonfly's eye to the center of the dragonfly's head, defined as the intersection of the yaw and the pitch axes.

If the fovea is held at a fixed position, the model dragonfly will display behavior known as 'classical pursuit' (if the fovea is at the center of the eye, see top panel of Figure 2). For this figure, the prey constantly travels in one direction only. During classical pursuit, the pursuer heads directly at the prey at all times. While this strategy for hunting can be successful, there is a tendency for the pursuer to end up in a tail chase (as is the case for this figure) in which the dragonfly falls directly behind the prey and fails to capture the prey (because both dragonfly and prey are moving at the same speed, if the dragonfly is directly behind the prey it is impossible for a capture to occur if the prey does not turn). For this engagement, the simulation was ended after 15 seconds of simulation time. A variant of classical pursuit known as 'deviated pursuit', in which the pursuer maintains a constant angle between the line-of-sight to the prey and its direction of motion, is given in the bottom panel of Figure 2). Depending on the location of the fovea, deviated pursuit can be successful for engagements in which classical pursuit is not (for comparison, the classical pursuit trajectory from the top panel is replotted in cyan). If complete information about the trajectory of the prey is known, deviated pursuit produces a trajectory equivalent to proportional navigation (see green trajectory in bottom panel) but this requires accurate calculation of the fovea position based upon complete knowledge of the prey's trajectory (because the prey does not turn in Figure 2, the correct fovea location to produce proportional navigation only needs to be computed at the start of the engagement). If prey velocity and position are not known, or if there is an error in calculation, deviated pursuit will also end in a tail-chase (not shown).

It is well-known that classical pursuit (or deviated pursuit) arises from holding the prey image at a fixed location on the eye, provided that the eye and head are held at a fixed angle relative to the body (for reviews see [2] [4]). These results are presented as a demonstration of the viability of the above-described neural network for generating trajectories driven by foveation. The generated trajectories are very similar to previously presented results from a similar model of dragonfly interception in which the dragonfly's turns were analytically (see Chance, presentation at ICONS2019).

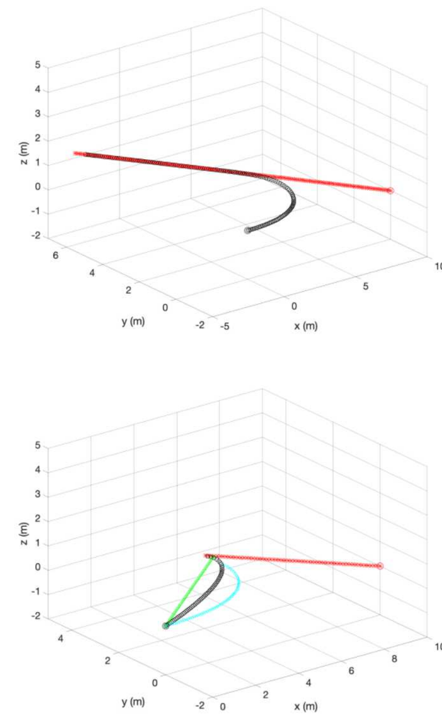


Figure 2: Variations of pursuit behavior arise from a fixed fovea position. The prey follows the same straight-line trajectory for both panels. Red and black circles indicate positions of prey and dragonfly, respectively, for each time step (larger red and black circles indicate starting positions). Top: The model dragonfly demonstrates classical pursuit behavior when the fovea is fixed at the center of the eye. The simulation ended after 15 seconds with no capture. Bottom: Deviated pursuit arises from an off-center but fixed fovea position. In this particular engagement, the model dragonfly was successful. The classical pursuit trajectory from above is replotted in cyan for comparison. The green trajectory is a special case of deviated pursuit in which the fovea position that supports a straight-line trajectory was pre-calculated.

2.2 Generating proportional navigation through forward model generated from motor output

Recent work [7] has suggested that dragonflies utilize internal models both to compensate for prey-image drift on the eye resulting from dragonfly-body rotations. Here we propose that dragonflies may use internally-generated forward models of prey-image translation on the eye as a feedback signal for generating proportional navigation. In this version of the model, the decoded motor output (d_1, d_2) is used not only to calculate the required trajectory, but is also used to determine changes to the location of the fovea on the eye. It should be noted that while this signal is easily decoded from the motor commands in a biological signal, the fovea of a biological eye is hardwired and cannot be moved. Instead, it is likely that the biological dragonfly adjusts other variables, for example head position relative to the body. For this version of the model dragonfly, the new fovea location is now $(e_1', e_2') = (e_1 - Qd_1, e_2 - Qd_2)$, where $e_1, e_2)$ is the former position of the fovea, (d_1, d_2) is the change in direction decoded from the motor output population, and Q is a gain factor. For the results shown here, $Q = 1$.

For these conditions ($Q = 1$), the fovea is essentially shifted in an equal but opposite direction to the prey-image translation on the eye. The behavior of the model with this additional component is shown in Figure 3. As in Figure 2, the prey is indicated in red and the dragonfly in black. The initial conditions in the top panel are identical to the top panel in Figure 2. The classical pursuit trajectory (from the top panel of Figure 2 except that the trajectory is truncated when the model dragonfly captures the prey) is provided in blue for comparison. Likewise, the geometrically shortest trajectory to capture the target (assuming full knowledge of target direction, speed, etc.), equivalent to following pure proportional navigation, is provided in green for comparison.

The initial conditions (including fovea position) are identical to those at the top of Figure 2, and the model dragonfly initially chases using classical pursuit (compare the dragonfly's early locations to the blue trajectory). However, as the dragonfly uses the feedback signal generated by the motor outputs to adjust the location of the fovea, the dragonfly's trajectory becomes more like proportional navigation (compare later dragonfly trajectory with green trajectory). This behavior is very similar to the previous version of the dragonfly model (see presentation by Chance at ICONS2019) in which required turns were analytically calculated.

It should be noted that proportional navigation generates the geometrically shortest trajectory to interception. If the prey does not turn, as in the top panel of Figure 3, the resulting trajectory is a straight line. If the fovea is at an appropriate location such that the dragonfly is is following proportional navigation while maintaining the prey image on the fovea, the prey image will remain aligned with the fovea for the remainder of the engagement until the prey is captured or turns. Thus, prey-image slippage away from the fovea is an appropriate "error" signal that could be used to search for a more optimal trajectory. It is likely that the feedback gain, Q, could be adjusted for more optimal trajectory calculation, in particular for engagements where the prey is turning or actively evading.

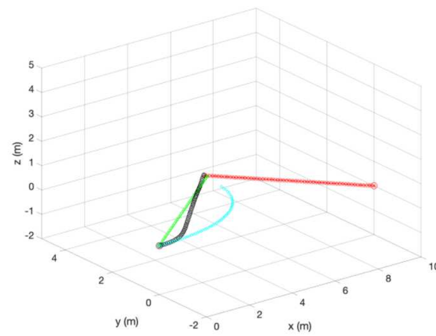


Figure 3: The model dragonfly follows trajectories closer to proportional navigation when an internal forward model of motor commands is used as a feedback signal to adjust fovea location. Red and black circles indicate positions of prey and dragonfly, respectively, for each time step. Large red and black circles indicate starting positions of prey and dragonfly. Blue trajectory is classical pursuit and green trajectory is proportional navigation. The initial conditions are identical to the top panel of Figure 2. The prey follows the same straight-line trajectory as in Figure 2 but here the prey is captured at approximately 5.6 seconds.

3 CONCLUSION

I have presented a model of dragonfly prey interception that calculates motor commands for prey-interception trajectories using a simple neural network. Specifically, the model uses visual and a proxy for proprioceptive input to determine turning commands that will align the prey image with the eye's fovea. The model also uses a feedback error signal to adjust approach trajectories to be more like proportional navigation. The feedback signal is a forward model of prey-image translation resulting from dragonfly turning generated from the model's motor outputs.

Certain simplifications were made to the dragonfly model to make the model more amenable to translation to a nonbiological system. For example, the dragonfly eyes are approximated as single two-dimensional screen, and a moving fovea is used in place of a pivotable head. Research is currently in progress to further develop the dragonfly model, in particular to develop methods for directly comparing the dragonfly model to data from the biological dragonfly.

ACKNOWLEDGMENTS

This work was supported by Laboratory Directed Research and Development program at Sandia National Laboratories. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

REFERENCES

[1] Caroline H Brighton and Graham K Taylor. 2019. Hawks steer attacks using a guidance system tuned for close pursuit of erratically manoeuvring targets. *Nature communications* 10, 1 (2019), 1–10.

[2] Caroline H Brighton, Adrian LR Thomas, and Graham K Taylor. 2017. Terminal attack trajectories of peregrine falcons are described by the proportional navigation guidance law of missiles. *Proceedings of the National Academy of Sciences* 114, 51 (2017), 13495–13500.

[3] Stacey A Combes. 2015. Neuroscience: dragonflies predict and plan their hunts. *Nature* 517, 7534 (2015), 279–280.

[4] Samuel T Fabian, Mary E Sumner, Trevor J Wardill, Sergio Rossoni, and Paloma T Gonzalez-Bellido. 2018. Interception by two predatory fly species is explained by a proportional navigation feedback controller. *Journal of The Royal Society Interface* 15, 147 (2018), 20180466.

[5] Mark A Frye and Robert M Olberg. 1995. Visual receptive field properties of feature detecting neurons in the dragonfly. *Journal of Comparative Physiology A* 177, 5 (1995), 569–576.

[6] Paloma T Gonzalez-Bellido, Hanchuan Peng, Jinzhu Yang, Apostolos P Georgopoulos, and Robert M Olberg. 2013. Eight pairs of descending visual neurons in the dragonfly give wing motor centers accurate population vector of prey direction. *Proceedings of the National Academy of Sciences* 110, 2 (2013), 696–701.

[7] Matteo Mischiati, Huai-Ti Lin, Paul Herold, Elliot Imler, Robert Olberg, and Anthony Leonardo. 2015. Internal models direct dragonfly interception steering. *Nature* 517, 7534 (2015), 333.

[8] Robert M Olberg. 1986. Identified target-selective visual interneurons descending from the dragonfly brain. *Journal of Comparative Physiology A* 159, 6 (1986), 827–840.

[9] Robert M Olberg, RC Seaman, MI Coats, and AF Henry. 2007. Eye movements and target fixation during dragonfly prey-interception flights. *Journal of comparative physiology A* 193, 7 (2007), 685–693.

[10] Robert M Olberg, Andrea H Worthington, Jessica L Fox, CE Bessette, and Michael P Loosemore. 2005. Prey size selection and distance estimation in foraging adult dragonflies. *Journal of comparative physiology A* 191, 9 (2005), 791–797.

[11] Robert M. Olberg, Andrea H Worthington, and Kurt R Venator. 2000. Prey pursuit and interception in dragonflies. *Journal of Comparative Physiology A* 186, 2 (2000), 155–162.

[12] Emilio Salinas and LF Abbott. 1995. Transfer of coded information from sensory to motor networks. *Journal of Neuroscience* 15, 10 (1995), 6461–6474.

[13] David Zipser and Richard A Andersen. 1988. A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons. *Nature* 331, 6158 (1988), 679–684.

ADDENDUM B "END-OF-YEAR POWERPOINT PRESENTATION"

Neuromorphic Processing and Sensing

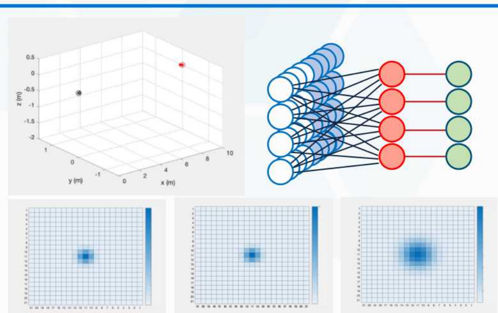
Project #218973, PI: Frances Chance

Purpose, Approach, and Goal

Purpose: leverage two existing LDRD efforts to demonstrate a proof-of-concept event-driven neuromorphic interception system

Approach: implement a neural-inspired interception algorithm ("Dragonfly-inspired interception, FY19 A4H LDRD) on neuromorphic hardware

Goal: evaluate potential efficiency gains from a system incorporating neuromorphic event-driven sensing array (SPARR) with a neuromorphic implementation of processing for interception (dragonfly on Intel Loihi)



Key R&D Results and Significance

Summary: Dragonfly-inspired interception algorithm was successfully converted to a neural network and a Loihi implementation was developed using the Nengo software package

Result: Dragonfly interception algorithm can be implemented on a spiking neuromorphic platform, but additional research is recommended to determine optimal implementation

Lessons Learned:

- "best practices" for developing neuromorphic implementations are not established (this was underestimated)
- If I had to do this again, I would spend more time devising alternate strategies for neuromorphic implementations earlier in the LDRD

Key Publication

Chance F.S. (2020) Interception from a Dragonfly Neural Network Model. ICONS 2020: International Conference on Neuromorphic Systems 2020. *(describes neural network architecture)*

1

Neuromorphic Processing and Sensing

Project #218973, PI: Frances Chance

R&D Summary

Going into this LDRD

In general, neuromorphic implementations excel when communication is event-driven and computations are simple and parallelizable. The dragonfly interception algorithm existed in an analytical form.

What we achieved

Dragonfly interception algorithm was first framed as a continuous-valued neural network (see ICONS 2020 conference paper). This described the general architecture that we then used for the spiking neural network version.

We have demonstrated that a spiking version of the dragonfly neural network (in principle) can be implemented on neuromorphic hardware (L. Parker, using NENGO software (ensembles) to implement the dragonfly algorithm on the Intel Loihi platform). However, neuromorphic implementation is not mature enough to warrant integration with event-driven neuromorphic sensing.

Lessons Learned:

- "best practices" for developing neuromorphic implementations are not established (this was underestimated)
- If I had to do this again, I would spend more time gaining expertise on developing for neuromorphic hardware earlier in the LDRD – circumstances made it difficult to connect to subject matter experts later in the LDRD
- The knowledge learned will facilitate future algorithmic implementation on neuromorphic hardware

Next steps (not part of this LDRD):

- Develop a better implementation of multiplication for spiking neuromorphic platform
- This will facilitate further development of the neuromorphic implementation of dragonfly-inspired interception

Neuromorphic Processing and Sensing

Project #218973, PI: Frances Chance

Project Metrics

Publications

Chance, F.S. (2020) A biologically-inspired approach to interception. EDFA Magazine 2: 16-21.

Chance F.S. (2020) Interception from a Dragonfly Neural Network Model. ICONS 2020: International Conference on Neuromorphic Systems 2020.

Chance, F.S. (submitted) An adaptive model of Dragonfly Interception. Frontiers in Computational Neuroscience.

Chance, F.S. (invited manuscript, delayed by COVID) The computer bug you want. IEEE Spectrum.

Presentations – External

"Interception from a Dragonfly Neural Network Model" at the International Conference on Neuromorphic Systems 2020 (virtual due to COVID-19 pandemic) July 28, 2020.

Presentations – Internal

"Dragonfly-Inspired Approaches to Interception", invited presentation at the 2019 Sandia National Laboratories Fall Leadership Forum (Albuquerque, NM, November 12, 2019)

"Dragonfly-Inspired Approaches to Interception", invited presentation at the 2020 Sandia National Laboratories Emeritus Event (Albuquerque, NM, March 3, 2020)

"Not all computer bugs are bad: What can we learn from insects for neural-inspired computing?", invited seminar at the Sandia National Laboratories New Research Ideas Forum (Albuquerque, NM, March 10, 2020)

Staff Development

This LDRD supported Luke Parker (summer intern who has transitioned to a full-year intern) for a portion of the summer while he worked on the neuromorphic implementation of the dragonfly neural network

Neuromorphic Processing and Sensing

Project #218973, PI: Frances Chance

Project Legacy

Key technical accomplishment

- We have successfully cast the dragonfly-inspired interception algorithm as a neural network that can be implemented on a neuromorphic platform. (see Chance F.S. Interception from a Dragonfly Neural Network Model. ICONS 2020).

How does this engage Sandia missions?

- This work is aligned with the A4H mission campaign and NSP (SDP/Hypersonics Focus Area)

Plans for follow-on

- Will investigate options for support to increase the maturity of the neuromorphic implementation so that robustness, SWaP requirements and other potential performance advantages can be assessed
- Collaboration with 6530 to address potential issues interfacing dragonfly algorithm with "real world" vehicles
- Work in progress to evaluate dragonfly algorithm against proportional navigation

What I wish I could have done but didn't: deep thinking/research towards a more optimal implementation of the dragonfly algorithm on a neuromorphic platform