A Spiking Model of Desert Ant Navigation Along a Habitual Route

Przemyslaw Nowak[1(✉)] and Terrence C. Stewart[2]

[1] Institute of Information Technology, Lodz University of Technology, Lodz, Poland
przemyslaw.nowak@p.lodz.pl

[2] Centre for Theoretical Neuroscience, University of Waterloo, Waterloo, ON, Canada
tcstewar@uwaterloo.ca

Abstract. A model producing behavior mimicking that of a homing desert ant while approaching the nest along a habitual route is presented. The model combines two strategies that interact with each other: local vector navigation and landmark guidance with an average landmark vector. As a multi-segment route with several waypoints is traversed, local vector navigation is mainly used when leaving a waypoint, landmark guidance is mostly used when approaching a waypoint, and a weighted interplay of the two is used in between waypoints. The model comprises a spiking neural network that is developed based on the principles of the Neural Engineering Framework. Its performance is demonstrated with a simulated robot in a virtual environment, which is shown to successfully navigate to the final waypoint in different scenes.

Keywords: Spiking neural networks
Neural Engineering Framework · Insect navigation

1 Introduction

Desert ants are solitary foragers, which exhibit amazing navigational skills [15]. When searching for food, they can go as far as 200 m away from their nest in relatively featureless terrain and then reliably return home [9]. Unlike other ants, however, these ants do not use pheromones to mark their trails because due to the high desert temperatures, those would evaporate too quickly. Consequently, despite their miniature brains, desert ants have developed a number of sophisticated mechanisms, including path integration and landmark guidance, to meet their navigational demands [9,15]. Moreover, they utilize different strategies depending on whether they are in familiar or unfamiliar terrain and whether the environment is mostly featureless or cluttered [14], which altogether has made them become model organisms for studying insect navigation [15].

Although much has been discovered about these mechanisms and strategies in recent years, the neural substrate is still largely unknown [1,9]. Here, we present a model using a spiking neural network (SNN) developed based on the principles of the Neural Engineering Framework (NEF), which produces behavior

R. Matoušek (Ed.): MENDEL 2017, AISC 837, pp. 211–222, 2019.
https://doi.org/10.1007/978-3-319-97888-8_18

similar to that of a homing ant approaching the nest along a habitual route. We demonstrate the performance of the model using a simulated robot in a virtual environment.

The rest of the paper is structured as follows: Sect. 2 provides an overview of desert ant navigation, Sect. 3 introduces the Neural Engineering Framework, the proposed model is described in Sect. 4 and preliminary results obtained from the model are presented in Sect. 5, and the final Sect. 6 is devoted to discussion and conclusions.

2 Desert Ant Navigation

Desert ants employ several strategies for navigation: path integration (also termed vector navigation), landmark guidance, and systematic search [15]. These strategies are utilized adaptively depending on the actual circumstances and interact with one another [15]. Path integration requires combining compass information, which is mainly provided by the polarization compass mechanism, with odometer information, which is mostly based on the pedometer mechanism (i.e., counting steps) and to a much lesser extent exploits optic flow [15]. Importantly, vector navigation can be broken down into global and local vector navigation: the global vector, obtained through integration of directions taken and distances covered since leaving the nest, maintains a continuously updated estimate of the direction and distance to the nest from the current location, whereas local vectors are associated with particular locations in the environment and store direction to the next waypoint on the route [6]. The next strategy, landmark guidance, involves "labeling" places using visual landmarks observed at these places from particular vantage points [14]. Those stored views can be later recalled to guide the ant to the corresponding places, using view-matching mechanisms that are retinotopically organized [14]. Several methods for such view-matching mechanisms have been proposed, the snapshot model being the canonical one, and its various variants such as the average landmark vector model later derived from it [9]. In the snapshot model, the current retinal image is compared with a stored one in terms of apparent sizes and bearings of individual landmarks, without the need for actual identification of the landmarks themselves, and the resulting differences collectively determine movement aimed at reducing the mismatch such that the two images coalesce. The average landmark vector (ALV) model is a parsimonious version, in which it is not individual landmarks that are compared, but average vectors defined by those landmarks all together. In other words, bearings to all landmarks in the view are lumped into a single two-dimensional ALV and only the ALVs of the current and stored views are compared [9]. Finally, systematic search strategy is employed when the ant has arrived close to the nest using its path integrator, but due to the inevitable cumulative errors in integration it is not exactly where it should be and thus begins to systematically search the vicinity of its current location to find the nest entrance. This strategy is mostly exploited in featureless terrain, where landmarks and view-matching mechanisms cannot be used for precise guidance to the nest [15].

When traveling over larger areas in featureless terrain, desert ants can rely only on the global vector navigation; on the other hand, in cluttered environments navigation using landmarks can override the global vector navigation, although path integration is continuously carried out [14]. Routes that are followed many times can become habitual; in such cases ants learn sequences of landmark scenes along the route and associate them with remembered heading directions (i.e., local vectors) [5,6], effectively partitioning the route into separate successive segments that can point in different directions [6]. Recognizing a particular scene triggers the associated local vector, which guides the ant to the visual catchment area of the next waypoint on the route [14]. An ant can store in its memory a number of routes and retrieve these memories correctly according to the current context [15]. However, this traditional approach to navigation along habitual routes has recently been challenged by a different, "view-familiarity" model, which postulates that the purpose of the movement aimed at aligning the current view with a stored one is not to ensure arrival at the corresponding waypoint, but rather to orient the ant in the correct direction for forward motion that will follow; thus, in the "view-familiarity" approach local vectors are actually eliminated [2].

To explain how navigation in desert ants is organized from the computational perspective, a number of models have been proposed, including those addressing homing in on a single goal using view-matching methods based on local landmarks [9], those concerning long-range navigation employing waypoints linked in a sequence and approached using view-matching methods [11], those concerning long-range navigation without explicit waypoints and using "view-familiarity" approach along with rotational scanning [1,2], and those dealing with long-range navigation combining global and local vectors [7]; some of these models also implemented learning [1,2,11], utilized spiking neural networks [1], and were even applied to real robotic platforms [9,11].

3 Neural Engineering Framework

To create our SNN model, we used the Neural Engineering Framework [8]. This is a method for taking a high-level description of a desired algorithm and converting it into a set of spiking neurons that approximate that algorithm. In our case, we used standard Leaky Integrate-and-Fire neurons, connected with exponential synapses.

In the NEF, an algorithm is defined in terms of variables and functions on those variables. Variables are represented in a distributed manner by groups of neurons. For example, a group of 100 neurons may *encode* a 2-dimensional numerical value (such as the x and y coordinates of a local vector) in their firing pattern. Each neuron within this group has some particular preferred value for which it will fire the fastest. For the model discussed here, we choose these preferred stimuli (and thus the "tuning curve" for the neuron) randomly, but in future work it would be constrained based on neural data. The NEF also specifies the inverse operation, *decoding* variables, that is recovering their values from the observed firing patterns.

To implement a particular algorithm, the NEF forms connections between groups of neurons. Consequently, whenever two groups of neurons are connected, we define the numerical *function* that we want to have computed. For example, if one group of neurons represents x, y components of the velocity command and we have another group of neurons that we want to represent the steering angle θ, then we want to connect these two groups of neurons such that $\theta = tan^{-1}(y/x)$. The NEF provides a method for finding the synaptic connection weights between these two groups of neurons that best approximate this function. That is, if there are 100 neurons in the first group and 100 neurons in the second group, the NEF generates a 100×100 matrix such that, if the first group of neurons is stimulated with the pattern of activity for any x, y value, the second group will be driven to fire with the correct corresponding θ value.

Of course, these neurons will only *approximate* the desired function. In order to determine the actual behavior of the system, given this approximation (and given the variability caused by the spiking neuron activity itself), we need to run the neural system and observe its behavior. To do this, we use the software toolkit Nengo [3], which also includes a built-in implementation of the NEF methods.

Given this framework, everything in the model must be described as continuous real-valued variables and functions on those variables (The NEF also uses recurrent connections, which can approximate differential equations on those variables.) However, this means that it is not obvious how to implement discrete items in such a framework. For example, how can we have one group of neurons representing which waypoint the ant has reached most recently? For this, we take the approach of claiming that the neurons in this group represent some high-dimensional vector space (e.g., 32 dimensions), and we randomly choose points in that space to represent each waypoint. In other words, each waypoint is assigned a particular randomly chosen 32-dimensional vector. With this approach, symbol-like functions can be implemented with the NEF. For example, we can define a function where the input is the 32-dimensional vector representing which waypoint has been reached most recently, and the output is the next waypoint.

4 Model

Our SNN model is based on the classic approach that combines view-matching recognition of subsequent waypoints along a habitual route with expression of local vectors guiding to the visual catchment area of the successive waypoint. Rather than using the original snapshot model to implement the view-matching mechanism, we utilize the ALV model for this purpose. The SNN is coupled in a closed-loop manner to a simulated robot in a virtual environment and controls its movement. The robot model is based on the popular robot platform Pioneer P3-DX and is simulated using V-REP simulator [10].

In our model, only the critical parts of the whole system are implemented neurally, with the remaining components being computed numerically. In particular, we do not implement the neural circuitry of the ant compass and odometer

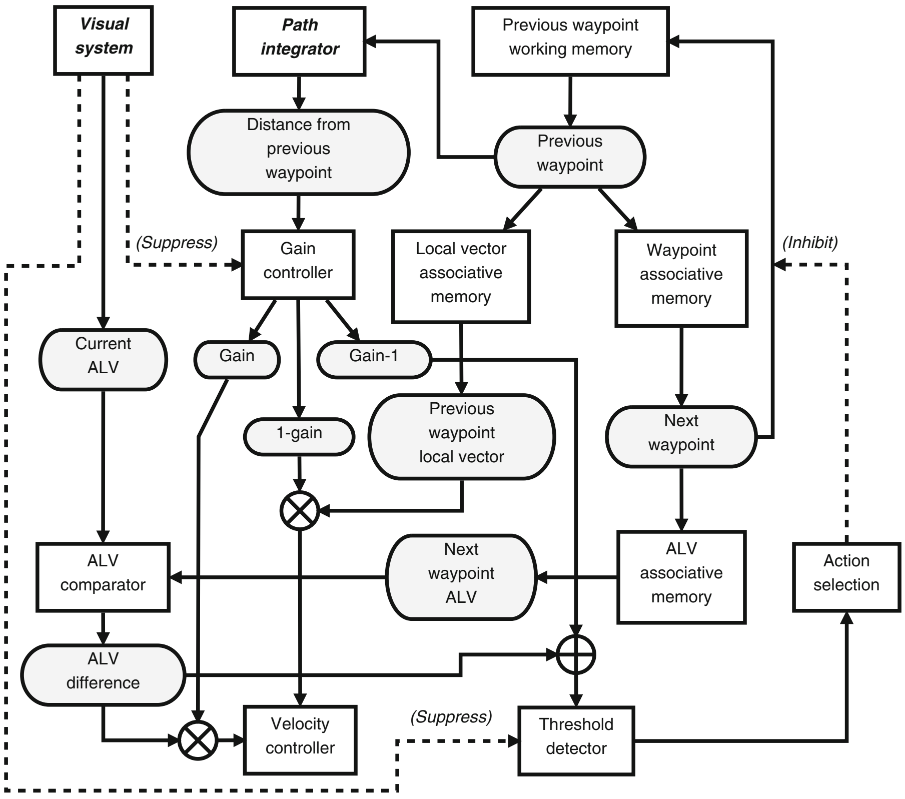

Fig. 1. Diagram of the model

mechanisms; instead, robot position and orientation are obtained directly from the simulator, while distance traveled from the previous waypoint is computed explicitly. Also, vision is not implemented, but substituted with direct calculations of ALVs based on the current robot pose and the position of landmarks within a predefined range. Finally, learning the habitual route is not considered; all the waypoints as well as the local vectors and ALVs associated with them are predetermined.

Figure 1 shows a diagram of the model. Rectangular boxes represent components of the system, while rounded boxes depict information that is computed by or transmitted between components. Bold italic font in rectangular boxes indicates that the corresponding components perform computations explicitly, to distinguish them from those that are implemented neurally.

There are two inputs to the system: vision and path integration. Vision obtains from the simulator positions of landmarks that are within the predefined range from the robot and calculates the currently perceived ALV in egocentric coordinates. Path integration obtains from the simulator robot current position and orientation and computes distance covered from the previously reached waypoint.

The neural part of the model can be divided into two subsystems, which partially overlap: one that controls robot velocity using local vector navigation and landmark guidance, and the other one that is responsible for keeping track of which waypoint has most recently been reached as well as detecting arrival at the next waypoint. The central element to both subsystems is the working memory that maintains a memory trace, represented in the form of a high-dimensional (HD) vector, of the previously reached waypoint.

In the case of the subsystem controlling robot velocity, information about the previously reached waypoint is delivered to three components: (1) the path integrator, so that it could properly keep track of the distance covered from the most recent waypoint; (2) the local vector associative memory, which stores associations between each waypoint on the route and the corresponding local vector leading from that waypoint to the next waypoint; and (3) the waypoint associative memory, which stores associations pairing each waypoint with its successor. The output of the waypoint associative memory, that is a HD vector representing the next waypoint, is subsequently conveyed to the ALV associative memory, which stores associations between each waypoint and the ALV observed at that waypoint. Consequently, this path in the model leads to retrieval of the ALV associated with the next waypoint, which is then supplied to the ALV comparator. The other input to this comparator is the current ALV that the robot is perceiving, which comes from the visual system, and thus the difference between the two ALVs is determined. At the same time, the local vector pointing to the next waypoint is obtained as the output from the local vector associative memory, and subsequently both the ALV difference and the local vector are combined with appropriate gain adjustments into a single velocity command in the velocity controller, which finally translates this command into motor commands for the robot wheels.

Gain adjustments applied to the ALV difference and to the local vector are a very important element in the model. Its purpose is to differentially weight contributions of the two signals depending on how far the robot has moved away from the previous waypoint and thus how close it has possibly come to the next one. Specifically, if the robot is near the previous waypoint, its velocity is mostly determined by the local vector associated with that previous waypoint and pointing to the next one; in turn, when the robot is far off from the previous waypoint and hence likely approaches the next one, it is the ALV difference that mainly dictates velocity. This design allows the robot to properly home in on the next waypoint while avoiding possibly detrimental confusion while going past a waypoint. Indeed, when the robot is leaving a waypoint that has just been reached in pursuit for the next one, landmarks belonging to that just reached waypoint are still visible and thus may confound the ALV difference, resulting in wrong velocity signals.

The other neural subsystem, which keeps track of which waypoint has most recently been reached and detects arrival at the next waypoint, shares some components with its counterpart controlling the robot velocity. Because detection of arrival at a waypoint is based on a threshold mechanism, the ALV difference determined by the ALV comparator is monitored by the threshold detector, which, when appropriate, triggers the action selection circuitry to update the working memory of the most recently reached waypoint. However, like in the case of velocity control, also in this process is the ALV difference subject to gain adjustment. The goal of this adjustment is to prevent another detrimental confusion that may arise when the ALV associated with the just-reached waypoint is very similar to that of the next waypoint. This confusion may occur because arrival at a waypoint makes the robot stop comparing the currently perceived ALV with that of the just-reached waypoint and start comparing it with that of the next waypoint, yet since at this moment the current ALV is almost identical to that of the just-reached waypoint, high similarity between the ALVs of the just-reached and of the next waypoints results in the current ALV being similar to that of the next waypoint as well. Consequently, the difference between the current ALV and the ALV of the next waypoint may happen to be below the threshold, which, if not adjusted, would in turn cause the robot to erroneously conclude that the next waypoint has also been reached, although the robot has even not left the previous one. To prevent such a misinterpretation, a negative gain is added to the ALV difference upon arrival at a waypoint, and then is progressively removed as the robot moves away from that waypoint until vanishing completely prior to approaching the next waypoint.

Because in vertebrates travel along a habitual route appears to be mediated by the basal ganglia [5], to implement the action selection circuitry we used the spiking model of the basal ganglia readily available in Nengo [12]. Obviously, the basal ganglia do not exist in ants, but since there is evidence suggesting that the central complex can be regarded as their homologue in arthropods [13], we adopted this approach as a first approximation. Consequently, the basal ganglia inhibit the connection between the waypoint associative memory and the working memory of the previous waypoint whenever the adjusted ALV difference is above the threshold, and only when this difference falls below the threshold does this connection become disinhibited. The result of disinhibition is that the output of the waypoint associative memory, that is the HD vector representing the next waypoint, which in this particular moment corresponds to the waypoint that has just been reached, overrides the current HD vector stored in the working memory of the previous waypoint, and thus appropriate update of the latter memory is achieved.

Finally, there are two additional suppression mechanisms employed in the model, which are activated when there are no landmarks in view. In such a case, the current ALV is undetermined, and therefore appropriate suppression of the gain controller cancels the contribution of the ALV difference to the velocity command, effectively making the robot steering depend only on the local vector, while similar suppression of the threshold detector prevents a possible false detection of arrival at the next waypoint.

5 Results

Figure 2 presents routes followed by the robot in two example scenes. Both routes consist of two segments, and therefore each scene contains three waypoints: the start one, which is in the upper right quadrant, the intermediate one, close to the middle, and the final one, close to the bottom. The start and the intermediate waypoints are depicted as orange circles, while the final waypoint is depicted as the red circle. The intermediate and final waypoints are surrounded by landmarks represented as green circles. Importantly, the robot does not know the locations of the waypoints; it only knows the ALVs associated with these waypoints and can perceive the landmarks. Moreover, at the beginning of simulations, it is not heading towards the intermediate waypoint, but is somewhat rotated to either side. The dark blue lines show the routes followed by the robot. As can be seen, the robot first goes to the intermediate waypoint, which requires an appropriate turn at the start location, and then turns accordingly and proceeds to the final waypoint. During the initial part of each segment of the routes, the robot is mainly driven by the corresponding local vector, while during the end part of each segment, it is mostly driven by the ALV difference. As a result, some minor turns appear in the second parts of the segments, which manifest corrections imposed by the ALV difference on the original directions taken at the beginning of the segments. Moreover, even if the robot does not home in precisely on the intermediate waypoint due to inherent noise in the SNN or imprecision of the ALV model, which can result in a direction slightly off from the optimal one during the beginning of the next segment, as is the case for the right scene, it can still reliably reach the final waypoint.

Some aspects of the model dynamics for the robot following the route in the right scene are presented in Fig. 3. The first plot shows distances to each waypoint over time. The second plot is a raster representing activity of a sample of 50 neurons in the working memory maintaining the trace of the previous waypoint. Evidently the pattern of activity changes when the robot reaches the intermediate waypoint. This is because the neuronal activity encodes the HD vector associated with the most recently reached waypoint, which changes upon arrival at the next waypoint. The value of this encoded HD vector can be decoded according to the NEF principles, and the similarity of the decoded value to the values of the actual HD vectors associated with each of the waypoints is visualized in the third plot, in which larger numbers correspond to higher similarity

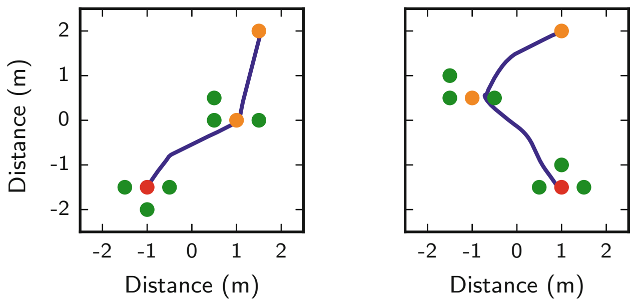

Fig. 2. Routes followed by the robot in two example scenes

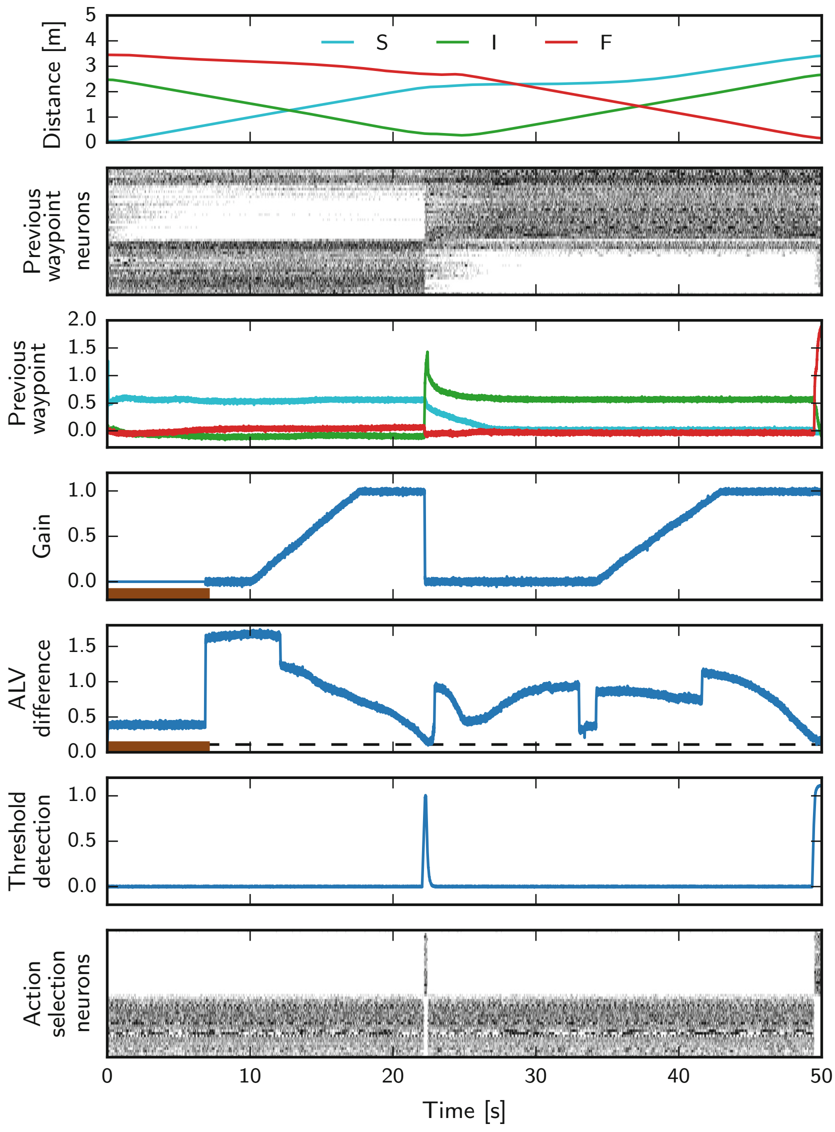

Fig. 3. Model dynamics when the robot travels along an example route (waypoints: S, start; I, intermediate; F, final)

(colors are the same as in the first plot). As expected, during the first segment of the route, the HD vector corresponding to the start waypoint is expressed, whereas during the second segment, the expressed HD vector corresponds to the intermediate waypoint. The similarity is not perfect, though, because the expressed HD vectors are noisy versions of the actual counterparts. The fourth plot shows modulation of the gain applied to the ALV difference. The gain, depicted as the blue line, equals 0 during the first part of the first segment of the route, then gradually increases to 1 in the second part of the segment, and maintains this value over the third part until the arrival at the intermediate waypoint, which begins the second segment of the route, resulting in the gain dropping to 0 and the whole cycle repeating. Moreover, whenever there are no landmarks in view, as in the case for the initial part of the route, which is indicated by the brown line at the bottom, the gain is forced to be 0 irrespective of its current value. In the fifth plot, the blue line represents the ALV difference, whereas the dashed line corresponds to the threshold at which arrival at the next waypoint is detected. During the initial part of the route with no landmarks in view, as indicated by the brown line, the ALV difference is small because the currently perceived ALV is a zero vector and the difference depends only on the magnitude of the ALV associated with the next waypoint. The ALV difference reaches the threshold around 23 s, and then increases rapidly as the ALV associated with the final waypoint, instead of that associated with the intermediate waypoint, starts to be used for its computation. Even though the ALV difference drops close to the threshold around 33 s, in the middle between the intermediate and final waypoints, there is no risk that arrival at the final waypoint could be falsely identified because the gain applied to the ALV difference is still 0 at that point. The sixth plot shows the output of the threshold detector: it is 0 for most of the time and displays a sudden peak only when the ALV difference reaches the threshold, that is when the intermediate and final waypoints are reached. The final, seventh plot, is a raster representing activity of a sample of 50 neurons in the action selection circuitry. As is evident, the pattern of activity of these neurons closely matches the time course of the output of the threshold detector.

6 Discussion

Our model of desert ant navigation provides a combination of ballistic navigation based on the local vector upon arrival at a waypoint, attractor navigation based on a simple view-matching mechanism when approaching a waypoint, and a weighted interplay of the two in between waypoints. Moreover, all crucial processes are implemented neurally using a SNN.

We have not implemented global vector navigation because experimental findings suggest that when an ant expresses a local vector, expression of the global vector is inhibited, which results in the global vector being ignored during navigation over a familiar, cluttered environment [6].

As a view-matching mechanism, we have adopted the ALV model, which, although parsimonious, has been successfully employed in several computational studies on desert ant navigation and even applied to real robots [9,11]. One problem with this model, however, is that it requires alignment of the perceived and stored ALVs to the common reference frame, such as the external compass reference, to allow their comparison, while experimental evidence suggests that in ants internal rotation of images can be excluded as can body rotation for this purpose [9]. A remedy would be to store a set of ALVs for a particular location, each corresponding to an image taken at a different orientation, which, in some circumstances, seems biologically plausible. In our model this problem does not exist because we further simplified the ALV model: ALVs associated with waypoints are stored only from the egocentric perspective of the approaching robot. This is because waypoints are maintained in a sequence and we assume that the local vector expressed at a previous waypoint should ensure approximately correct orientation of the robot while approaching the next waypoint, thus making rotations unnecessary.

Utilizing distance-dependent gain control applied to the ALV difference has made it possible to avoid the problem of perceptual aliasing, both when steering away from a previously reached waypoint and when detecting arrival at the next waypoint. It has also made our model consistent with experimental findings that in desert ants landmark memories are combined with vector information in a manner depending on how far along the route the ant is [4].

There are some limitations to our model as well. It suffers from the problems common to models based on thresholds [11], namely it is very sensitive to the actual choice of parameters, which so far had to be adjusted individually for each scene. Also, fine choice of neural time constants is required to achieve desired behavior.

In the future, the model could be extended in at least several ways: fixed thresholds could be replaced with some adaptive alternatives to make it more robust, more advanced view-matching mechanisms, such as the original snapshot model, could be adopted, and finally learning could be added.

References

1. Ardin, P., Peng, F., Mangan, M., Lagogiannis, K., Webb, B.: Using an insect mushroom body circuit to encode route memory in complex natural environments. PLoS Comput. Biol. **12**, e1004683 (2016). https://doi.org/10.1371/journal.pcbi.1004683
2. Baddeley, B., Graham, P., Husbands, P., Philippides, A.: A model of ant route navigation driven by scene familiarity. PLoS Comput. Biol. **8**, e1002336 (2012). https://doi.org/10.1371/journal.pcbi.1002336
3. Bekolay, T., Bergstra, J., Hunsberger, E., DeWolf, T., Stewart, T.C., Rasmussen, D., Choo, X., Voelker, A.R., Eliasmith, C.: Nengo: a Python tool for building large-scale functional brain models. Front. Neuroinform. **7**, 48 (2014). https://doi.org/10.3389/fninf.2013.00048
4. Bregy, P., Sommer, S., Wehner, R.: Nest-mark orientation versus vector navigation in desert ants. J. Exp. Biol. **211**, 1868–1873 (2008). https://doi.org/10.1242/jeb.018036
5. Collett, M.: How desert ants use a visual landmark for guidance along a habitual route. In: Proceedings of the National Academy of Sciences U.S.A., vol. 107, pp. 11638–11643 (2010). https://doi.org/10.1073/pnas.1001401107
6. Collett, M., Collett, T.S., Bisch, S., Wehner, R.: Local and global vectors in desert ant navigation. Nature **394**, 269–272 (1998)

7. Cruse, H., Wehner, R.: No need for a cognitive map: decentralized memory for insect navigation. PLoS Comput. Biol. **7**, e1002009 (2011). https://doi.org/10.1371/journal.pcbi.1002009
8. Eliasmith, C., Anderson, C.: Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems. MIT Press, Cambridge (2003)
9. Lambrinos, D., Möller, R., Labhart, T., Pfeifer, R., Wehner, R.: A mobile robot employing insect strategies for navigation. Rob. Auton. Syst. **30**, 39–64 (2000). https://doi.org/10.1016/S0921-8890(99)00064-0
10. Rohmer, E., Singh, S.P.N., Freese, M.: V-REP: a versatile and scalable robot simulation framework. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1321–1326, Tokyo (2013). https://doi.org/10.1109/IROS.2013.6696520
11. Smith, L., Philippides, A., Graham, P., Baddeley, B., Husbands, P.: Linked local navigation for visual route guidance. Adapt. Behav. **15**, 257–271 (2007). https://doi.org/10.1177/1059712307082091
12. Stewart, T.C., Choo, X., Eliasmith, C.: Dynamic behaviour of a spiking model of action selection in the basal ganglia. In: Salvucci, D.D., Gunzelmann, G. (eds.) Proceedings of the 10th International Conference on Cognitive Modeling, pp. 235–240, Philadelphia (2010)
13. Strausfeld, N.J., Hirth, F.: Deep homology of arthropod central complex and vertebrate basal ganglia. Science **340**, 157–161 (2013). https://doi.org/10.1126/science.1231828
14. Wehner, R.: Desert ant navigation: how miniature brains solve complex tasks. J. Comp. Physiol. A. **189**, 579–588 (2003). https://doi.org/10.1007/s00359-003-0431-1
15. Wehner, R.: The architecture of the desert ant's navigational toolkit (Hymenoptera: Formicidae). Myrmecol. News **12**, 85–96 (2009)