Insect-Inspired Visual Navigation On-Board an Autonomous Robot: Real-World Routes Encoded in a Single Layer Network

James C. Knight[1], Daniil Sakhapov[2], Norbert Domcsek[1], Alex Dewar[1], Thomas Nowotny[1] and Andrew Philippides[1]

[1]Centre for Computational Neuroscience and Robotics, University of Sussex, Brighton, UK
[2]Department of Computer Science, Tomsk State University, Tomsk, Russia
J.C.Knight@sussex.ac.uk

Abstract

Inspired by the behaviour of ants, models of visually-guided navigation that operate by scanning for previously experienced views of the world have been shown to be capable of robust route navigation. These algorithms operate by training an artificial neural network (ANN) to output a measure of the familiarity views experienced along a training route, hence we refer to them as familiarity-based navigation algorithms. In this paper we show that an ANN with an Infomax learning rule is capable of delivering reliable direction information, even when scenes contain few local landmarks and high-levels of noise (from changing lighting conditions, uneven terrain etc). Indeed, routes can be precisely recapitulated, with our robot straying an average of only $10\,\mathrm{cm}$ from the $6\,\mathrm{m}$ training path. Additionally, we show that the required computation does not increase with the number of training views and thus the ANN provides a compact representation of the knowledge needed to traverse a route. Finally, rather than this compact representation necessarily losing information, there are instances where the use of an ANN ameliorates the problems of sub-optimal paths caused by very tortuous training routes. Our results therefore suggest the feasibility of familiarity-based navigation for long-range autonomous visual homing.

1 Introduction

Visual homing – the ability to navigate back to a place of interest using visual information alone – is a problem of great interest for both engineers seeking to build autonomous robots and neuroethologists seeking to understand its neural basis in animals **(TODO: CITE ANDY AND PAUL?)**. In the animal kingdom, solitary foraging ants are one of the champion visual navigators (Wehner, 2009). Despite having small brains and low resolution vision, these ants can learn visually guided routes many metres long through complex terrain (Knaden and Graham, 2016).o However, in direct contrast with most modern robotic methods, to navigate between two locations, insects use route knowledge not mental maps (Wehner et al., 2006). That is, insects learn procedural instructions for navigation: "What should I do here?" rather than "Where am I?". This allows for much simpler mental representations of the visual world with corresponding potential for computational efficiencies. We have shown that route knowledge can be learnt and represented holistically using an artificial neural network (ANN) (Philippides et al., 2015), without specifying when or what to learn (Baddeley et al., 2011a) and from a single exposure to the route data (Baddeley et al., 2012). One of the reasons to use an ANN is that route knowledge can be encoded and used with memory and computational constraints that do not scale with route length, are plausible for an ant and, as a corollary, well-suited to a small, power-efficient, robot. In this paper we therefore explore how well a single layer ANN can guide a small robot through real world routes of up to $10\,\mathrm{m}$ with all compution being performed on-board.

Our route navigation algorithms start with two observations. Firstly, if an agent stores a view when facing a given direction, the difference between it and views from a nearby location, but at random headings, will be minimised when the agent is facing the same direction as when the first image was stored (Zeil et al., 2003). Secondly, for ants and many wheeled robots, there is a fixed relationship between viewing direction and direction of travel, meaning that a view implicitly defines a movement direction. Therefore, when an agent is facing in a familiar direction, it is likely travelling in the correct direction. This allows the problem of navigation to be reframed in terms of a search for familiar views, that is, views that are associated with a previously learned route. Based on this, we have developed a parsimonious insect-inspired navigation algorithm in which a route, or routes, are learnt holistically and route recapitulation is driven by a search for familiar views (Baddeley et al., 2012).

The algorithm proceeds as follows: an agent equipped with a low-resolution 360° panoramic visual sensor first travels a route. The views it experiences along this route – crucially determined by both the agents positions and headings (poses) – are used to sequentially train an artificial neural network (ANN) which learns a holistic representation of the views encountered. Subsequently, the network is used to estimate the likelihood of whether a given view – and thus a pose – has been experienced before. When trying to repeat the route, the agent derives a direction of movement at a position by visually scanning the environment (either

Figure 1: Robot platform with onboard computation.

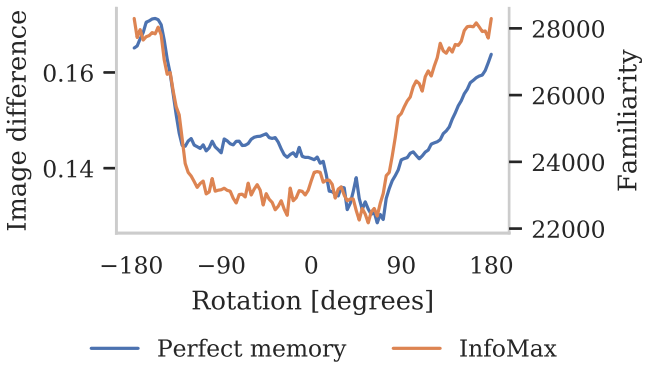

Figure 2: Example Rotational Image Difference Function (RIDF) from Perfect memory algorithm and Rotational Familiarity Function (RFF) from InfoMax.

by physically rotating – a behaviour seen in ants (Wystrach et al., 2014) – or rotating the view 'in silica'). Each rotated version of the current view is applied as an input to the network which outputs an estimate of its familiarity. The agent then moves in the direction corresponding to the view most similar to those encountered during learning.

Algorithms of this sort have been used to successfully learn routes through simulations of the habitats of solitary foraging ants, using a variety of neural networks as the route encoding, ranging from restricted Boltzmann machines (Baddeley et al., 2011b) to a spiking model of part of an ants brain known as the mushroom body (Ardin et al., 2016). The paths of these simulated agents have then been shown to demonstrate many of the characteristics of the paths of ants (Wystrach et al., 2013). Here, we follow Baddeley et al. (2012) who showed that a single layer network trained with an Infomax learning rule can not only navigate robustly, but can learn multiple paths to a single goal in a single trial with performance robust to sensor and motor noise. We choose to use this approach mainly because it only requires a single pass through the data, meaning that each view is experienced just once and then discarded. While the parsimony of encoding and recall makes this algorithm suitable for use on a small autonomous robot, it had not previously been tested in the real world. Here, we show that a robot can navigate fully autonomously outdoors using only a single layer neural network, with all processing performed onboard the robot using a Jetson TX1 (NVIDIA Corporation, 2016).

2 Methods

As described above, at the heart of our algorithm an agent navigates by sampling views from the current position at a number of headings and finding the direction that is deemed most familiar by comparison to the views perceived during a training route. We find this most familiar direction in two ways. Firstly, we train a neural network to output the familiarity of the training views and input rotated versions of the current view to it. This results in a rotational familiarity function (RFF – Figure 2) which, for each heading faced, gives a familiarity score with the heading with the highest score being the most familiar and the movement direction taken. Secondly, as a baseline and because it allows easier interpretation of the data (for instance, to identify points in routes that are easy/hard or contain misleading information) we also implement what we previously termed the Perfect Memory model. In this variant, the rotated current views are compared directly in a pixel-wise manner to every training view in turn (which would not be plausible for an ant). In the following sections we first describe the perfect memory algorithm before describing the ANN and Infomax learning rule.

2.1 Route navigation with a perfect memory

In our 'perfect memory' version of the algorithm, each rotated view is sequentially compared to all of the training views. The best matching heading is then defined as the one with the lowest image difference, across all training views and rotations. Image difference can be calculated by various functions but here we use the average absolute difference between each of the image pixels to calculate the Image Difference Function (IDF):

$$IDF\Big(C(\vec{a},\theta), S(\vec{b},\phi)\Big) = \frac{1}{p \times q} \sum_{i=1}^{p} \sum_{j=1}^{q} |C_{i,j} - S_{i,j}| \tag{1}$$

where $C(\vec{a},\theta)$ is a $p \times q$ pixel view captured at location $\vec{a}$ with heading θ; $S(\vec{b},\phi)$ is a $p \times q$ pixel snapshot stored in memory; and $C_{i,j}$ and $S_{i,j}$ refers to the intensity of pixels in row i and column j of the captured view and stored snapshot respectively. A Rotational Image Difference Function (RIDF) is thus generated by varying theta through a range of angles and calculating the IDF at each rotation.

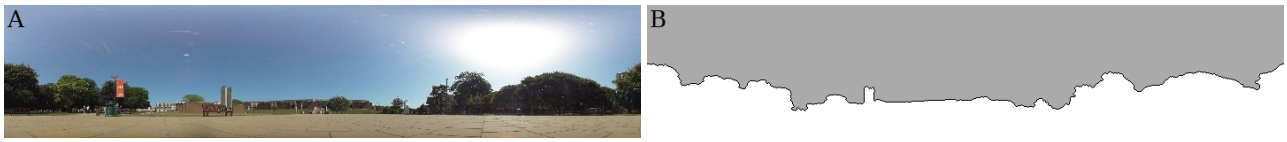

Figure 3: Example panoramic images from our database. **A** Raw unprocessed image. **B** Sky-segmented binary image.

Where there is a good match, there will be a minimum in the RIDF which defines the best matching direction. An example RIDF is shown in figure 2 which has a clear minimum at the best matching heading of around $60°$. As the RIDF will also have minima at other matching headings, we can use this to, for example, analyse locations where there has been a bad match to see whether it is the result of visual aliasing.

2.2 Familiarity-based route navigation

To perform familiarity discrimination we follow Baddeley et al. (2012) and use a neural network model that was specifically designed to perform this task (Lulham et al., 2011). The architecture consists of an input layer and a novelty layer with $tanh()$ activation functions. The number of input units is equal to the dimensionality of the input which in our case is $[120 \times 25] = 3000$, the number of pixels in a down-sampled view of the world. The number of novelty units is arbitrary and here we use the same number of novelty units as inputs although, using fewer novelty units has worked in simulation and will be tested in future work. The network is fully connected by feedforward connections w_{ij}. Weights are initialised randomly from a uniform distribution in the range $[-0.5, 0.5]$ and then normalised so that the mean of the weights feeding into each novelty unit is 0 and the standard deviation is 1. The network is then trained using the Infomax principle (Bell and Sejnowski, 1995), adjusting the weights so as to maximise the information that the novelty units provide about the input, by following the gradient of the mutual information using equation 4 which performs gradient ascent using the natural gradient (Amari, 1998) of the mutual information over the weights (Lee and Sejnowski, 1997) During learning the activation of each of the M novelty units h_i is computed as:

$$h_i = \sum_{j=1}^{N} w_{ij} x_j \tag{2}$$

where x_j is a row vector assembled by concatenating the rows of $C(\vec{a}, \theta)$ and $N = p \times q$ (the number of input units). The output y_i of the novelty units is then given by:

$$y_i = tanh(h_i) \tag{3}$$

The weights are then adjusted using the following learning rule:

$$\Delta w_{ij} = \frac{\eta}{N}\Big(w_{ij} - (y_i + h_i)\sum_{k=1}^{N} h_k w_{kj}\Big) \tag{4}$$

where η is the learning rate and is set as 0.01 for this paper. Finally, the response of the network to the presentation of an unseen N-dimensional input $\vec{x}$ is computed as

$$\text{fam}(\vec{x}) = \sum_{i=1}^{N} |h_i| \tag{5}$$

where $||$ denotes the absolute value. By applying $C(\vec{a}, \theta)$ to the ANN for a range of θ, an RFF can be calculated from $\text{fam}(\vec{x})$.

2.3 Robot platform

In this work we use the robot platform developed by Domcsek et al. (2018) shown in figure 1. This robot is based on a Parallax 'Shield-Bot' chassis (Parallax Inc., 2012), with a Jetson TX1 embedded computer (NVIDIA Corporation, 2016) mounted on top for additional onboard computation and additional batteries mounted underneath. The Jetson TX1 is connected via USB to a Kodak PixPro SP360 4K camera (JK Imaging Ltd., 2016), mounted on top of the robot connected which provides panoramic visual input.

2.4 Image processing and data collection

Using a Kodak PixPro SP360 4K panoramic camera (JK Imaging Ltd., 2016), we recorded 195 images of the Library Square at the University of Sussex. These were taken on a $1.2\,\text{m}$ grid, aligned with the slabs the square is paved with. As well as this reference grid of images, we also recorded videos from the same camera mounted on the mobile robot described in the previous section, as we manually drove it along six routes of varying lengths and tortuosity across the square. We tracked the robot by using the Discriminative Correlation Filter Tracker with Channel and Spatial Reliability (Lukežič et al., 2018) implementation provided by OpenCV (Bradski, 2000) to extract the position of the robot over time from video captured by a tripod-mounted camera. Finally, we used OpenCV to apply a perspective transform to the positions extracted by the tracker and married these final positions with the video frames captured by the robot.

As it has previously been shown that the sky can give erroneous information for visual homing and that visual homing can in fact be achieved using a binary image consisting of sky/not-sky (Philippides et al., 2011; Stone et al., 2014) – we wanted to compare the use of raw and binary images. To achieve this in real-time on the robot, we used the watershed segmentation algorithm (Beucher, 1979) with markers

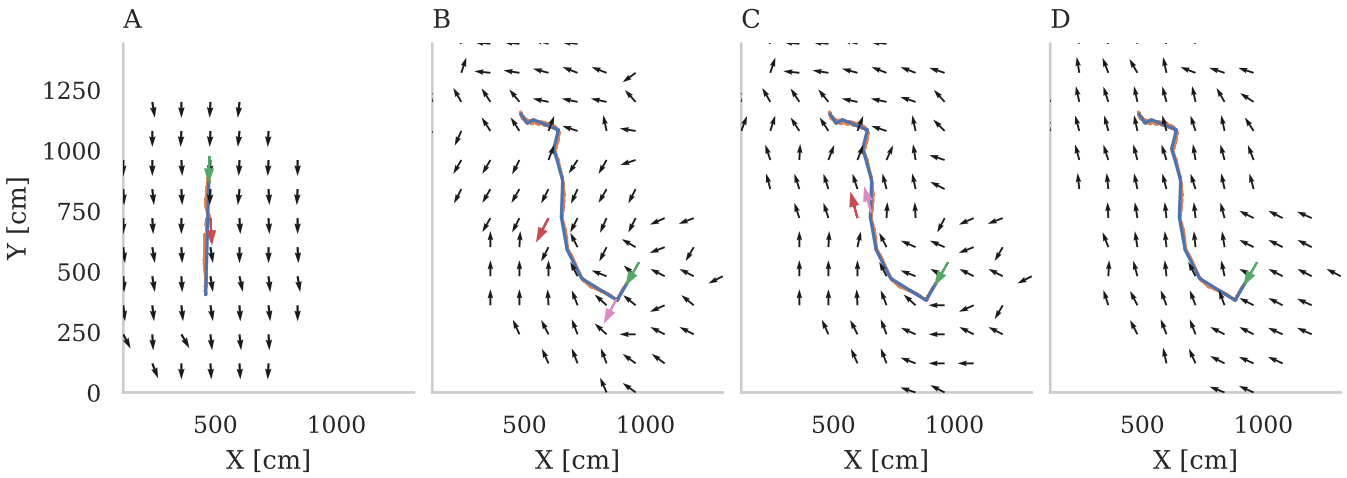

Figure 4: Vector fields showing the directions an agent, trained on a route, would move at each point within 4 m of the route using a Perfect Memory algorithm with the skyline extracted using the watershed algorithm. Orange lines shows data from our camera-based tracking of the robot and blue lines show the version simplified using the Ramer-Douglas-Peucker algorithm (Ramer, 1972). Green arrows indicate the position and direction the robot starts at. Red and pink arrows indicate positions in the vector field referred to in later analysis. **A** Simple route (route 3). **B** Longer route (route 5) where visual aliasing occurs in the middle section. **C** By only considering rotated views within 90° of the route, visual aliasing problems can be avoided in route 5. **D** Using InfoMax, tortuous elements of route 5 are smoothed out in the vector field.

placed at the top and bottom of each image. Figure 3 shows some example images from this database.

3 Results

3.1 Offline simulations

We trained both the InfoMax and Perfect Memory algorithms on the images taken along the six routes in our dataset and then used them to find the direction a robot would move when placed at each grid point within 4 m of the trained route. Figure 4 shows some example vector fields obtained by plotting this direction at each of the chosen grid locations when using the Perfect Memory algorithm described in section 2.1 with the watershed segmentation-based pre-processing discussed in section 2.4. While the vector field suggests that the route shown in Figure 4a would be recapitulated successfully, in the middle section of the route shown in Figure 4b, errors occur. Figure 5a shows a RIDF taken at one of the problematic locations on this route (600 m, 720 m) and it is clear that, as well as the local minima representing the correct heading at 15°, there is an additional, slightly lower global minimum at 153° which is overriding the correct choice. Figure 5 show the per-pixel differences between the best-matching and the closest images (in terms of distance to the current point) taken from training route; and the rotated versions of current image which best match these training images. Although the *shape* of the skyline in the closest image (figure 5c) is clearly better matched than it is with the best-matching image (figure 5b), there is a vertical offset – probably caused by camera shake – which introduces a large difference between the two images. Therefore the perfect memory model matches the more distant image better and thus the direction taken is similar to the direction taken at this point in the route which results in an error at this location (indicated with a pink arrow in figure 4b).

However, if we were recapitulating the route using a real robot, these false-positive matches could be eliminated and computation could be saved by simply *not* scanning the full ±180° but instead only scanning ±90° around the robot's current heading. However, unlike in the live robot tests, in the database there is no current heading. We therefore simulated the effect of this modified algorithm using our database of images by simplifying each route using the Ramer-Douglas-Peucker algorithm (Ramer, 1972) and calculating the direction of each of the resultant segments. Each point in the database is then assigned a 'current' heading which is equal to the direction of the nearest segment. Using these headings, we can then ignore matches that would involve heading more than ±90° away from the direction of the nearest section of the route and figure 4c shows that this step solves the aliasing problems in this particular case. In order to quantify the performance of our algorithms, we used the absolute difference between the most familiar heading angle obtained at each location on the grid and the direction of the nearest route segment as an error measure. The distributions of these errors across all of the routes, algorithms and image pre-processesing steps is plotted in Figure 7 and shows that, in fact, only scanning ±90° improves performance in all cases. Furthermore, figure 7 also

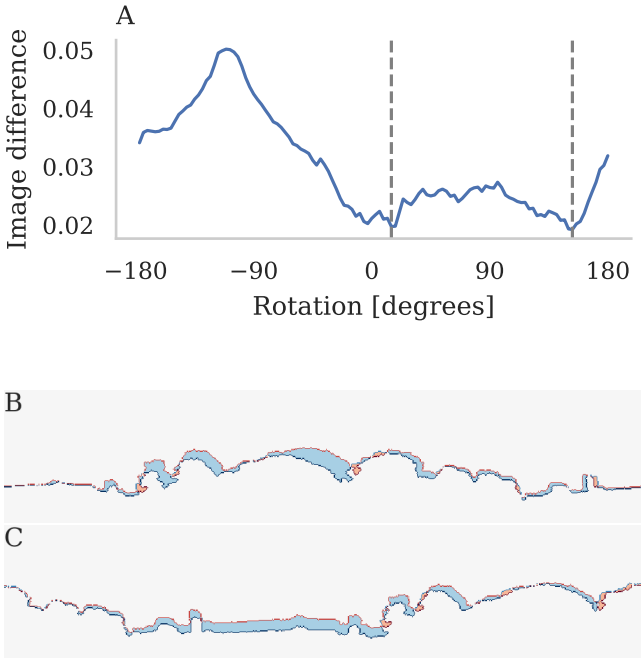

Figure 5: Analysis of aliasing shown in figure 4b. Using grid image taken at location marked with red arrow in figures 4b and 4c (600 m, 720 m). All images were preprocessed using watershed segmentation algorithm. **A** Rotational Image Difference function. **B** Difference image with visual aliased route image found at location marked with pink arrow in figure 4b. **C** Difference image with correct route image found at location marked with pink arrow in figure 4c

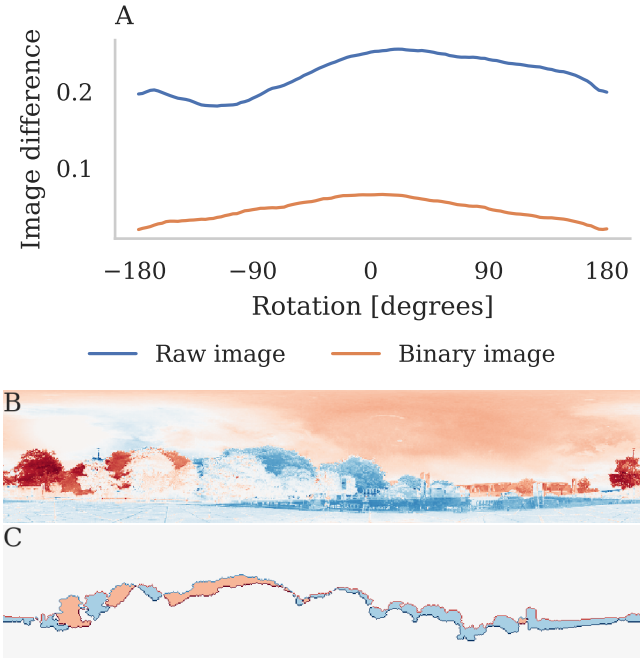

Figure 6: Analysis of poor performance on route 3 when using raw input images. Using grid image taken at location marked with red arrow in figure 4a (480 m, 720 m). **A** Rotational Image Difference function. **B** Difference image based on raw input images. **C** Difference image based on sky-segmented binary input images.

shows that, when using the Perfect Memory algorithm, sky-segmentation improves performance for almost all routes. This is particularly noticeable on route 3 (a straight line) where, when using sky-segmented input images, all of the algorithms reconstruct the direction of the route with a very small median error of around $4°$ but, when using unprocessed input images, this increases to around $40°$. Figure 6a shows that, unlike the situation explored earlier in this section, this is not an aliasing problem as there is only a single minima present in each RIDF. However, the magnitude of the average image differences between the raw images is much larger than that between the sky-segmented images and the minima is located at the incorrect location. The difference images shown in figures 6b and 6c suggest that the incorrect position of the minima when comparing the raw images is likely to be due to the large differences in the sky portion of the image due to clouds and the images having been recorded at a different time of day.

After applying sky-segmentation to the input images and only scanning $\pm 90°$ away from the direction of the nearest section of the route, the InfoMax ANN results in a lower median error than the Perfect memory control algorithm in 3 of the 6 routes although this difference is not statistically significant. Furthermore, if we compare figures 4c and 4d, the vector field plotted using the InfoMax ANN appears much smoother, suggesting that rather than losing useful information, the InfoMax encoding may actually be beneficial for smoothing out overly tortuous training routes.

3.2 Computational cost of algorithm

In the previous section, we showed that the algorithms described in sections 2.1 and 2.2 are capable of robustly extracting heading direction in outdoor scenes. However, in order to deploy these algorithms on a real robot with constrained on board processing, their performance is important. In order to measure performance in a controlled scenario, we wrote a benchmarking application which runs on the Jetson TX1 and trains either the Perfect memory or the InfoMax algorithm with varying numbers of images and then times how long it takes to extract a heading from a testing image (averaged over 100 iterations) using `std::chrono::high_resolution_clock`. Figure 8 shows the performance of our implementation of the InfoMax algorithm compared to the Perfect memory control using 120×25 pixel input images. Clearly, for small numbers of stored images, extracting heading information using the Perfect Memory is faster but, assuming that training images

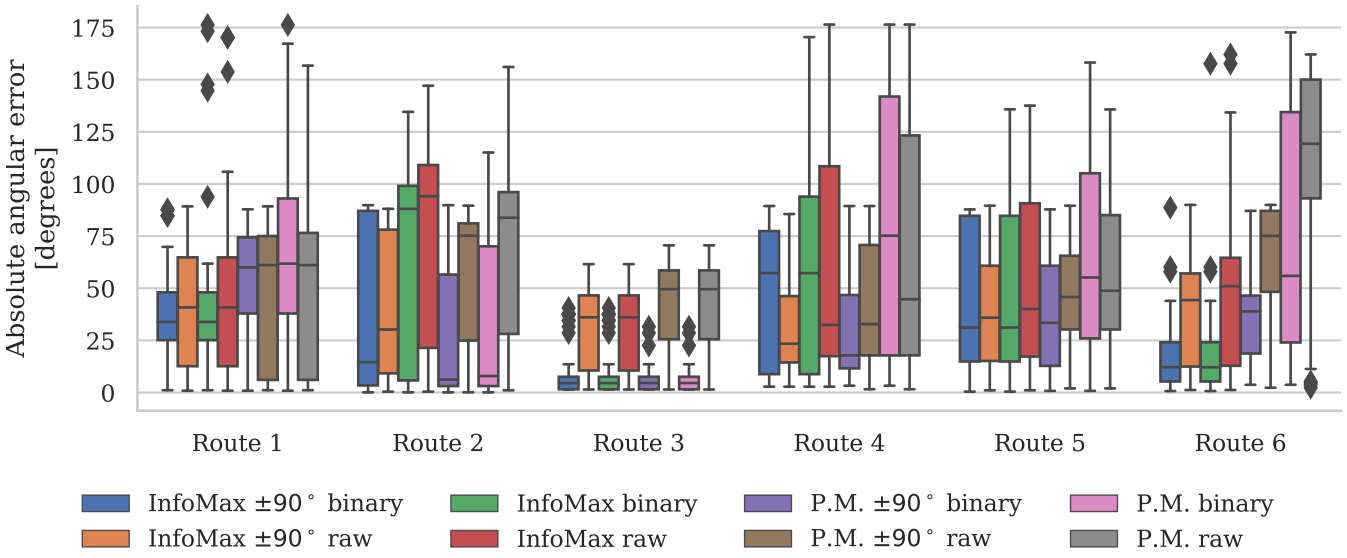

Figure 7: Performance of different algorithms on each of the 6 routes.

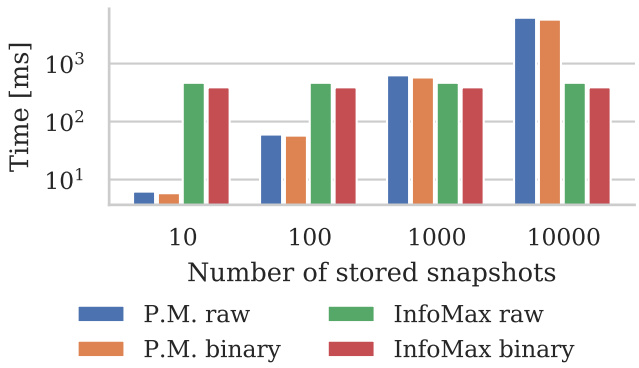

Figure 8: Performance of visual navigation algorithms running on Jetson TX1. Reported times are measured using `std::chrono::high_resolution_clock` and average is taken over 100 images.

Figure 9: Environment used for robot testing.

are sampled every 100 ms, InfoMax begins to more efficient for routes with only a little over 1 min of training data making it much more feasible approach for long-range visual homing.

3.3 Autonomous robot

Using the implementations of InfoMax and the Perfect Memory algorithms from our Brains-on-Board Robotics library (Dewar et al., 2017), we built a simple application which can recapitulate learned routes on the robot described in section 2.3 by running the following simple algorithm every 500 ms (based on the performance for 1000 images established in the previous section):

1. Capture and unwrap a panoramic image.
2. Perform one of the image processing steps described in section 2.4.
3. Using either the InfoMax or Perfect Memory algorithm, calculate the familiarity with the processed image in-silico when rotated through ±90°.
4. Find the orientation with the highest familiarity and, if it is within 4°, start driving forwards. Otherwise, start turning in the correct direction to align with the image.

In order to compare the performance of the navigation algorithms and image processing steps described in sections 2.1, 2.2 and 2.4 running on the robot, we first manually drove the robot along a sinuous route through the wooded area shown in figure 9, recording training images every 100 ms. We then trained each of the navigation algorithms on the resultant dataset of 455 images and allowed the

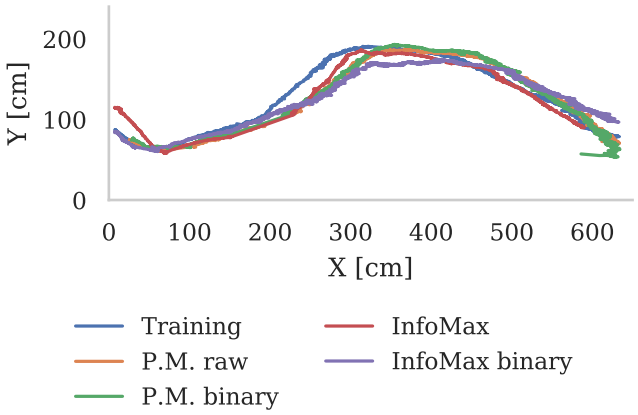

Figure 10: Reconstructed paths of autonomous robot during training and testing using each algorithm.

robot to recapitulate the path using the procedure described above. Throughout this training and testing process, we used the method described in section 2.4 to track the robot, resulting in the data shown in figure 10. The robot was able to successfully recapitulate the training path using each of the navigation and image processing algorithms with little difference in performance immediately apparent in figure 10. We confirmed this by calculating the shortest distance to the training path at each location along each recapitulated path and found that, in this environment, there was no significant difference between the performance of the different algorithms and overall the mean of the distance between the training and recapitulated paths was $9\,\mathrm{cm}$, with a standard deviation of $8\,\mathrm{cm}$.

4 Summary and Discussion

In this paper we present the first familiarity-based navigation algorithm that can function effectively in the real world on board a small mobile robot using an ANN-based route encoding. Indeed, we show that in some cases, using an ANN can improve performance as it imposes a level of generalisation meaning that particularly tortuous elements of the training route (which we presume to be from a noisy training process and not something that needs to be recapitulated exactly to enable homing) are smoothed out in the recapitulated trajectories.

In addition, we reinforce results showing that removing the sky and using only a binary image make these algorithms more robust to different lighting and weather conditions (Philippides et al., 2011; Stone et al., 2014). Furthermore, these improvements are seen even if the segmentation is performed automatically on board the robot using an RGB image, processed using a simple watershed algorithm rather than requiring specialist sensors such as a UV camera (although such a specialist sensor would undoubtedly improve performance and reduce computation). Interestingly, these improvements are not noticeable in the results collected using the autonomous robot. We believe that this is due to the fact that we performed these experiments in a more enclosed environment, where the sky covered a smaller portion of each image and because the testing and training both occurred at the same time of day. These two factors both act to reduce the differences between the raw images meaning that raw image comparisons are successful. Additionally, we show that the computation and storage needed to recapitulate a route using the Infomax ANN does not scale with the number of images used in training. To put this into context of robotic control algorithms, this is not typically a property of SLAM based localization systems where more keyframes are accumulated over time although there has been recent work to cap (Maddern et al., 2012) or at least cull (Mur-Artal et al., 2015) the number of keyframes accumulated.

Moreover, we note that this is not the only relevant computational advantage of our algorithm. Firstly, our familiarity based algorithms work well for low-resolution visual sensors and, as with the low-res eyes of ants **(TODO: REF)**, it is indeed better to not us high-resolution images (Wystrach et al., 2016). In contrast, visual SLAM implementations extract features such as SURF or SIFT which require several hundreds of milliseconds to extract per frame (Bay et al., 2006). Recent SLAM implementations such as FLaME (Greene and Roy, 2017) have been demonstrated running on autonomous quadrotors at much higher framerates than our current InfoMax implementation, running on a Jetson TX1, can achieve. However, not only was FLaME implemented on an Intel CPU which Biddulph et al. (2018) measured as being $5\times$ faster than a Jetson TX1, but the performance of our InfoMax algorithm could be significantly improved. Firstly, 120×25 pixel images were used throughout the work presented in this paper but, when Baddeley et al. (2012) first demonstrated InfoMax for visual navigation, they used input images with around half this number of pixels (90×17). Equation 5 can be implemented as a matrix-vector product – the computational complexity of which scales quadratically with N. Therefore, using input images with half the number of pixels would quarter the time taken to evaluate equation 5. Furthermore, while our InfoMax implementation uses OpenMP **(TODO: REF)** to take advantage of the the Jetson TX1's four CPU cores, it does not utilise the 256 core GPU present on the Jetson TX1. Initial experiments using the cuBLAS (NVIDIA Corporation, 2007) GPU-accelerated linear algebra library suggest that equation 5 could be evaluated in around $100\,\mathrm{ms}$ for 120×25 pixel images – a $5\times$ speedup over our current implementation. Thus, as with SLAM which has seen vast increases in performance efficiency, further work on the ANN encoding, visual processing and behavioural strategy, will extend the range of homing and reduce computation and training times still further.

5 Acknowledgements

This work was funded by the EPSRC (Brains on Board project, grant number EP/P006094/1).

References

Amari, S. (1998). Natural gradient works efficently in learning. *Neural Computation. 02/15/98*, 10:251–276.

Ardin, P., Peng, F., Mangan, M., Lagogiannis, K., and Webb, B. (2016). Using an Insect Mushroom Body Circuit to Encode Route Memory in Complex Natural Environments. *PLoS Computational Biology*, 12(2):1–22.

Baddeley, B., Graham, P., Husbands, P., and Philippides, A. (2012). A model of ant route navigation driven by scene familiarity. *PLoS Computational Biology*, 8(1).

Baddeley, B., Graham, P., Philippides, A., and Husbands, P. (2011a). Holistic visual encoding of ant-like routes: Navigation without waypoints. *Adaptive Behavior*, 19(1):3–15.

Baddeley, B., Graham, P., Philippides, A., and Husbands, P. (2011b). Models of visually guided routes in ants: Embodiment simplifies route acquisition. In *International Conference on Intelligent Robotics and Applications*, pages 75–84. Springer.

Bay, H., Tuytelaars, T., and Gool, L. V. (2006). SURF: Speeded Up Robust Features - Demonstration. *Computer Vision ECCV*, pages 404–417.

Bell, A. J. and Sejnowski, T. J. (1995). An Information-Maximization Approach to Blind Separation and Blind Deconvolution. *Neural Computation*, 7(6):1129–1159.

Beucher, S. (1979). Use of watersheds in contour detection. In *Proceedings of the International Workshop on Image Processing*.

Biddulph, A., Houliston, T., Mendes, A., and Chalup, S. K. (2018). Comparing Computing Platforms for Deep Learning on a Humanoid Robot. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11307 LNCS:120–131.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Dewar, A., Knight, J. C., Domcsek, N., and Cope, A. J. (2017). BoB robotics. `https://github.com/BrainsOnBoard/bob_robotics`. Last accessed on 07/03/2019.

Domcsek, N., Knight, J., and Nowotny, T. (2018). Autonomous robot navigation using GPU enhanced neural networks. In *Journal of Robotics and Autonomous Systems*, volume 1, pages 77–79, Bristol.

Greene, W. N. and Roy, N. (2017). FLaME: Fast Lightweight Mesh Estimation Using Variational Smoothing on Delaunay Graphs. *Proceedings of the IEEE International Conference on Computer Vision*, 2017-Octob:4696–4704.

JK Imaging Ltd. (2016). SP360 4K 360 Degree VR Camera. `https://kodakpixpro.com/cameras/360-vr/sp360-4k`. Last accessed on 05/03/2019.

Knaden, M. and Graham, P. (2016). The sensory ecology of ant navigation: from natural environments to neural mechanisms. *Annual review of entomology*, 61:63–76.

Lee, T.-w. and Sejnowski, T. J. (1997). Independent Component Analysis for Mixed Sub-Gaussian and Super-Gaussian Sources. *Joint Symposium on Neural Computation*, 441:6–13.

Lukežič, A., Vojí, T., ČehovinZajc, L., Matas, J., and Kristan, M. (2018). Discriminative Correlation Filter Tracker with Channel and Spatial Reliability. *International Journal of Computer Vision*, 126(7):671–688.

Lulham, A., Bogacz, R., Vogt, S., and Brown, M. W. (2011). An infomax algorithm can perform both familiarity discrimination and feature extraction in a single network. *Neural Computation*, 23(4):909–926.

Maddern, W., Milford, M., and Wyeth, G. (2012). Capping computation time and storage requirements for appearance-based localization with CAT-SLAM. In *2012 IEEE International Conference on Robotics and Automation*, pages 822–827. IEEE.

Mur-Artal, R., Montiel, J. M., and Tardos, J. D. (2015). ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, 31(5):1147–1163.

NVIDIA Corporation (2007). cuBLAS. `https://developer.nvidia.com/cublas`. Last accessed on 05/03/2019.

NVIDIA Corporation (2016). Jetson TX1. `https://developer.nvidia.com/embedded/buy/jetson-tx1`. Last accessed on 05/03/2019.

Parallax Inc. (2012). Robot Shield with Arduino. `https://www.parallax.com/product/32335`. Last accessed on 05/03/2919.

Philippides, A., Baddeley, B., Cheng, K., and Graham, P. (2011). How might ants use panoramic views for route navigation? *Journal of Experimental Biology*, 214(3):445–451.

Philippides, A., Graham, P., Baddeley, B., and Husbands, P. (2015). Using Neural Networks to Understand the Information That Guides Behavior: A Case Study in Visual Navigation. In *Artificial Neural Networks*, pages 227–244.

Ramer, U. (1972). An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1(3):244–256.

Stone, T., Mangan, M., Ardin, P., and Webb, B. (2014). Sky segmentation with ultraviolet images can be used for navigation. *Robotics: Science and Systems X*.

Wehner, R. (2009). The architecture of the desert ants navigational toolkit (hymenoptera: Formicidae). *Myrmecol News*, 12(September):85–96.

Wehner, R., Boyer, M., Loertscher, F., Sommer, S., and Menzi, U. (2006). Ant navigation: one-way routes rather than maps. *Current Biology*, 16(1):75–79.

Wystrach, A., Dewar, A., Philippides, A., and Graham, P. (2016). How do field of view and resolution affect the information content of panoramic scenes for visual navigation? A computational investigation. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, 202(2):87–95.

Wystrach, A., Mangan, M., Philippides, A., and Graham, P. (2013). Snapshots in ants? new interpretations of paradigmatic experiments. *Journal of Experimental Biology*, 216(10):1766–1770.

Wystrach, A., Philippides, A., Aurejac, A., Cheng, K., and Graham, P. (2014). Visual scanning behaviours and their role in the navigation of the australian desert ant melophorus bagoti. *Journal of Comparative Physiology A*, 200(7):615–626.

Zeil, J., Hofmann, M. I., and Chahl, J. S. (2003). Catchment areas of panoramic snapshots in outdoor scenes. *JOSA A*, 20(3):450–469.