Critical Exponent of Species-Size Distribution in Evolution

James C. Knight[1] and Andrew Philippides[1]

[1]Centre for Computational Neuroscience and Robotics, University of Sussex, Brighton, UK
J.C.Knight@sussex.ac.uk

Abstract

Inspired by the visual scanning behaviour seen in ants, models of visually-guided navigation that operate by searching for previously experienced views of the world have been shown to be capable of facilitating robust navigation. However, these algorithms have only been previously tested using simulated agents in virtual environments, not on natural scenes or with physical robotic agents.**(TODO: ANDY IS THIS TRUE? LIVING MACHINES PAPER DOES LOOK AT NATURAL SCENES BUT NOT REALLY NAVIGATION.)** In this work we explore the performance of these algorithms both offline, using a dataset consisting of images and video routes captured in an outdoor environment, and running on-board an autonomous robot. The results from our experiments show that these insect-inspired algorithms are capable of extracting reliable direction information even when scenes contain few local landmarks and high-levels of noise. Furthermore, when running this algorithm using an ANN memory on our autonomous robot, we demonstrate that routes can be precisely recapitulated with our robot only straying an average of 10 cm away from the trained path. Additionally, the computation required for this ANN memory does not scale with the number of images used in the training route and thus the ANN provides a compact representation of the knowledge needed to traverse a route. This suggests the feasibility of our approach for long-range 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. In the animal kingdom, desert ants are one of the champion visual navigators. Despite having small brains and low resolution vision, these ants habitually traverse long routes (**(TODO: X)** m) through complex terrain. However, in direct contrast with most modern robotic methods, to navigate between two locations, insects use route knowledge not mental maps **(TODO: REFERENCE)**. 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 without specifying when or what to learn **(TODO: REFERENCES)**.

Our algorithm starts 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. 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 **(TODO: REF)**. 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 **(TODO: REF)**.

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 by physically rotating – a behaviour seen in ants **(TODO: REFERENCE)** – 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 complex environments as well as learning multiple paths to a single goal in a single trial with performance robust to sensor and motor noise (Baddeley et al.,

Figure 1: Robot platform with onboard computation.

2012). However the algorithm equipped has not previously been tested in an outdoor context. Here, we show that a robot equipped with a Jetson TX1 (NVIDIA Corporation, 2016) can navigate fully autonomously outdoors using a single layer neural network.

2 Methods

As described above, at the heart of our algorithm is the use of a remembered view as a 'visual compass' which can be used to recall the heading at which that view was stored. For an ant or wheeled robot that is constrained to move in the direction that is facing, the visual compass thus recalls the direction of movement when the view was stored. This process is implemented by rotating the current view and finding the most familiar direction either by inputting to the ANN encoding or, as a control, comparing iteratively to the set of training views. In the following sections we first describe the iterative algorithm followed by the Infomax ANN architecture.

2.1 RIDF and Perfect memory

As a comparison to the Infomax ANN we describe in the next section, we also use a 'perfect memory' version of the algorithm. In this variant, each rotated view is simply compared to all of the training views in turn. The best matching heading is then defined as the one with the lowest image difference, across all training views and rotations. The 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): **(TODO: FIX FORMULA)**

$$IDF(I(x,t), Ti(y,p)) = sqrtSumSum() \qquad (1)$$

(a) Unprocessed panoramic image

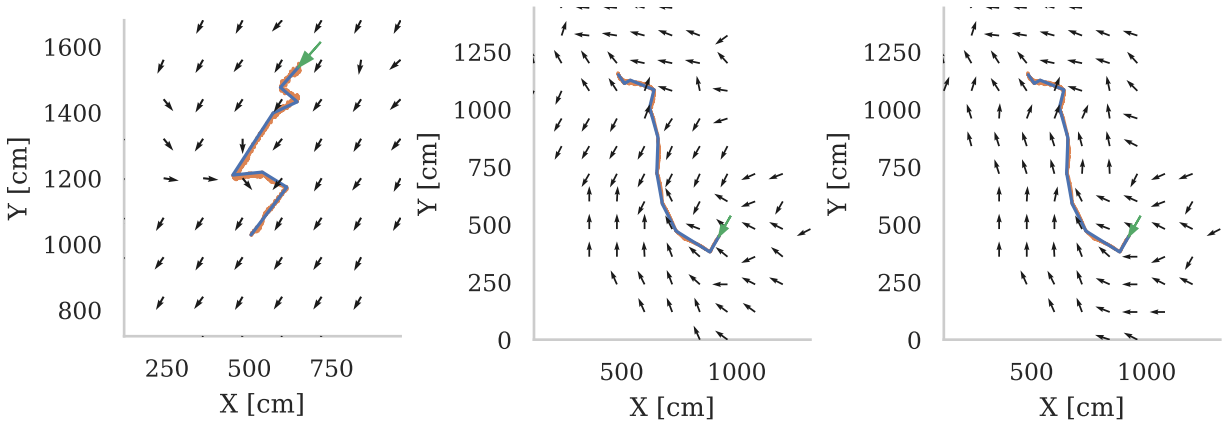
(b) Sky-segmented panoramic image

Figure 2: Example panoramic images from our database

where $I(x,t)$ and $Ti(y,p)$.**(TODO: FINISH)** An RIDF is thus generated by varying theta**(TODO: THERE IS NO THETA)** though a range of angles and calculating the IDF at each rotation. 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 **(TODO: DO WE INCLUDE ANOTHER EXAMPLE OR JUST POINT AT THE ONE LATER IN THE PAPER?)** which has a clear minimum at the best matching heading. As the RIDF will also have minima at other matching headings, we can use this to analyse points where there has been a good or bad match to see if there is any aliasing.

2.2 Familiarity and Infomax

In order to perform familiarity discrimination we chose to 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 follow Lulham et al. (2011) and use the same number of novelty units as inputs. We found that using as few as 200 novelty units can work well in many instances, but we do not explore this aspect of the problem as we were more interested in the behavioural consequences of a familiarity driven approach. 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. In our learning scheme, equation 4 is used to perform gradient ascent using the natural gradient (Amari, 1998) of the mutual information over the weights (Lee and



(a) Simple route

(b) Longer route where visual aliasing occurs in the middle section

(c) By only considering rotated views within 90 of the route, visual aliasing can be avoided

Figure 3: Vector fields showing the directions an agent, trained on a route, would move at each point within 4m 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 shvaryingows the version simplified using the Ramer-Douglas-Peucker algorithm (Ramer, 1972)

Sejnowski, 1997) (use of the natural gradient avoids the computationally expensive calculation of the inverse of the entire weight matrix). Since two novelty units that are correlated carry the same information, adjusting weights to maximise information will tend to de-correlate the activities of the novelty units and the algorithm can thus be used to extract independent components from the training data (Lee and Sejnowski, 1997). 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, with a limited amount of data, the algorithm is unlikely to converge to a particularly good set of independent components, it is enough that the components that are extracted provide a more suitable decomposition of the training data than of an arbitrary input. 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_i is the value of the ith input and N is 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

$$d(\vec{x}) = \sum_{i=1}^{M} |h_i| \tag{5}$$

where $||$ denotes the absolute value. The network response could be viewed as an output layer but, as it is a function of the activations of the novelty units, we follow Lulham et al. (2011) and do not represent it with another layer. As noted above, in this paper, we set $M = N$ and the network is trained with each training view presented just once to the network in the order in which it is experienced in training. Lulham et al. (2011) use $d(\vec{x})$ together with a threshold that must be determined empirically to determine whether the input is novel or familiar. For our purposes it is not necessary to determine a threshold as we only need to choose the most familiar input from a limited number of possibilities i.e. the views experienced during a single scan of the environment.

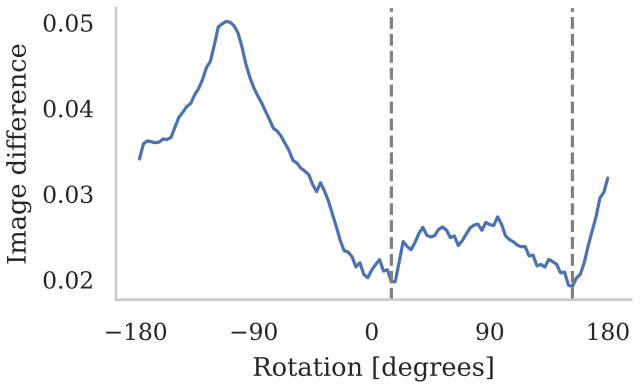
The difference between the way an image difference function and a neural network trained using an Infomax principle represent familiarity will be subtle. In essence, the difference is manifest in the way the information is stored. For image differences, each stored view defines a single point in an N-dimensional space, with N equal to the dimension of the images ($N = 120 \times 25 = 3000$) and the image difference function gives the squared Euclidean distance of an input image from one of these stored points. This requires that all of the views are stored, meaning that memory load increases proportional to the number of views experienced. The Infomax approach instead decomposes each view into a fixed number of components (determined by the number of hidden units in the network) which remains fixed, independent of the number of views experienced. Therefore, the Infomax measure is more abstract and reflects whether a test input is well described in terms of the learned components that the hidden units represent. However, by decomposing the input in this way, it is possible to compress redundant data resulting in more efficient memory storage.

2.3 Robot platform

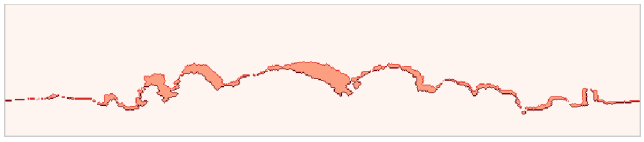
In this work we use the robot platform developed by Domcsek et al. (2018) shown in figure 1. This robot is based on a Parallax 'Shield-Bot' chassis (Parallax Inc., 2012), with a Jetson TX1 embedded computer (NVIDIA Corporation, 2016) mounted on top for additional onboard computation and additional batteries 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 database

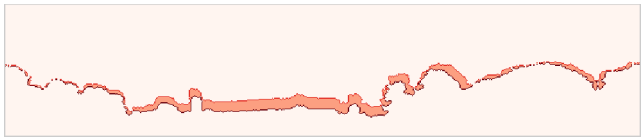
Using a Kodak PixPro SP360 4K panoramic camera (JK Imaging Ltd., 2016), we recorded 195 images of the Library Square at the University of Sussex. These were taken on a 1.2 m grid, aligned with the slabs the square is paved with. As well as this reference grid of images, we also recorded videos from the 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. Stone et al. (2014) showed that sky-segmented, binary images can be used for robust visual navigation and, while our robot does not have a suitable UV camera, by using the watershed segmentation algorithm (Beucher, 1979) with markers placed at the top and bottom of each image, we obtained automatically sky-segmented versions of all of the images in our dataset. Figure 2 shows some example images from this database.



(a) Rotational Image Difference function



(b) Difference image with visual aliased route image



(c) Difference image with correct route image

Figure 4: Analysis of aliasing shown in figure 3b. Using grid image taken at (600 m, 720 m). All images were preprocessed using watershed segmentation algorithm.

3 Results

3.1 Image database

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 3 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 preprocessing discussed in section 2.4. While the vector field suggests that the route shown in Figure 3a would be recapitulated successfully, in the middle section of the route shown in Figure 3b, errors occur. Figure 4a shows a RIDF taken at one of the problematic locations on this route **(TODO: MARK GRID AND ADD ARROWS SHOWING ALIASES)** (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 local minima at 153° which is overriding the correct choice. Figures 4b and 4c show the per-pixel dif-

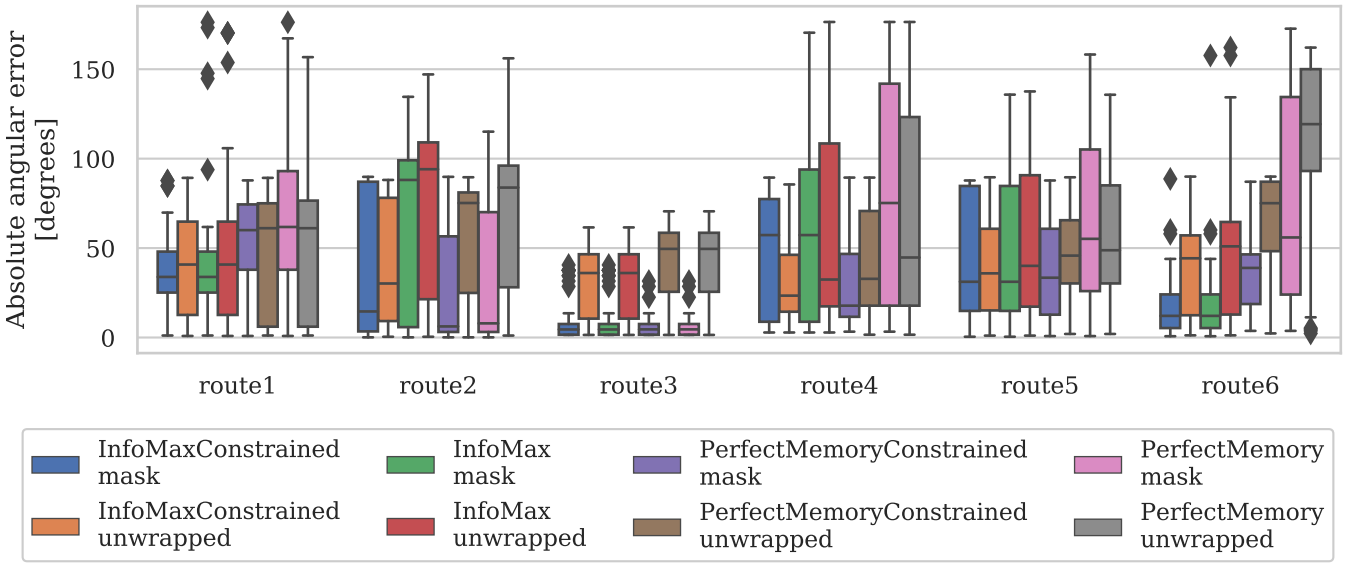


Figure 5: Performance of different algorithms on each of the 6 routes. **(TODO: DECIDE ON CONSISTENT NAMES FOR ROUTES)**

ferences between the two route images and the rotated versions of the grid image which correspond with these minima and this makes the cause of the problem become clear. Although the *shapes* of the skylines in figure 4c are clearly more similar than those in figure 4b, there is a vertical offset – probably caused by camera shake – which introduces a large difference between the two images.

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 $\pm 180°$. We 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. Using these headings, we can then ignore matches that would involve heading more than $\pm 90°$ away from the direction of the nearest section of the route. Figure 3c shows that this step solves the aliasing problems in this particular case and figure 5 confirms that, infact, this step improves performance across all algorithms and routes. Furthermore, figure 5 also shows that, when using the Perfect Memory algorithm, sky-segmentation improves performance for almost all routes. This improvement is less significant when using the InfoMax algorithm but, **(TODO: EXPLORE CASE WHERE INFOMAX SUCCEEDS WHEN PM DOESN'T)**

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.

3.3 Autonomous robot

Using the implementations of InfoMax and the Perfect Memory algorithms from our Brains-on-Board Robotics **(TODO: REF)** library, 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 $500\,\mathrm{ms}$ (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 $\pm 90°$.
4. Find the orientation with the highest familiarity and, if it is within $4°$ **(TODO: CHECK)**, 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 7, recording training images every $100\,\mathrm{ms}$**(TODO: CHECK)**. We then trained each of

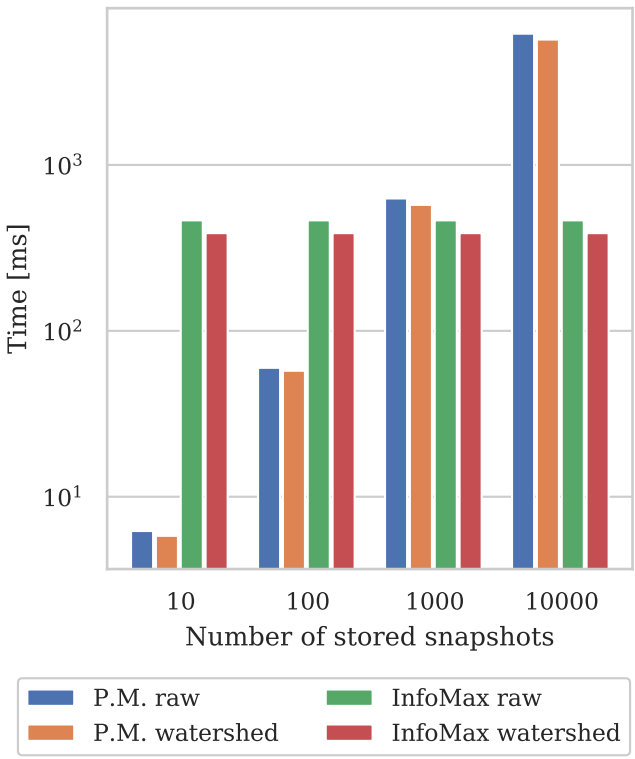


Figure 6: Performance of visual navigation algorithms running on Jetson TX1. Reported times are measured using `std::chrono::high_resolution_clock` and average is taken over 100 images.

the navigation algorithms on the resultant dataset of 455 images and allowed the 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 8. 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 8. 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 cm, with a standard deviation of 8 cm.

4 Discussion

- how/why SLAM is the obvious alternative
- General limitations of monocular SLAM

Typically, visual SLAM implementations extract features such as SURF or SIFT which require several hundreds of milliseconds to extract per frame (Bay et al., 2006). This

Figure 7: Environment used for robot testing.

makes such implementations impractical to use in real-time, especially on constrained mobile platforms. However, more recent SLAM implementations such as FLaME (Greene and Roy, 2017) have been shown to run on autonomous quadrotors in only around 10 ms per frame. While this is significantly faster than our current InfoMax implementation, Greene and Roy (2017) were using a much more powerful Intel CPU on their quadrotor which Biddulph et al. (2018) measured as being 5× faster than a Jetson TX1. Additionally, we used 120×25 pixel images for all of 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 cost 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: CITATION)** 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 ms for 120×25 pixel images – a 5× speedup over our current implementation.

5 Conclusions

ss

6 Acknowledgements

This work was funded by the EPSRC (Brains on Board project, grant number EP/P006094/1). We would also like to thank Daniil Sakhapov for his work collecting the 'Library Square' database of images, Alex Dewar for developing the BoB robotics framework and Norbert Domcsek for assisting with the data collection for this paper.

References

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

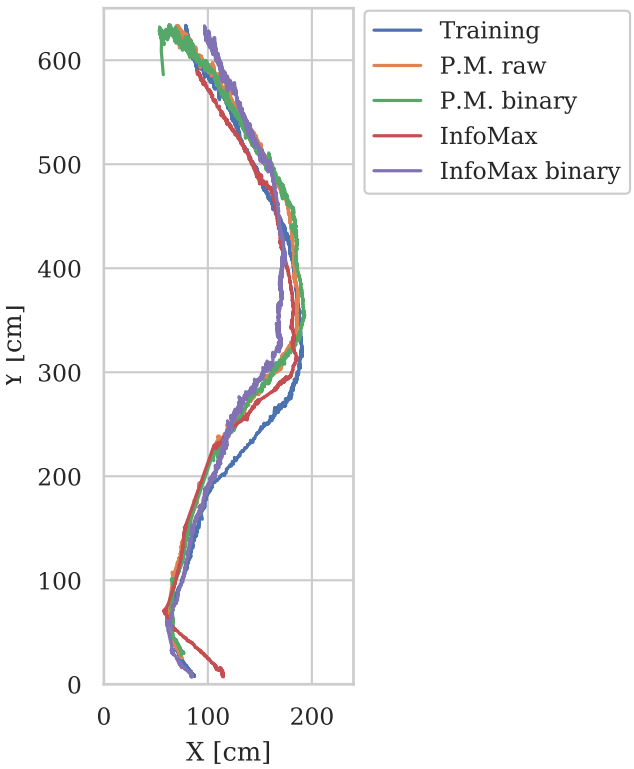


Figure 8: Reconstructed paths of autonomous robot during training and testing using each algorithm.

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).

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*.

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.

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.

NVIDIA Corporation (2007). cuBLAS.

NVIDIA Corporation (2016). Jetson TX1.

Parallax Inc. (2012). Robot Shield with Arduino.

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*.