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 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 onboard 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 these algorithms on our autonomous robot, we demonstrate that routes can be successfully recapitulated precisely – our robot remains an average of only $10\,\mathrm{cm}$ away from the trained path.

Figure 1: Robot platform with onboard computation.

1 Introduction

dd

2 Methods

2.1 Familiarity-driven navigation algorithms

- PerfectMemory
- InfoMax

2.2 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.

2.3 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 (figure?). These were taken on a $1.2\,\mathrm{m}$ grid, defined by squares paving slabs. Alongside this reference grid, we also recorded videos from the same camera, mounted on the mobile robot described in the previous section, while we manually drove it along six routes of varying tortuosity across the square. As well as these videos, taken from the robots point-of-view, we simultaneously recorded the position of the robot using a camera mounted on a tripod. After synchronising these videos, we extracted the position of the robot over time using the Discriminative Correlation Filter Tracker with Channel and Spatial Reliability (Lukežič et al., 2018) implementation provided by OpenCV (Bradski, 2000). Finally, we again used OpenCV to apply a perspective transform to the positions extracted by the tracker and synchronised these final positions with the video frames captured by the robot. Stone et al. (2014) showed that sky-segmented, binary images can

(a) Unprocessed panoramic image

(b) Sky-segmented panoramic image

Figure 2: Example panoramic images from our database

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.

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 $4\,\mathrm{m}$ of the trained route. Heading direction were calculated by rotating the grid image taken at a point through $\pm 180°$ and finding the angle at which the algorithm reported the highest familiarity. 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 with the watershed segmentation-based preprocessing discussed in section 2.3. 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 Rotational Image Difference Function (RIDF) taken at one of the problematic locations on this route **(TODO: MARK GRID AND ADD ARROWS SHOWING ALIASES)** ($600\,\mathrm{m}$, $720\,\mathrm{m}$) 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 differences 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 saved by not scanning the full $\pm 180°$. We can simulate this using our existing database of images by first making a simplified version of each route using the Ramer-Douglas-Peucker algorithm (Ramer, 1972) and determining 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 this is the case 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,

3.2 Performance of algorithm

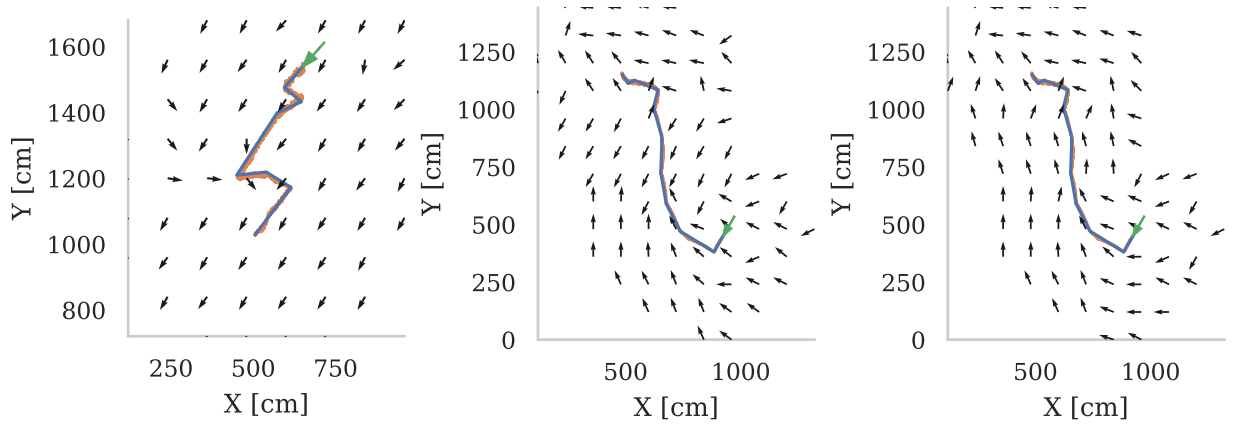
In the previous section, we showed that these simple insect-inspired algorithms are capable of robustly extracting heading direction in outdoor scenes. However, in order to deploy these algorithms on a real robot with constrained on board processing, their performance is important.

3.3 Autonomous robot

Using the implementations of InfoMax and the Perfect Memory algorithms from our Brains-on-Board Robotics **ref** library, we built a simple application which can recapitulate learned routes on the robot described in section 2.2 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 2.3.
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 and 2.3 when using this algorithm, 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)** (resulting in a dataset of 455 images). We then trained each of the navigation algorithms on this dataset and allowed the robot to recapitulate the path using the procedure described above. Throughout this training and testing process, we used the same method



(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 shows the version simplified using the Ramer-Douglas-Peucker algorithm (Ramer, 1972)

described in section 2.3 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 $9\,\mathrm{cm}$, with a standard deviation of $8\,\mathrm{cm}$.

4 Discussion

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

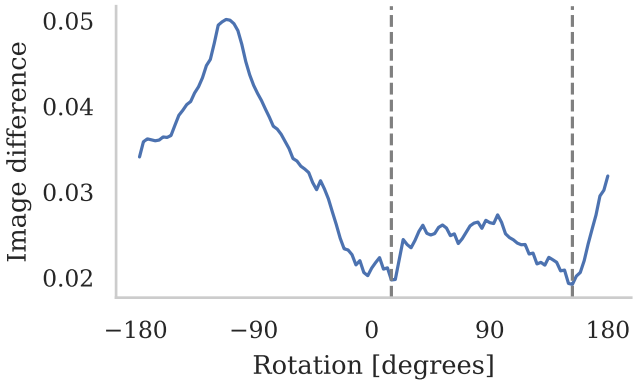
Typically, visual SLAM implementations extract features such as SURF or SIFT which require several hundreds of milliseconds to extract per frame (Bay et al., 2006). This makes such implementations impractical to use in real-time, especially on constrained mobile platforms. However, more recent SLAM implementations such as FLaME (Greene and Roy, 2017) have been shown to run on autonomous quadrotors in only around $10\,\mathrm{ms}$ 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 (Biddulph et al. (2018) measure these devices as being $5\times$ faster than a Jetson TX1). For consistency 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. Because the time taken to calculate the matrix-vector product required for equation **(TODO: REFERENCE)** scales quadratically, using input images with half the number of pixels would quarter the time taken to evaluate equation **(TODO: REFERENCE)**. 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 **(TODO: CITATION)** GPU-accelerated linear algebra library suggest that InfoMax inference could be performed in around $100\,\mathrm{ms}$ for 120×25 pixel images.

5 Conclusions

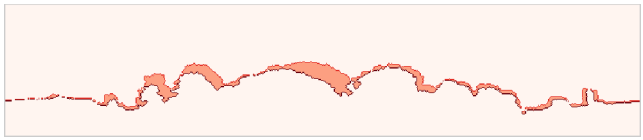
ss

6 Acknowledgements

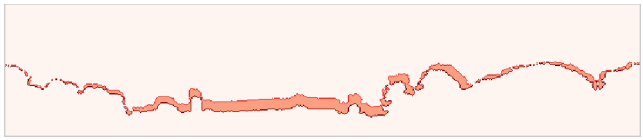
This work was funded by the EPSRC (Brains on Board project, grant number EP/P006094/1). We would also like



(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 pre-processed using watershed segmentation algorithm.

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

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.

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.

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.

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*.

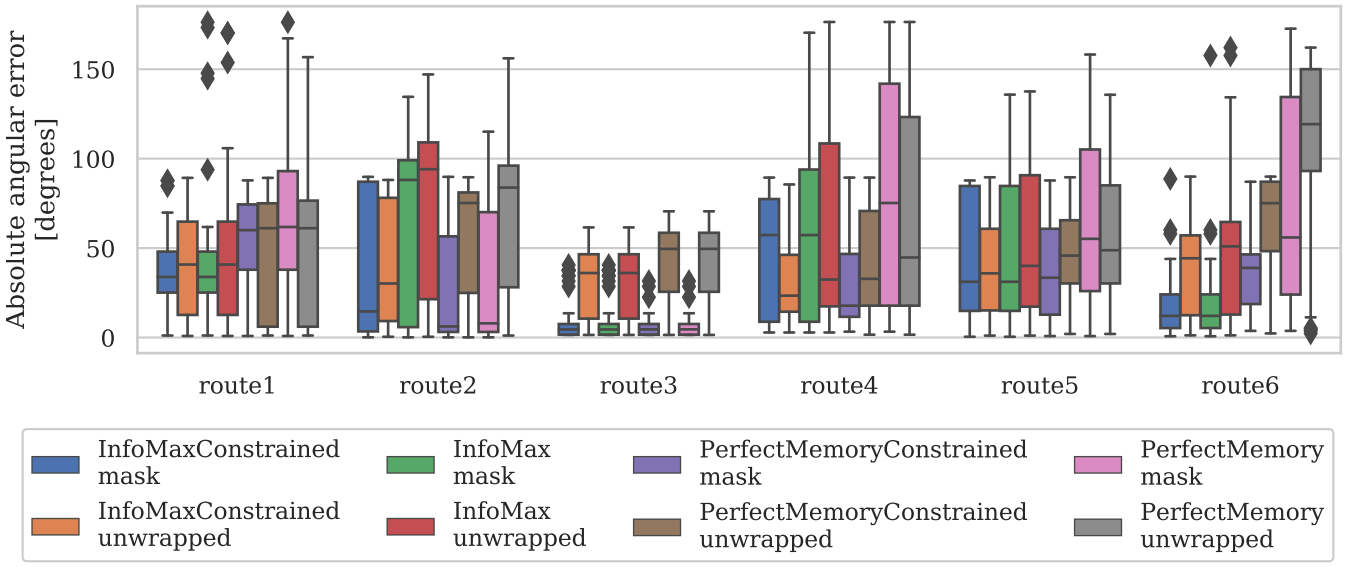


Figure 5: Performance of different algorithms on each of the 6 routes. **(TODO: DECIDE ON CONSISTENT NAMES FOR ROUTES)**

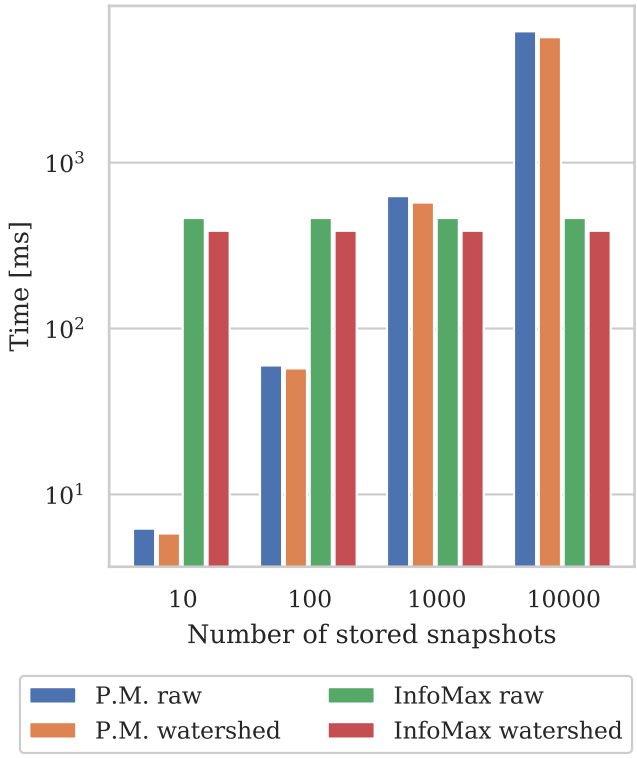


Figure 6: Performance of visual navigation algorithms running on Jetson TX1. Reported times are measured using `std::chrono::high_resolution_clock` and average is taken over 100 images.

Figure 7: Environment used for robot testing.

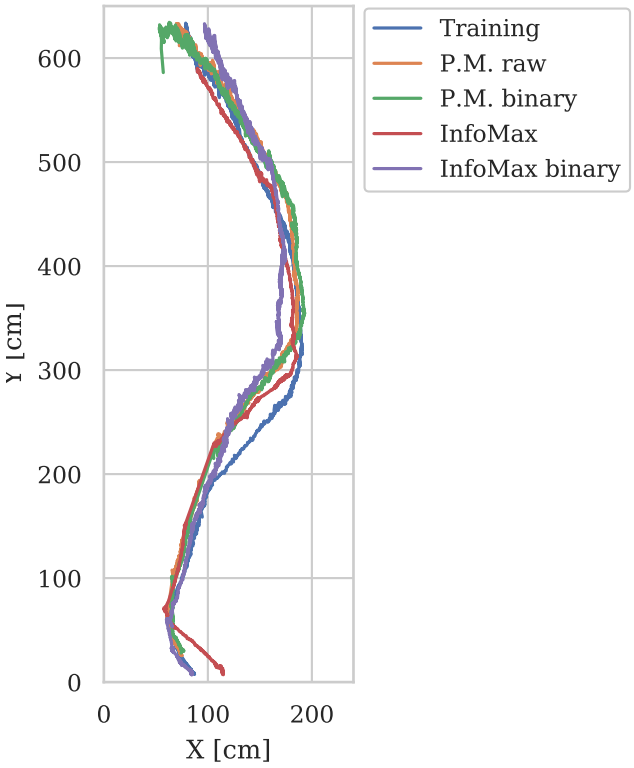


Figure 8: Reconstructed paths of autonomous robot during training and testing using each algorithm.