Lessons from reinforcement learning for biological representations of space

Alex Muryy\textsuperscript{a}, N. Siddharth\textsuperscript{b}, Nantas Nardelli\textsuperscript{b}, Andrew Glennerster\textsuperscript{a,}\textsuperscript{*}, Philip H. S. Torr\textsuperscript{b}

\textsuperscript{a}School of Psychology and Clinical Language Sciences, University of Reading, UK
\textsuperscript{b}Department of Engineering, University of Oxford, UK

Abstract

Neuroscientists postulate 3D representations in the brain in a variety of different coordinate frames (e.g. ‘head-centred’, ‘hand-centred’ and ‘world-based’). Recent advances in reinforcement learning demonstrate a quite different approach that may provide a more promising model for biological representations underlying spatial perception and navigation. In this paper, we focus on reinforcement learning methods that reward an agent for arriving at a target image without any attempt to build up a 3D ‘map’. We test the ability of this type of representation to support geometrically consistent spatial tasks, such as interpolating between learned locations, and compare its performance to that of a hand-crafted representation which has, by design, a high degree of geometric consistency. Our comparison of these two models demonstrates that it is advantageous to include information about the persistence of features as the camera translates (e.g. distant features persist). It is likely that non-Cartesian representations of this sort will be increasingly important in the search for robust models of human spatial perception and navigation.

Keywords: deep neural networks, 3D spatial representation, moving observer, navigation, view-based, parallax

*Corresponding author
Email address: a.glennerster@reading.ac.uk (Andrew Glennerster)
1. Introduction

The discovery of place cells, grid cells, heading direction cells, boundary vector cells and similar neurons in the mammalian hippocampus and surrounding cortex has been interpreted as evidence that the brain builds up an allocentric, world-based representation or ‘map’ of the environment and indicates the animals movement within it [1–4]. However, this interpretation is increasingly questioned and alternative models are proposed that do not involve a ‘cognitive map’ [5–7]. Computer vision and robotics provide a useful source of inspiration in the search for alternative models. Until recently, the predominant model for 3D navigation has been simultaneous localisation and mapping (SLAM) where a 3D reconstruction of the scene and the agents location within it are continually updated as new sensory information is received [8, 9]. However, since the advent of deep neural networks, there has been a move to try out a quite different approach to navigation, in which the agent is tasked with matching a particular camera pose (i.e. a image, not a 3D location), rewarding, however sparsely, the actions that lead it on a path to that goal. Eventually, after many trials, the agent learns to take a sequence of actions (‘turn left’, ‘turn right’, ‘go forward’) that take it from the current image to the goal although it never builds a map in the sense of a representation of the scene layout with an origin and coordinate frame.

In this paper, we explore in detail one of the key papers in this emerging field of reinforcement learning (RL) for navigation, Zhu et al. [10]. We illustrate what the system has learned by relating the stored vectors in the representation to the agent’s location and orientation in space, discovering that the contexts the representation recognises are heavily dependent on the agent’s current goal. This makes the representation quite different from a ‘cognitive map’ as envisaged by many neuroscientists and, instead, makes this a promising type model for biological spatial representation, in accordance with increasing evidence that human representation of 3D scenes is influenced by the observer’s goal and is not a task-independent reconstruction or ‘map’ [6, 7, 11–13].
Zhu et al. [10] showed that reinforcement learning could be applied successfully to a navigation task in which the agent was rewarded for arriving at a particular image (i.e. a given location and pose of the camera, although these 3D variables were not explicitly encoded in the input the agent received, only the current image and the goal image). Since then, various papers have built and expanded on this early result. For example, Mirowski et al. [14] adapted this method to cover much larger spatial regions using images from Google StreetView. Eslami et al. [15] have shown that behaviour one might have thought would require a 3D model (e.g. predicting a novel view from a novel location in a novel scene) can be learned by carrying out the same task in many similar scenes. Other approaches have resorted to an explicit use of \((x,y)\) coordinates in the stored memory, including Mirowski et al. [14], Chen et al. [16], Gupta et al. [17], Mirowski et al. [18], Kumar et al. [19], and Kantitscheider and Fiete [20], who have built on the biologically-inspired (but allocentric, coordinate-based) RatSLAM model of Milford and Wyeth [21]. In contrast to these coordinate-based advances, progress since Zhu et al. [10] on pure image-based approaches to large-scale spatial representation for navigation has slowed, as the community has been primarily focused on improving the visual navigation testbeds [22]. Another interesting paper, incorporating an explicit biological perspective in relation to navigation is Wayne et al. [23]. They have shown the importance of storing in memory, not the feature vectors derived from raw sensory input, but “predictions that are consistent with the probabilities of observed sensory sequences from the environment”. They use a Memory Based Predictor to generate such predictions and claim that this strategy is closely analogous to the operation of the hippocampus, which acts as an auto-association network with similar consequences.

Of particular interest amongst recent work exploring reinforcement learning (RL)-based navigation, is work relating to perceptual-goal-driven navigation [24–30]—where the goal to achieve is defined in terms of what an agent “sees” (images). Here, we restrict ourselves to the case where the goal is defined in terms of a particular view (location, bearing) in the environment, similar
to view-based navigation in biology [31]. Specifically, we examine the feature vectors in the stored representation after learning in the Zhu et al. [10] study to explore the extent to which they reflect the spatial layout of the scene. We show that, although spatial information is present in the representation, sufficient to be decoded, the organisation of the feature vectors is dominated by other factors such as the goal and the orientation of the camera (as one might expect, given the inputs to the network) and that it is possible to use these feature vectors to carry out simple spatial tasks such as interpolating between two learned locations.

We compare the performance of this RL network to a hand-crafted representation that is designed to display geometric consistency. The purpose of this comparison is to show how, in principle, a representation that does not include a 3D coordinate frame could, nevertheless, contain information to separate out the features in the image are likely to persist as the agent moves (translates) and those are likely to be more transient. In the search for a realistic, implementable model of the representation underlying human navigation, models of this sort are likely to be far more relevant than SLAM-like reconstructions of a scene.

2. Methods

Our goal is to compare performance of two algorithms, one based on a learned representation, developed by Zhu et al. [10], and one based on a hand-crafted representation. To analyze these methods, we use two different tasks: the first is to find the mid-point in space between two locations that have been learned (or are ‘known’) already; the second is to do the same in the orientation domain, i.e. to find the mid-bearing between two ‘known’ bearings. These tasks test for geometric consistency within a representation. We also probe the representations more directly, looking for systematic spatial organisation in the arrangement of the learned feature vectors when related to corresponding locations in space.

We begin with an account of the contribution Zhu et al. [10] make in the con-
text of reinforcement learning and describe how decoding can be used to query the information stored in the network. We then describe our hand-crafted representation which records information about the angles between pairs of visible points and about the extent to which these change as the optic centre translates. It is hardly a surprise that this representation performs well on geometric tasks, and we are not making a claim that this representation is in any sense ‘better’ than the learnt one - the representations are, after all, utterly different. Nevertheless, it is informative to compare the performance of the representations side-by-side in order to inform the debate about more biologically-plausible learned representations that are not based on a 3D coordinate frame.

2.1. Reinforcement Learning for Visual Navigation

Reinforcement learning (RL) [32] is a framework for optimizing and reasoning about sequential decision making. Tasks are modelled as Markov Decision Processes (MDPs), \( \langle S, A, T, R, \gamma \rangle \) tuples where \( S \) represents the state space, \( A \) the set of actions, \( T : S \times A \times S \to [0, 1] \), \( R \) the reward function \( R : S \times A \times S \to \mathbb{R} \), and \( \gamma \in [0, 1) \) a discount factor. Solving an MDP is defined as finding a policy \( \pi(a|s) = p(A = a|S = s) \) that maximizes the expected discounted cumulative return \( \sum_{k=0}^{\infty} \gamma^k r_{k+1} \).

Deep reinforcement learning (DRL) is an extension of standard RL in which the policy is approximated by a Deep Neural Network, and where RL algorithms are combined with stochastic gradient descent to optimise the parameters of the policy. Popular instances of DRL methods include: Deep Q-Network (DQN) [33] and its variants [34], which regress a state-action value function; policy gradient methods, which directly approximate the policy [35], and actor-critic methods [36, 37], which combine value-based methods with policy gradient algorithms to stabilize the training of these policies. DRL methods have been successful in solving complex tasks such as Go and other popular board games [38, 39], and have proved to be necessary to tackle decision making tasks with high-dimensional or visual state representation [33, 40]. These breakthroughs in visual learning and control have also created a surge in work on active vision [41],
and several visual-based navigation frameworks have recently been proposed to formalize and tackle many 3D navigation tasks.

We focus here on the task of goal-driven visual navigation, where the agent is asked to navigate to an entity in a high-fidelity 3D environment, given either an image of the entity, a natural language description, some coordinates, or other relevant information. Of particular interest is the framework proposed by Zhu et al. [10], which aims to solve the problem of learning a policy conditioned on both the target image and the current observation. The architecture is composed as follows: the observation and target images are first separately passed through a set of siamese layers [42], based on pretrained ResNet-50 network and a feedforward layer, embedding these images into the same embedding space. These embeddings are then concatenated and further passed through a fusion layer, which outputs a joint representation of the state. The joint representation is finally sent to scene-specific feedforward layers, which produce a policy output and a value as required by a standard actor-critic model. This split architecture allows for the embedding layers to focus on providing a consistent representation of the MDP instance based on the goal and the agent’s observation, while providing capacity to the network to create separate feature filters that can condition on specific scene features such as map layouts, object arrangement, lighting, and visual textures, thus obtaining the capability to arbitrarily generalize across many different scenes.

2.2. Knowledge Decoder

The architecture described in the previous section however does not provide any indication of whether any of the environment properties relevant to the task (e.g., location/orientation of the target, agent position, angles to the target) are present in the transformations encoded in the network’s weights.

To check whether any such information is encoded, we train a decoder which takes the agent’s internal representation as input and outputs one of the desired properties, such as e.g., \((x, y)\) coordinates of a chosen observation. More specifically, to build the dataset we use Zhu et al. [10]’s architecture trained on
AI2-THOR [43], generating an embedding up till the final feedforward layer (before it gets sent into the policy and value heads) for each target-observation pair of the training set, while also recording the agent’s \((x, y)\) coordinates and angle \(\theta\). The decoder is composed of two fully connected linear layers in the case of a single value regressor (i.e., the angle), or three linear layers, one shared and two heads, when regressing to \((x, y)\) coordinates. We use an MSE loss trained with Adam [44], together with dropout (see Table A1 for hyperparameters).

2.3. Relative Visual Direction (RVD) representation

Figure 1: A) 2D scene containing \(N\) random points and camera \(C\) in the centre. The points are ordered clockwise in angular sense with respect to the reference point \(P_1\), which is marked red. B) Angular and parallax features. \(P_i\) and \(P_j\) are scene points, \(C\) is camera location, \(C^1 - C^6\) are sub-cameras.

Figure 1a shows a 2D scene containing an optic centre \(C\), surrounded by \(N\) random points \(P_1, \ldots, P_N\). The points in the scene are numbered and ordered clockwise with respect to the first point \(P_1\), marked in red (this is relevant for the task shown in Fig. 4). The angle subtended by a pair of points \((P_i, P_j)\) at the optic center \(C\), indicating the relative visual direction, is denoted \(\alpha_{ij} = \angle CP_i P_j\) (such that \(\alpha_{ji} = \angle CP_j P_i = 2\pi - \alpha_{ij}\)). The vector of all such angles, between every possible pair of points \(P_i\) and \(P_j\) viewed from the optic center \(C\) is
denoted $\boldsymbol{\varepsilon}$ (Fig. 1b). We assume an omnidirectional view with no occlusions. The dimensionality of $\boldsymbol{\varepsilon}$ is thus $M = N^2 - N$, since we exclude angles between a point and itself. The elements of $\boldsymbol{\varepsilon}$ are ordered in a particular way, following

$$\boldsymbol{\varepsilon} = \{\alpha_{ij} : i = 1, \ldots, N, j = (i + 1), \ldots, N, 1, \ldots, (i - 1)\}. \quad (1)$$

An illustration of the magnitude of elements in $\boldsymbol{\varepsilon}$ is shown in Fig. 4d. The reason $\boldsymbol{\varepsilon}$ is ordered in such a manner is to assist in extracting subsets of elements when the task relates to visual direction (see Section 2.5). However, in the following section, we order elements of $\boldsymbol{\varepsilon}$ using parallax instead, since the task involves estimating the translation of the camera.

2.4. Mid-point for translation of the camera

Although $\boldsymbol{\varepsilon}$ contains all possible angular features, for certain tasks such as interpolating between locations (Fig. 3) some angular features are more informative than others. In particular, angular features that arise from pairs of distant points are more stable (i.e. vary less) during translation of the optic centre and thus are more useful for the interpolation task than are the angles between nearby points since these vary rapidly with optic centre translation. First, we will extract a subset of the elements of $\boldsymbol{\varepsilon}$ using a criterion based on parallax information. We define a measure of parallax that assumes we have access to more views of the scene, as if the camera has moved by a small amount as shown in Fig. 1b. For such individual ‘sub-cameras’ $C^k$, $k = 1, \ldots, n_C$, where $n_C$ is the number of sub-cameras, we can construct angular feature vectors $\boldsymbol{\varepsilon}_{C^k}$ similar to that constructed at the optic centre, $\boldsymbol{\varepsilon}$, with exactly the same ordering of elements. A ‘mean parallax vector’, $\boldsymbol{\psi}$, can then be computed from the difference between these sub-camera views $C^k$ and the original view at $C$.

$$\boldsymbol{\psi} = \{\psi_n\}_{n=1}^N = \frac{1}{n_C} \sum_{k=1}^{n_C} \frac{\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_{C^k}}{\boldsymbol{\varepsilon}} \quad (2)$$

Since $\boldsymbol{\psi}$ has the same ordering of elements as $\boldsymbol{\varepsilon}$, each element of $\boldsymbol{\psi}$ contains a parallax-related measure referring to that particular pair of points.
It will prove useful to identify the pairs of points that are more distant, using the observation that the parallax values recorded in $\psi$ are small in these cases. For a particular threshold value $T_\psi$ on parallax, we define $\rho$ as the mask on $\psi$, such that $\rho_i = 1$, if $\psi_i \leq T_\psi$, to identify the subset of $\varepsilon$ with relatively small parallax values—elements that are likely to change relatively slowly as the camera moves over larger distances—as $\varepsilon \oslash \rho$.

Figure 3b shows a random 2D scene with a 6×6 grid of cameras in the middle. For each camera $C_j, \ j = 1, \ldots, 36$, we calculated a feature vector $\varepsilon_{C_j}$ and a parallax mask $\rho_{C_j}$ as for Fig. 2. We estimated the mid-points between all possible pairs of the 36 cameras (n=630) as follows. The feature vector for the mid-point between two cameras $C_i$ and $C_j$ was computed as $\varepsilon_{C_i,C_j} = \frac{1}{2}\left( (\varepsilon_{C_i} \oslash \rho_{C_i}) + (\varepsilon_{C_j} \oslash \rho_{C_j}) \right)$.

Then, to find the midpoint, we searched over a fine regular grid (step = 1) of camera locations to find the camera $C_k^*$ that was best matched with the estimated feature $\varepsilon_{C_i,C_j}$, that is,

$$C_k^* = \arg \min_{C_k} \| \varepsilon_{C_i,C_j} - \varepsilon_{C_k} \|$$

2.5. Mid-bearing for rotation of the camera

We now consider a task of interpolating between camera bearing (viewing direction), rather than location. The goal is to estimate a bearing that is half way between two given views of the camera. A view, $\vartheta^{\theta,\omega}$, in this context, involves both a bearing, $\theta$, and an angular range, $\omega$, —typically taken to be a fixed value— specifying the field of view for that camera. $\vartheta^{\theta,\omega}$ consists of all the elements of $\varepsilon$ that appear within that field of view. Note that the goal here is closely related, but not identical, to the task of picking out an entire view that is half-way between two given views. The way $\varepsilon$ is organised, such that elements are ordered by reference point (see Eqn. 1), means that there is a consistent (albeit approximate) relationship between the index of the element and the bearing of the reference point for that element.

To consider all the elements of $\varepsilon$ that appear in a given view we construct a mask, $\kappa$, similar to $\rho$ above, but now the mask is based on whether both scene
points $P_i$ and $P_k$ that define an element in $\varepsilon$ are visible in a particular view: $\kappa^{\theta,\omega} = \{\kappa_j\}_{j=1}^N$, where $\kappa_j = 1$, if $P_i, P_k \in \vartheta^{\theta,\omega}$, where $\varepsilon^j = \alpha_{ik} = \angle P_i P k$. The relevant elements are denoted $\varepsilon \odot \kappa^{\theta,\omega}$.

Given two such views $\vartheta^{\theta_i,\omega_i}$ and $\vartheta^{\theta_j,\omega_j}$, we can use the indices of the elements in each view to estimate the indices of the view that is mid-way between the two. Figures 4b and 4d shows how the bearing of a mid-view ($\theta_{mid}$) is estimated using the simplistic assumption that the bearing of the reference point in a pair of views varies linearly with index in $\varepsilon$. The mean index of a view $\vartheta^{\theta,\omega}$ is computed from its corresponding mask $\kappa^{\theta,\omega}$ as the middle index $\mu$ of all “on” mask elements, $\kappa_j = 1$, for that view. Given two views $\vartheta^{\theta_i,\omega_i}$ and $\vartheta^{\theta_j,\omega_j}$, we estimate a nominal bearing of the mid-view image from the average of their mean indices:

$$\mu_{mid} = (\mu_i + \mu_j)/2.$$  \tag{4}

and $\theta_{mid} \propto \mu_{mid}$, as illustrated in Fig. 4d.

3. Results

Figure 2 compares the representation of a scene in the two models we have discussed, based on Zhu et al. [10] (left hand column) or relative visual direction (RVD, right hand column). Figure 2a shows a plan view of the scenes used by Zhu et al. [10] (where filled and closed symbols show the camera locations at test and training) and Fig. 2b shows the 2D layout of scene points (black dots) and camera locations (coloured points) in a synthetic 2D scene that was used as input for the RVD method. In Zhu et al. [10], the environment was a highly realistic 3D scene in which the agent was allowed to make 0.5m steps and turn by 0, +90 or -90 degrees (figures are from the Bathroom scene, see Appendix for others). Target views are marked by blue stars. For the RVD method, we generated a random 2D scene with 100 points. Cameras were placed in the middle of the scene as a regular $50 \times 50$ grid, which occupied 1/5 of the scene (Fig. 2b). The colour indicates the distance of a camera from the central reference camera.
For each learned context in Zhu et al. [10] (where a learned context is defined by an observation location, a camera orientation and a target), there is a corresponding feature vector (i.e. 20 feature vectors per location). Figure 2c shows the Euclidean distance between pairs of feature vectors ($\mathbb{R}^{512}$) from the test set, for all possible pairings, and plots this distance against the distance between the corresponding observation locations ($\mathbb{R}^{2}$). Figure 2c shows that there is only a weak correlation between distance in the embedding space and physical distance between observation locations for this scene in the Zhu et al. [10] paper (Pearson correlation coefficient, $R$, is 0.09, see Fig. A3 for other scenes) whereas Fig. 2d shows that, for the RVD method, there is a clear positive correlation ($R = 0.99$). Zhu et al. [10] quoted a correlation of 0.62 between feature vector separation and separation in room space, but we are only able to reproduce a similarly high correlation by considering the distance between pairs of feature vectors when the agent had the same goal and the same viewing direction ($R = 0.67$ for all such pairings in the Bathroom scene). By contrast, Fig. 2c refers to all possible pairings in the test phase.

The right hand column of Fig. 2 shows results of the ‘relative visual direction’ (RVD) model. At each camera camera location ($N = 2500$), we generated a truncated angular feature vector $\mathbf{\varepsilon} \odot \mathbf{\rho}$ as a representation of the scene as viewed from that location. We used the 30$^{th}$ percentile of the parallax values as a threshold for inclusion of elements ($T_{\psi}$), i.e. the truncated feature vectors contained only the elements of $\mathbf{\varepsilon}$ that corresponded to pairs of points with the smallest parallax values (see Section 2.4), where ‘small’ in this case means the bottom 30% when ordered by parallax magnitude. Note that we have used the same ordering of elements in $\mathbf{\varepsilon} \odot \mathbf{\rho}$ for all cameras. Specifically, the ordering of $\mathbf{\varepsilon}$ and $\mathbf{\rho}$ were established for the central reference camera and applied to all other cameras (see Eqn. 1).

Figures 2e and 2f visualise the embedding space for the Zhu et al. [10] and RVD representations respectively using a t-SNE projection [45]. This projection attempts to preserve ordinal information about the Euclidean distance between high dimensional vectors when they are projected into 2-D. In Zhu et al. [10]
(Fig. 2e), feature vectors are clumped together in the t-SNE plot according to the agent’s target image. Targets 4 and 5 were very similar images so it is understandable that the feature vectors for locations with these targets are mixed (yellow and orange points). Although target is the dominant determinant of feature vector clustering, information about camera orientation and camera location is still evident in the t-SNE plot. The top-right sub-panel colour-codes the same T4/T5 cluster but now according to the orientation of the camera; this shows that orientation also separates out very clearly. Finally, there is also information in the t-SNE plot about camera location. Colours in the bottom right subplot indicate distance of the camera location from a reference point, (0,0); there is a gradation of colours along strips of a common camera orientation and this systematic pattern helps to explain why camera location can be decoded (see Fig. 3c). For the RVD method, the configuration of feature vectors preserves the structural regularity of the camera positions, as can be seen from the t-SNE projection in Fig. 2f. We now explore how these differences affect the ability of each representation to support interpolation between learned/stored locations.

Figure 3 shows the results of the location interpolation task which was to estimate the mid-point between two locations (e.g. in Fig. 3a \( \text{O}_{\text{mid}} \) is halfway between \( \text{O}_1 \) and \( \text{O}_2 \)) based on the midpoint between two feature vectors. For the Zhu et al. [10] model, this requires a decoder for 2-D position learned from the stored feature vectors (see Section 2.2). The results are shown in Fig. 3c using a normalized scale to illustrate the errors relative to the two input locations. For the RVD model, decoding is much more direct, as one would expect from the t-SNE plot (Fig. 2f). Figure 3d shows estimated mid-points calculated this way. Figure 3e shows the absolute errors relative to the true mid-point for the Zhu et al. [10] method as a function of separation between \( \text{O}_1 \) and \( \text{O}_2 \). Figure 3f shows the same for the RVD method: this is more accurate than the Zhu et al. [10] model and, as expected from any geometrically consistent representation, has a monotonic rise with separation between stored locations.

Figure 4a shows the scene layout from Zhu et al. [10] and two views from a single location (orange and purple). The goal in this case is to predict a view
with an intermediate bearing half way between these two (as shown by the black arrow). Figure 4c shows the error in the decoded mid-bearing when the input images are taken from views that are 0, 90 or 180° apart. Note that the two input images need not necessarily be taken from the same location in the room (either in training the decoder or in recovering a mid-bearing). Figures 4c and 4e show that there is no systematic bias to the mid-bearing errors but the spread of errors is large compared to that for the ‘relative visual direction’ (RVD) method (Fig. 4f). The RVD method uses a very simple algorithm to estimate the mean bearing. It assumes that the ordering of elements in $\varepsilon$ has a linear relationship to the bearing of a view, i.e. that as the bearing changes (going from orange view to purple view in Fig. 4d) the index of the corresponding elements in $\varepsilon$ will change systematically and hence the mean index of the elements within a view is useful in determining the bearing of that view. The fact that this is approximately the case is due to the way that the vector, $\varepsilon$, was set up in the first place (Eqn. 1). In a little more detail, Fig. 4b shows two input view-directions (orange and purple) and the ground-truth direction of the mid-view (black arrow). For the purposes of illustration only, Fig. 4d shows the $i^{th}$ element in the orange image (pair of dots outlined in orange) and the $i^{th}$ pair in the purple image (outlined in purple): Considering the indices of these two elements in $\varepsilon$, the rounded mean of these two indices corresponds to a pair of points, as shown by the black squares in Fig. 4d. In this case, the black points happen to lie close to the mid-bearing direction but this is not important (and is not always the case). The heuristic does not attempt to synthesise the mid-orientation image, it simply reports the estimated orientation of the mid-view (Fig. 4d) as described in Section 2.5. There is no systematic bias for either model (Fig. 4e) but the range of errors is considerably greater for Zhu et al. [10] than for the RVD model (Fig. 4f).
Figure 2: **Relationship between scene location and feature vectors for Zhu et al. [10] and the relative visual direction (RVD) method.**

a) shows a plan view of the Bathroom scene in Zhu et al. [10]. Open circles show the camera locations for images used in the training set, closed circles show the locations used in the test set. Blue stars and black arrows show the location and viewing direction of the camera for the target images.

b) An example of a random 2D scene with N=100 points used in the RVD model. Cameras are placed in the middle of the scene as a 50×50 grid, which is 1/5 of the scene. The colour of each camera location indicates the distance of the camera from the central camera, C. For each of the 2500 camera locations we calculated a vector, ε, describing the angle between pairs of scene points as viewed from that camera (see Methods).

c) For the Zhu et al. [10] method, the Euclidean distance in $\mathbb{R}^{512}$ between pairs of embedded feature vectors is plotted against the separation between the corresponding pairs of camera locations in the scene.

d) For the ‘relative visual direction’ method, the Euclidean distance between the feature vectors for each camera and the feature vector for the central camera, C is plotted against the separation between the corresponding pairs of camera locations in the scene.

e) A t-SNE plot that projects the stored feature vectors in the Zhu et al. [10] network into 2D (see text for details).

f) Same as e) but now for the RVD model.
Figure 3: **Estimate of midpoints between pairs of observation locations.** a) shows the Bathroom scene with two observation locations, $O_1$ and $O_2$ and a midpoint, $O_{mid}$. b) shows a random 2D scene with a $6 \times 6$ grid of cameras in the middle. For each camera we calculated a feature vector $\mathbf{e} \odot \mathbf{p}$ (see Section 2.4). c) shows the estimated midpoints using the decoding of Zhu et al. [10] feature vectors (see Section 2.2). Orange and purple circles show the normalised location of the two observation locations and the black dots show, in this normalised coordinate frame, the location of the estimated midpoint for all possible pairs of observations (where an observation is defined as a location, orientation and target). d) shows the same as c) but for the feature vectors in the RVD model. The black dots show midpoints for all possible pairs of camera locations. e) shows the midpoint prediction error from c) (absolute errors) plotted against the separation of the observation locations ($O_1$ and $O_2$). The separation between observation locations is normalised by the maximum possible location of two observation locations in the room. Error bars show one standard deviation. f) shows the same for the RVD method. We considered all possible pairs of cameras (n=630).
Figure 4: **Estimate of new views at an orientation halfway between learned views.**

a) shows a plan view of a bathroom scene in Zhu et al. [10] and the 45 locations the camera could occupy. Orange and purple arrows indicate two camera orientations and the black arrow indicates an orientation halfway between these (not used in Zhu et al. [10]).

b) Similar to a) but for the RVD method. Points visible in views 0° (north) and 135° (south-east) are marked as orange and blue symbols, where the field of view (ω) is limited to 90°. The ground-truth mid-view is indicated by the black arrow (see text).

c) Distribution of errors in computing the mid-view orientation from a decoding of orientation in the Zhu et al. [10] trained network. Red, green and blue distributions are for camera orientations separated by 0, 90 and 180 degrees respectively.

d) Full vector of angular features, ε, (black saw-tooth plot). The y-axis shows the magnitude the elements in ε, i.e. the angle between pairs of points. The x-axis represents indices of the vector’s elements (9900 in this case) see Eqn. 1. The x-axis also provides an approximate indication of visual direction, from 0° to 360°, see text. The elements that correspond to points visible in the north and south-east views are marked as orange and purple respectively. Mid-indices μ1 and μ2 are marked as orange and purple arrows, while the index of the predicted mid-view μmid is marked as a black arrow.

e) All the mid-view errors for the Zhu et al. [10] method for camera orientations separated by 0, 90 and 180 degrees. Mean and standard error shown in blue. Plot below shows the same for the RVD method. Mean and standard error shown in red.

f) Shows the RMSE error of predicted mid-view with respect to the ground truth as a function of angular separation between the views. For the RVD method we considered views separated by many different angles (in increments of 10°), while for Zhu et al. [10] the data limited us to only three separations. RMSE error for the RVD method is considerably smaller than for Zhu et al. [10].
4. Discussion

There has been a long-standing assumption that the brain generates 3D representations in a variety of coordinate frames including eye-centred (V1), ego-centred (parietal cortex) and world-centred (hippocampus and parahippocampal gyrus) coordinates. 3D navigation and computer vision research have also concentrated on algorithms that generate 3D representations (SLAM) [8]. This paper has focused instead on the implications of the recent advances in reinforcement learning where representations for navigation are developed through exploration and reward, not geometry. It has exciting implications for understanding the mechanisms underlying biological navigation because the method of learning and storing representations are likely to be much more similar to the ones animals employ than the computations that are used in SLAM-type algorithms.

We have chosen to examine in detail the RL method described by Zhu et al. [10] because this has now become a general method on which several more recent and complex algorithms have been based [14, 16–19, 46]. We have compared the Zhu et al. [10] representation to a hand-crafted representation (based on relative visual directions and using highly simplistic input) in order to illustrate two particular points. First, in Zhu et al. [10], the relationship between stored feature vectors and the locations of the camera in the scene (Fig. 2a) is quite a complex one, while for the RVD model the relationship is simple and transparent. In the case of Zhu et al. [10], it is possible to build a decoder to describe the mapping between feature vectors and location (as illustrated by the systematic distance information visible in Fig. 2e) but this is quite different from the smooth, one-to-one relationship between stored feature vectors and space illustrated in Fig. 2f, at least over the range of camera locations illustrated here (Fig. 2b)). This is relevant in biology since any motor output that depends on spatial location would need input that was in a form that could be interpolated: interpolation between stored inputs for generating motor outputs has been discussed in relation to models of the cerebellum [47] and evidence for
it shown in human behaviour [48]. Clearly, interpolation of the outputs of the RVD model can generate a sensible result (e.g. Fig. 3d) whereas interpolation of the output of the Zhu et al. [10] model Fig. 2e cannot without prior decoding of the information stored in the representation and even then the results are relatively erratic.

The second conclusion is that, in order to achieve the regularity of Fig. 2f and the consequent precision in interpolating between spatial locations with the ‘relative visual direction’ (RVD) method (Fig. 3f), it was necessary to use a subset of elements of the stored feature vectors (a lower-dimensional vector). Motion parallax was critical in determining which elements to use. Essentially, we have ensured that elements whose values change slowly as the camera translates are used in Figs. 2f and 3f, while elements that change rapidly are excluded. One can imagine situations in which the filtering is reversed and only those elements that change rapidly as the camera translates are included for a particular task, for example a visual servo mechanism for stabilising the camera’s location (see Fig. 5).

Figure 5: Visual servo-ing to maintain postural stability Looking straight out on this scene, almost all motion parallax is removed and so cannot drive postural reflexes. Usually, there are objects visible at a range of distances giving rise to both large and small magnitudes of motion parallax. This range is useful for guiding different tasks (see text). License to use Creative Commons Zero - CC0.
Answering the question ‘where am I?’ does not necessarily imply a coordinate frame [6, 7, 49–51]. Instead, one can offer a restricted set of alternative hypotheses. These potential answers to the question may correspond to widely separated locations in space, in which case the catchment area of each hypothesis is large, but the answer can be refined by adding more alternatives (i.e. more specific hypotheses about where the agent is). This makes the representation of space hierarchical [52, 53] and compositional. The sorting of feature vectors according to motion parallax information, as we have done here, offers a way to encode information in a hierarchical way. The main message from the comparison of Zhu et al. [10] and RVD models is that it is advantageous for an algorithm to separate out those visual features that are likely to be long- or short-lived in the scene as the camera translates. Neuroscientists will no doubt follow with interest the development of RL models of navigation develop, as they are likely to have an important influence in the debate over the nature of biological representations of space.

Acknowledgements

This research was supported by EPSRC/Dstl grant EP/N019423/1 (AG). We are grateful to Abhinav Gupta for providing code and advice and to Aidas Kilda and Andrew Gambardella for their help. PHST was additionally supported by ERC grant ERC-2012-AdG 321162-HELIOS and EPSRC grant Seebibyte EP/M013774/1, and would also like to acknowledge the Royal Academy of Engineering and FiveAI.

References

References

[1] J. O’Keefe, J. Dostrovsky, The hippocampus as a spatial map. Preliminary evidence from unit activity in the freely-moving rat, Brain Research 34 (1971) 171–175.
[2] S. Taube, U. Muller, B. Ranck, Head-direction cells recorded from the post-subiculum in freely moving rats. I. Description and quantitative analysis., The Journal of Neuroscience 10 (1990) 420–435.

[3] Microstructure of a spatial map in the entorhinal cortex., Nature 436 (2005) 801–6. URL: http://www.ncbi.nlm.nih.gov/pubmed/15965463.

[4] C. Lever, S. Burton, A. Jeewajee, J. O’Keefe, N. Burgess, Boundary vector cells in the subiculum of the hippocampal formation, Journal of Neuroscience 29 (2009) 9771–9777.

[5] L. Acharya, Z. M. Aghajan, C. Vuong, J. J. Moore, M. R. Mehta, Causal influence of visual cues on hippocampal directional selectivity, Cell 164 (2016) 197–207.

[6] W. H. Warren, Non-euclidean navigation, Journal of Experimental Biology 222 (2019) jeb187971.

[7] A. Glennerster, A moving observer in a three-dimensional world, Phil. Trans. R. Soc. B 371 (2016) 20150265.

[8] A. J. Davison, Real-time simultaneous localisation and mapping with a single camera, in: ICCV, 2003, pp. 1403–1410.

[9] J. Fuentes-Pacheco, J. Ruiz-Ascencio, J. M. Rendón-Mancha, Visual simultaneous localization and mapping: a survey, Artificial Intelligence Review 43 (2015) 55–81.

[10] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, A. Farhadi, Target-driven visual navigation in indoor scenes using deep reinforcement learning, in: Robotics and Automation (ICRA), 2017 IEEE International Conference on, IEEE, 2017, pp. 3357–3364.

[11] A. Glennerster, B. J. Rogers, M. F. Bradshaw, Stereoscopic depth constancy depends on the subject’s task 36 (1996) 3441–3456.
[12] M. F. Bradshaw, A. D. Parton, A. Glennerster, The task-dependent use of binocular disparity and motion parallax information. 40 (2000) 3725–3734.

[13] J. B. Smeets, E. Brenner, Grasping weber’s law, Current Biology 18 (2008) R1089–R1090.

[14] P. Mirowski, M. Grimes, M. Malinowski, K. M. Hermann, K. Anderson, D. Teplyashin, K. Simonyan, A. Zisserman, R. Hadsell, et al., Learning to navigate in cities without a map, in: Advances in Neural Information Processing Systems, 2018, pp. 2419–2430.

[15] S. A. Eslami, D. J. Rezende, F. Besse, F. Viola, A. S. Morcos, M. Garnelo, A. Ruderman, A. A. Rusu, I. Danihelka, K. Gregor, et al., Neural scene representation and rendering, Science 360 (2018) 1204–1210.

[16] T. Chen, S. Gupta, A. Gupta, Learning exploration policies for navigation, arXiv preprint arXiv:1903.01959 (2019).

[17] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, J. Malik, Cognitive mapping and planning for visual navigation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2616–2625.

[18] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, et al., Learning to navigate in complex environments, arXiv preprint arXiv:1611.03673 (2016).

[19] A. Kumar, S. Gupta, J. Malik, Learning navigation subroutines by watching videos, arXiv preprint arXiv:1905.12612 (2019).

[20] I. Kanitscheider, I. Fiete, Training recurrent networks to generate hypotheses about how the brain solves hard navigation problems, in: Advances in Neural Information Processing Systems, 2017, pp. 4529–4538.

[21] M. Milford, G. Wyeth, Persistent navigation and mapping using a biologically inspired slam system, The International Journal of Robotics Research 29 (2010) 1131–1153.
[22] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, et al., Habitat: A platform for embodied ai research, arXiv preprint arXiv:1904.01201 (2019).

[23] G. Wayne, C.-C. Hung, D. Amos, M. Mirza, A. Aluja, A. Grabska-Barwinska, J. Rae, P. Mirowski, J. Z. Leibo, A. Santoro, et al., Unsupervised predictive memory in a goal-directed agent, arXiv preprint arXiv:1803.10760 (2018).

[24] P. Sermanet, K. Xu, S. Levine, Unsupervised perceptual rewards for imitation learning, CoRR abs/1612.06699 (2016). URL: http://arxiv.org/abs/1612.06699.

[25] A. Singh, L. Yang, K. Hartikainen, C. Finn, S. Levine, End-to-end robotic reinforcement learning without reward engineering, CoRR abs/1904.07854 (2019). URL: http://arxiv.org/abs/1904.07854.

[26] A. D. Edwards, Perceptual Goal Specifications for Reinforcement Learning, Ph.D. thesis, PhD thesis, Georgia Institute of Technology, 2017.

[27] W. Yang, X. Wang, A. Farhadi, A. Gupta, R. Mottaghi, Visual semantic navigation using scene priors, CoRR abs/1810.06543 (2018). URL: http://arxiv.org/abs/1810.06543.

[28] Y. Zhu, D. Gordon, E. Kolve, D. Fox, L. Fei-Fei, A. Gupta, R. Mottaghi, A. Farhadi, Visual semantic planning using deep successor representations, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 483–492.

[29] V. Dhiman, S. Banerjee, B. Griffin, J. M. Siskind, J. J. Corso, A critical investigation of deep reinforcement learning for navigation, CoRR abs/1802.02274 (2018). URL: http://arxiv.org/abs/1802.02274.

[30] P. Anderson, A. X. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, A. R. Zamir, On
evaluation of embodied navigation agents, CoRR abs/1807.06757 (2018).
URL: http://arxiv.org/abs/1807.06757. arXiv:1807.06757.

[31] B. Cartwright, T. Collett, Landmark Learning in Bees - Experiments and Models, Journal of Comparative Physiology 151 (1983) 521–543.

[32] R. S. Sutton, A. G. Barto, Reinforcement learning: An introduction, MIT press, 2018.

[33] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (2015) 529.

[34] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, D. Silver, Rainbow: Combining improvements in deep reinforcement learning, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018, pp. 3215–3222.

[35] R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour, Policy gradient methods for reinforcement learning with function approximation, in: Advances in neural information processing systems, 2000, pp. 1057–1063.

[36] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, Deterministic Policy Gradient Algorithms, in: International Conference on Machine Learning, 2014, pp. 387–395.

[37] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, in: International conference on machine learning, 2016, pp. 1928–1937.

[38] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al., Mastering the game of go without human knowledge, Nature 550 (2017) 354.
[39] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al., A general reinforcement learning algorithm that masters chess, shogi, and go through self-play, Science 362 (2018) 1140–1144.

[40] S. Levine, C. Finn, T. Darrell, P. Abbeel, End-to-end training of deep visuomotor policies, The Journal of Machine Learning Research 17 (2016) 1334–1373.

[41] J. Ruiz-del Solar, P. Loncomilla, N. Soto, A survey on deep learning methods for robot vision, arXiv preprint arXiv:1803.10862 (2018).

[42] S. Chopra, R. Hadsell, Y. LeCun, et al., Learning a similarity metric discriminatively, with application to face verification, in: CVPR (1), 2005, pp. 539–546.

[43] E. Kolve, R. Mottaghi, D. Gordon, Y. Zhu, A. Gupta, A. Farhadi, Ai2-thor: An interactive 3d environment for visual ai, arXiv preprint arXiv:1712.05474 (2017).

[44] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[45] L. v. d. Maaten, G. Hinton, Visualizing data using t-sne, Journal of machine learning research 9 (2008) 2579–2605.

[46] A. Chen, G. C. DeAngelis, D. E. Angelaki, Convergence of vestibular and visual self-motion signals in an area of the posterior sylvian fissure, Journal of Neuroscience 31 (2011) 11617–11627.

[47] N. R. Luque, J. A. Garrido, R. R. Carrillo, J.-M. C. Olivier, E. Ros, Cerebellar input configuration toward object model abstraction in manipulation tasks, IEEE Transactions on Neural Networks 22 (2011) 1321–1328.

[48] A. A. Ahmed, D. M. Wolpert, J. R. Flanagan, Flexible representations of dynamics are used in object manipulation, Current Biology 18 (2008) 763–768.
[49] S. Gillner, H. A. Mallot, Navigation and acquisition of spatial knowledge in a virtual maze., Journal of Cognitive Neuroscience 10 (1998) 445–463.

[50] A. Glennerster, M. E. Hansard, A. W. Fitzgibbon, View-based approaches to spatial representation in human vision, Lecture Notes in Computer Science 5064 (2009) 193–208.

[51] D. Rosenbaum, F. Besse, F. Viola, D. J. Rezende, S. Eslami, Learning models for visual 3d localization with implicit mapping, arXiv preprint arXiv:1807.03149 (2018).

[52] S. C. Hirtle, J. Jonides, Evidence of hierarchies in cognitive maps, Memory & Cognition 13 (1985) 208–217.

[53] J. M. Wiener, H. A. Mallot, 'fine-to-coarse’route planning and navigation in regionalized environments, Spatial Cognition and Computation 3 (2003) 331–358.
Appendix
Figure A1: Consequences of using large-parallax elements in the RVD model. 

a) re-plots the t-SNE projection of the RVD feature vectors from Fig 2f.
b) shows the disruption in the representation caused by using a different subspace of $\varepsilon$, namely picking out the elements of $\varepsilon$ that have the greatest magnitude of motion parallax (Eqn. 2) rather than the smallest parallax, as we have used in all the figures above. c) shows the effect of using all of $\varepsilon$ rather than a subspace. d), e) and f) show the distance between feature vectors plotted against distance to the central camera (see Fig. 2d) using the feature vectors illustrated in a), b) and c) respectively. g), h) and i) show the consequence of using the vectors illustrated in a), b) and c) for the mid-point task (so i is a repeat of Fig. 3d). j), k), l) show the magnitude of the midpoint errors, following Fig. 3f.
Figure A2: Plan views of all 4 scenes. a) bathroom, b) bedroom, c) kitchen, d) living room. Symbols are as for Fig. 2a.
Figure A3: Results for the bathroom scene were shown in Figs. 2 to 4 and are re-plotted here (left hand column). Results for the bedroom, kitchen and living room are shown in columns 2 to 4 respectively. In the top row, a-d), the correlations, R, are 0.088, 0.22, 0.24 and 0.14 respectively. For e-h) see Fig. 3c, for i-l) see Fig. 3e, and for m-p) see Fig. 4c.
Table A1: Hyperparameters for the original trained network and the two decoder networks. The original trained network from Zhu et al. [10] was used throughout the paper, e.g., the tSNE plot in Fig. 2e. The position decoder was used for the results shown in Fig. 3. The angle decoder was used for Fig. 4.