Deep Visual Teach and Repeat on Path Networks

Tristan Swedish
MIT Media Lab
tswedish@mit.edu

Ramesh Raskar
MIT Media Lab
raskar@mit.edu

Abstract

We propose an approach for solving Visual Teach and Repeat tasks for routes that consist of discrete directions along path networks using deep learning. Visual paths are specified by a single monocular image sequence and our approach does not query frames or image features during inference, but instead is composed of classifiers trained on each path. Our method is efficient for both storing or following paths and enables sharing of visual path specifications between parties without sharing visual data explicitly. We evaluate our approach in a simulated environment, and present qualitative results on real data captured with a smartphone.

1. Introduction

Visual navigation is a complex problem that inspires similarly complex solutions to its numerous challenges. Many methods require specialized vision hardware and significant computational resources. However, the problem of visual navigation can be simplified when autonomy in novel environments is not necessary. Methods known as Visual Teach & Repeat (VT&R) are able to repeatedly follow paths driven first by another expert (often a human). The VT&R paradigm has shown promise in applications where paths must be routinely followed, and expert control examples are available: intra-site delivery and patrol, tours, and return-to-site missions. In this paper, we propose a new technique using deep learning methods to solve VT&R type tasks in environments where directions can be described by discrete instructions: "forward", "left" or "right." We call this class of environments "path networks" since they can be represented as a set of paths connected at intersections.

VT&R does not require specialized hardware and has been shown to be effective in scenarios where a recording of odometry and visual observations along a specific path is available to the navigating agent. This paper explores a novel way to compactly represent these recordings within the weights of a neural network. We evaluate our technique for paths in sparsely connected path networks found predominately in the built environment (e.g. Indoor Floor Plans, Road Networks).

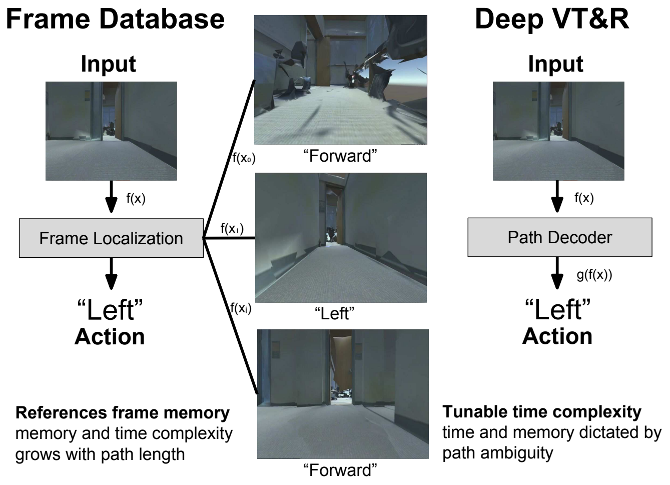


Figure 1: The proposed system predicts the necessary actions to follow a visual path from an image sequence. Our method does not refer to the image sequence at the time of inference, instead using a compact representation.

The motivation for our approach is the observation that deep learning methods are effective at solving challenging inference problems with tunable computational capacity. Computational complexity, such as the memory and operations required, can be controlled by the capacity of the neural network architecture. When compared to more traditional VT&R, our deep learning based approach allows for greatly reduced memory requirements during the "repeat" phase (Table 1). In other words, the trained model parameters contain all the information needed to follow the path. This memory reduction may also be advantageous when sharing path representations over low-bandwidth networks.

Data driven methods are also well suited for the challenges posed by visual navigation. For example, monocular visual odometry (MVO) must overcome fundamental and practical ambiguities. The movement of a pinhole camera can only be estimated up to a relative scale since it is impossible to estimate distance without an established baseline. Practically, small field of view (FOV) sensors also make it difficult to determine if the camera is moving sideways or rotating. Limited FOV may also be more susceptible to perceptual aliasing, where identical observations of different hidden states confound estimation. In order to overcome these challenges, reasonable estimates can be made using priors learned from training data [18]. This enables a system to infer the distance of certain objects in a frame simply by knowing the size of similar objects (e.g. people, furniture, vehicles).

1.1. Contributions

We propose a novel method based on deep learning that seeks to solve VT&R type tasks, preserving the ability to follow a visual path without a graphical representation or frame feature database. We list our contributions for clarity:

- An unsupervised labeling scheme using monocular visual odometry for image sequences.
- Deep Visual Teach and Repeat framework for navigating in sparsely connected path networks.
- Demonstration of the framework for defining routes and following them in realistic environments.

1.2. Limitations

The proposed method does not perform visual servoing or correction to ensure that it stays on the intended path. This is not a serious problem, however, when paths are restricted to a set of simple intersections, what we refer to in this paper as sparsely connected path networks. We assume the navigating agent using the proposed system is able to interpret instructions like "forward" or "left" and is able to stay on track (such as staying centered walking down a hallway, or taking a left at an intersection) independently.

2. Related Work

2.1. Monocular Visual SLAM

The robotics and computer vision communities have studied mobile navigation for decades. Most robotic planning approaches are applied to some known map and the robot's pose within this map. This process is fused in SLAM, which has has been successfully applied in numerous applications, including indoor navigation and autonomous driving [32, 22, 17, 16]. Numerous approaches have been suggested that incorporate depth-sensitive sensors (stereo, RGBD, Lidar) [21, 36]. Recently, real-time monocular SLAM up to a scale factor has been demonstrated [7], as well as data-driven methods that estimate real scale [33, 29]. A comprehensive overview of SLAM methods is outside the scope of this work, we direct the reader to an introductory text [34] or an overview of the state of the art [5].

However, monocular visual SLAM can suffer from poor scalability, the search space grows with the number nodes in the pose graph [6], incurring large memory and computational costs in increasingly connected and large environments.

2.2. Visual Teach & Repeat

Visual Teach and Repeat consists of a training or "teach" phase and an execution phase known as the "repeat" phase. In the teach phase, an image sequence and odometry information is recorded. In the repeat phase, the odometry information is used to replay the desired motion along the path, and the recorded image sequence is used to adjust the current viewpoint and estimated position along the path.

In previous VT&R applications, a robot or sensor platform is driven along a path by a human and then uses this example to follow the visual path in the future using replayed motor commands and a frame database to correct localization along the path [30, 8]. Recent work has shown applications to aerial vehicles and multiple paths through an environment [23, 35, 25]. The setting of VT&R differs from traditional SLAM in that maps may not attempt to preserve a global coordinate system, so accumulating errors from visual odometry are less likely to produce maps unsuitable for navigation.

Our approach distinguishes itself from from traditional VT&R because it is does not save frame features in a database, but instead encodes information from the teach phase directly in the neural network weights.

2.3. Topological Localization

Topological localization attempts to localize the position of the current frame to a prerecorded image sequence [13, 10]. A topological map recovers the local connectivity of the scene based on a sequence of measurements, detecting loop closures, and attempting to recover the global topological structure of the scene. Loop closure detection typically consists of a similarity metric applied to an image sequence and an input image with a minimum threshold that indicates a loop closure exists at that point in the image sequence. Recently, Convolutional Neural Networks (CNNs) have been used to construct robust representations which are compared using the cosine similarity or other distance metrics [14, 9]. CNNs have also been used to memorize 6DOF for specific image sequences labeled using structure from motion [15].

Visual place recognition tries to identify places from unstructured data such as images on the internet, but does not use these metrics for planning paths through an environment [19]. NetVLAD is a more recent approach using a special VLAD layer to refine image queries from a large

Table 1: Computational performance of the baseline and proposed method in experiments. In this table, D is the embedded feature size (512), N is the number of frames in any sequence and M is the total number of sequences (we expect $M \ll N$). Both methods use a ResNet encoder, adding a fixed overhead. The train time is shown for one epoch.

	Train Time (ResNet: 63ms $\times N$)	Inference Time Complexity (ResNet +63ms)	Memory (ResNet +44.7MB)
Baseline Action Prediction	-	$D\log(N)$	2.0KB $\times M \times N$
Baseline Path Selection	-	$D\log(N) \times M$	2.0KB $\times M \times N$
Proposed Action Prediction	+10ms (2ms total GPU)$\times N$	$D \times 3$	6.2KB $\times M$
Proposed Path Selection	+10ms (2ms total GPU)$\times N$	$D \times M$	2.1KB $\times M$

database of images of places [1].

2.4. Map-less Navigation

Recently, it has been suggested that an end-to-end fully learned functional mapping from sensor input to action may be able to solve some kinds of navigation problems [20]. These methods do not construct a map explicitly, but learn a representation useful for the given task. These methods can be further differentiated as "planning based" and "reactive control" approaches [4, 12]. In a planning based method, prior data and inference over expected scene regularities is used to select actions to achieve some goal state [31, 37]. Reactive control methods such as [11] use learned policies that map observations to reasonable actions within some general class of environment. In simplified environments these strategies perform similarly, but in sufficiently complex scenarios planning based methods may be necessary to achieve improved performance.

3. Approach

The problem we consider in this work is following a visual path captured using a monocular RGB camera. We assume there is no other source of odometry and the vertical FOV is restricted to less than 50 degrees. These restrictions are imposed to make our method suitable in applications utilizing commodity general purpose hardware (e.g. smartphone cameras).

Table 1 enumerates the computational benefits of our approach. Encoding time is not excessive and the amount of data needed to perform inference is smaller than performing a lookup on a saved sequence of frames.

Within this problem framework, our proposed system consists of three parts.

1. **Unsupervised VO Labeling**: Estimate the local motion from a monocular image sequence.
2. **Path Encoding**: Learn an encoder that embeds input images so that a classifier can select the correct high-level action corresponding to each part of the sequence.

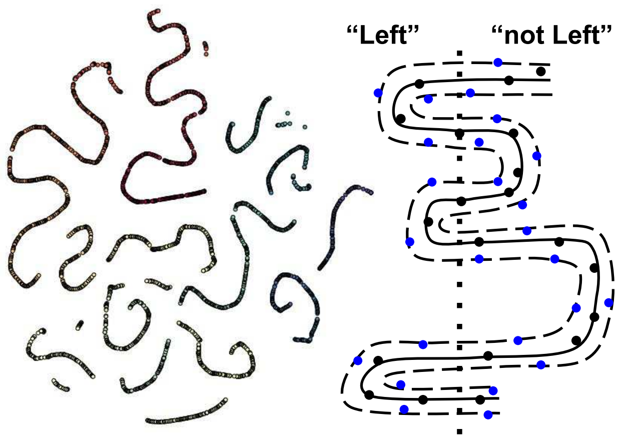


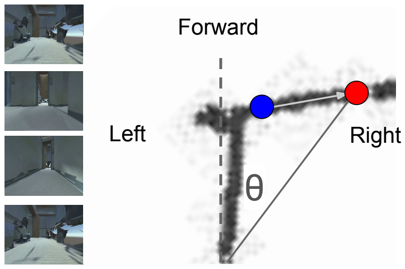
Figure 2: Left: t-SNE embedding of experiment path "a1-00" using ResNet-18 features trained on ImageNet. Notice that image order is preserved in the encoding (color maps to frame number). Right: Visualization of linear classifier learning sections of the sequence corresponding to the "left" label. Black and blue dots are representative of training and test data respectively.

3. **Path Selection**: Determine where the input corresponds with the path or select from a set of memorized paths.

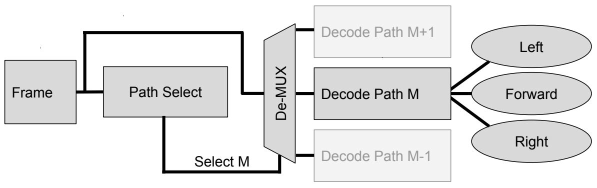
3.1. Unsupervised VO Labeling

We estimate the per-frame visual odometry in order to determine the prediction labels for training (forward, left, right). Our method is entirely unsupervised the only input required is the image sequence and the camera field of view. We adapted an open source visual odometry implementation [28] for our experiments. Our unsupervised labeling method could utilize another odometry method as long as it estimates proper metric scale.

Visual Odometry with Estimated Scale This approach uses tracked FAST features [26, 27] and Nister's



(a)



(b)

Figure 3: (a) Visual odometry is performed on a sequence of frames. Each frame is labeled by the future trajectory determined by VO. The labels are used to train an action prediction network for each sequence and a path selection network. (b) At inference time the correct path is selected and directed to the appropriate action selection network.

method [24] to estimate the essential matrix. We then factor the essential matrix and select the feasible solution to recover pose changes between the current frame and a keyframe to a relative scale. The baseline of the pose is normalized and each feature is then triangulated so that the depth relative to the baseline is known.

We solve for scale by estimating the distance of each feature in the input images with an pre-trained FCRN-depth [18] network trained on the NYU dataset. As in [33], the output of this network is scaled to match the intrinsic camera parameters since depth estimates predicted by the network are incorrect for our input FOV. The accumulated FCRN-depth estimates are averaged over a small window for each feature and a scaling factor is selected that incorporates the local depth estimate. The median scaling factor over all features is used to scale the pose estimate appropriately.

In order to reduce spurious solutions from small baselines, we select a keyframe to localize subsequent images until the estimated baseline exceeds a threshold. We also limit the gradient of the scaling factor over time and apply a smoothing function to reduce the impact of spurious odometry estimates.

This pipeline runs real-time using a GPU to estimate depths from each image, while the rest of the pipeline runs faster than real-time on a CPU.

Unsupervised Labeling using Visual Odometry With our visual odometry in place, we label each frame with its local motion (Figure 3). For each frame we label future motion after following the path a distance d. After the pose on the path ahead of the current frame is found, a reference point is placed 1 meter in front of the pose in order to incorporate information about the pose orientation. We assume that the agent preferentially moves in the direction of their orientation, and that local future motion does not occur along the y axis (up/down).

Look ahead ensures the network learns to predict motion for a specific sequence and does not associate image features such as motion blur with particular directions. In our experiments, we set d to 0.5 meter, which provides a good trade-off of future prediction and relevance.

Once the future location along the path is found for a given frame, the label is chosen with the following rule:

$$l = \begin{cases} \text{left} & \theta > \theta_0 \\ \text{forward} & |\theta| < \theta_0 \\ \text{right} & \theta < -\theta_0 \end{cases} \tag{1}$$

Where $\theta = \arctan(z, x)$ is the angle between the current orientation and a vector emanating from the current frame position to the point found through look ahead. In all experiments, θ_0 was chosen to be 15 degrees.

3.2. Path Encoding and Action Selection

The path encoding step attempts to learn a classifier over the sequence of observations, mapping each observed input to the high-level action to take. We briefly sketch how a classifier can be found that is able to label each subset of a sequence. Consider a sequence of frames: we would expect that each frame in the sequence is a point on a 1D manifold moving through pixel space. We then consider a differentiable transformation Φ that projects this set of points into feature space. We note that we're interested in transformations that preserve the topology of the 1D pixel space manifold.

The structure of the 1D manifold in feature space enables the discrimination of its elements with a hyperplane. In Figure 2, notice that a 1D line can easily be warped in 2D so that a 1D linear decision boundary can be defined to label any subset of the line. We thus pose path encoding as the problem of finding Φ that warps the manifold containing our input sequence so that it can be cut into two desired subsets with a hyperplane. Our general approach is to represent Φ with a deep neural network.

$$\text{argmax} g_c(\Phi(x_n)) \tag{2}$$

Where g_c is a decoding function that receives an embedding from Φ and predicts whether the desired class is present or not. In our experiments we use fully connected layers followed by rectified linear units to represent g_c (the output is not followed by a ReLU). We choose Φ such that the tangent space of the manifold containing the input sequence warps along with the data, so that the classifier generalizes to known invariants to the path sequence (blue dots Figure 2), such as lighting or slight changes in appearance along the path. In the sequence labeling task, our model needs enough capacity to produce the necessary folds so that the linear classifier is effective.

The consolidated paths should preserve the desired high-level actions even when the observations are perturbed, either due to lighting, new objects in the scene, or changes in viewpoint. CNNs in general have been shown to be capable of learning representations invariant to these types of perturbations. We use the ResNet-18 trained on ImageNet as our sequence encoding function, utilizing the feature layer before the classification layer. Instead of predicting the image class, we add a three output layer corresponding to the three possible high-level actions: “forward”, “left”, and “right”.

Method 1: Reactive Control In the case where the input sequence never overlaps with itself in feature space, every element can be uniquely mapped to an action. This problem reduces to a reactive control problem: each action can be determined from the current input, and no global map understanding is required. A reactive control network recieves embeddings from ResNet-18 and attempts to directly predict the action to take. We train the network by randomly selecting images from the input sequence and providing the target predictions from the local motion graph.

This approach is fundamentally susceptible to perceptual aliasing, where the projected data $\Phi(x)$ may produce a 1D path representation that overlaps with itself from real metric loop closures or the appearance of loop closures that do not actually exist. In these cases, representations from adjacent frames could be combined over time using a long short-term memory (LSTM), producing an encoding that incorporates more information about the sequence leading up to the current frame. However, LSTMs do not entirely address the problem and may still suffer from perceptual aliasing. Instead, we introduce an alternative that predicts location along the path directly.

Method 2: Path Sequence Recall An alternative architecture that more closely resembles traditional VT&R methods is to use incoming frames to estimate progress along the path itself. Action selection is then accomplished by referencing a lookup table of prior actions for each section along the path.

The path sequence recall network observes an incoming frame and then predicts the likelihood that the frame correspondes to discrete sections along the path. In our experiments we divide each path into 16 sections and then interpolate the progress along the path by combining the activations of the 16 output position prediction neurons. We use a sliding mask to restrict the combination of positions to adjacent sections with a known start. This masking step reduces confusion between similar sections of the path and enables the recovery of the original sequence even for paths that overlap.

It may be advantageous to combine Reactive Control and Path Sequence Recall into a heirarchical model. In such a construction, a path selection network predicts what “sub-path” the agent is on, such as the discrete sections used for Path Sequence Recall, and then the appropriate reactive control network is selected (Figure 3). This is done by introducing a “path selection” supervisory network.

3.3. Path Selection

Our path following formulation becomes significantly more useful if it can select from a set of previously observed paths. This path selection enables high-level path control and more flexible navigation for certain applications. We compose memorized paths that each contain some minimal overlap with each other, and then select from them to determine what action to take.

For path selection, we train a new network that takes the current frame as input and predicts the likelihood of the observation corresponding to each path. As such, each output corresponds to a particular path. During inference, the most likely path would be selected, corresponding to action selection network that would then use the visual input to determine the correct action.

4. Experiments

4.1. Evaluation Environment

We use the Stanford 2D-3D-Semantics Dataset (2D-3D-S) to evaluate our approach [2, 3].

The textured 3D mesh from the 2D-3D-S dataset was used to produce a Unity evaluation environment consisting of a camera and controls for forward, rotate right, and rotate left. This enables a user to navigate virtually in real time through the environment mesh. Ground truth pose information of the camera was recorded for each captured frame. A more detailed overview of this environment can be found in the Supplementary Material.

All experiments use a Nvidia GeForce GTX 1070 with 8GB of RAM on a machine with an 8-core Intel i7–6700 with 32GB of RAM. All experimental results are reported for validation runs (not used for training). Action and Path Selection networks are trained with Stochastic Gradient Descent with a batch size of 4 and a learning rate of 0.001 and momentum of 0.9. Unless otherwise noted, we train each encoding/decoding network for 7 epochs. For a typical path length of 3k frames, training takes 20s per epoch

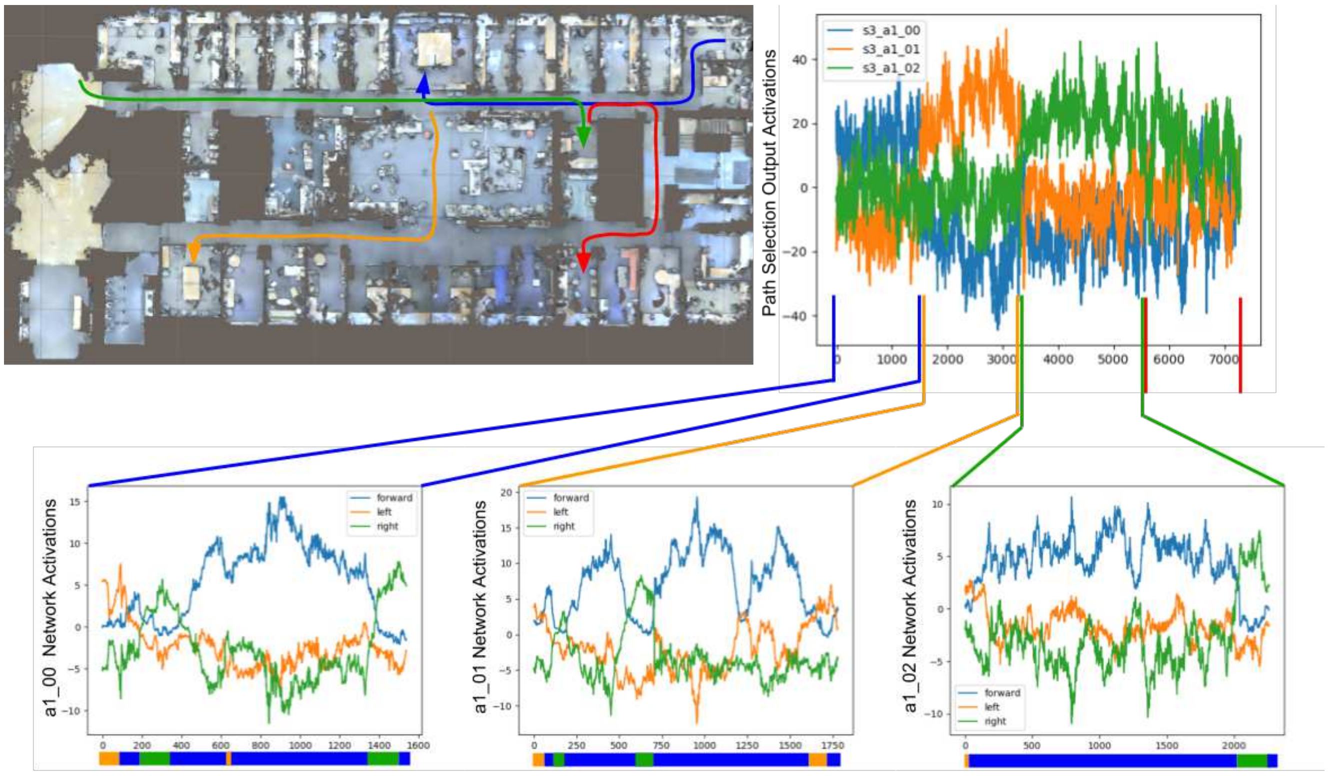


Figure 4: Top view of "Area 1" in the 2D-3D-S dataset [2, 3] with paths drawn for the first 4 visual paths. The top right plot shows the corresponding activations of the path selection decoder when replaying the three paths in order (color coded). The bottom plots show the individual action decoder for each path along with the ground truth label along the bottom of each.

and running the inference over all validation frames takes 12s (batch size 1 for average latency of 4 ms or 250 fps). Performing a forward pass using the CPU takes 63ms.

4.2. Baseline

Our approach is compared to the baseline of matching the input frame to a database of saved frames. This is an essential procedure in VT&R for the localization step during the "repeat" phase. This operation also contributes to a large share of the operational resource requirements of VT&R systems.

Following the recent trend replacing feature based visual bag of words with CNN feature extractors, we pre-process the known path sequence by extracting CNN features using the same pre-trained ResNet-18 used by the proposed decoders. We find the nearest neighbor in the training set and associated label to produce the high level action class. Our experiments use the Python Scikit Learning library implementation of a ball tree nearest neighbor algorithm.

The proposed system functions much differently than this baseline, but solves the same task of identifying what action to take based on prior experience. See Table 1 to see the computational requirements of the baseline compared to the proposed method. Our implementation requires less memory and performs faster inference than the optimized baseline implementation while performing comparably at recalling the proper action compared to the baseline.

We show results for each method using ground truth pose graphs available from the evaluation environment in addition to labels produced by the visual odometry system in Table 2.

4.3. Single Path Task

The single path task is to determine the high-level action from a single reference image sequence while traveling along the same path for a second time. For these experiments, we evaluate the performance of our approach as the accuracy of future action prediction over each frame in the sequence compared to ground truth. Area 1,3, and 6 were used from the 2D-3D-S dataset, following the recommended validation split.

Reactive Control The path decoding network successfully memorizes the set of future actions to take (Table 2). This network learns to associate certain key frame landmarks with each action, effectively discovering the regions

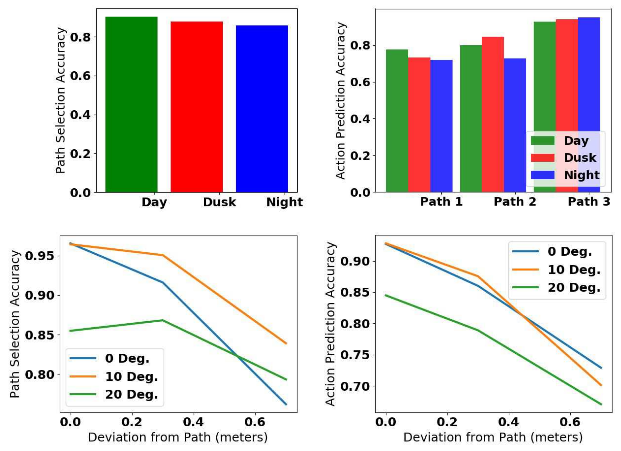


Figure 5: Accuracy of path selection (left) and action prediction (right) under lighting pertubations (top) and translational and rotational offset from the original paths (bottom).

in feature space that are relevant to future prediction. We see in Figure 4 that the predicted actions are stable throughout the route, and exhibit interesting inhibition behavior as the input transitions from a region in feature space associated with one action to another. This demonstrates that the single layer linear classifiers are able to effectively separate each region in the sequence.

One reasonable concern is that the paths have a strong forward bias. The classifiers may appear to perform accurately while only predicting the "forward" label. This does not seem to be the case for the experiments. As seen in Figure 4, the disagreement between the labels and the network predictions is typically isolated to a few frames before and after direction changes.

The action prediction network performed comparably to the baseline. It did not perform as well for Area 6. Fine-tuning the ResNet seems to only provide marginal improvement. Fine-tuning sometimes decreases performance, suggesting there may be some over-fitting introduced by lower layers in the ResNet encoder in these cases. Regardless, the error differences are not dramatic between the baseline and proposed action prediction network, despite the action prediction network making inferences without the frame database and using two orders of magnitude less memory.

Path Sequence Recall We implemented a path sequence recall network to investigate ways of recovering the sequence when reactive control alone is not sufficient. The network predicted the location along the path and a sliding window was applied to ensure only adjacent regions along the path could be activated at one time. Figure 6 demonstrates the sequence recall for a route in Area 1 (the green

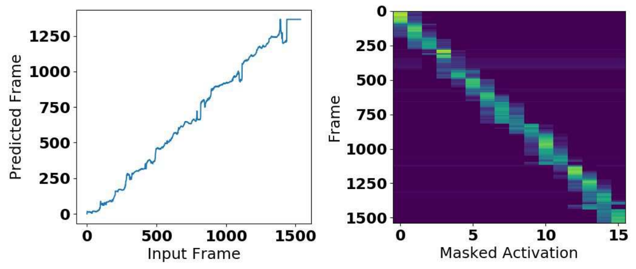


Figure 6: Path Sequence Recall for Area 1 Route 1 (s3-a1-00). The left plot shows the ground truth input frame position plotted against the predicted frame. This prediction is found by interpolating from the masked neuron activation shown in the activation plot to the right.

Table 2: Comparison between accuracy on 3 areas in the 2D-3D-S dataset for the path action prediction and path selection. The nearest neighbor baseline is compared to the Path action Decoder (PD) while fine-tuning the ResNet (+FT) and when training using perfect ground truth (+GT). All results are shown for validation data based on ground truth.

Method	**2D-3D-S Area 1**	**2D-3D-S Area 3**	**2D-3D-S Area 6**
	Path Action Prediction Accuracy		
Baseline	0.7333	0.6472	0.7318
PD	0.7567	0.6991	0.7099
PD+FT	0.7678	0.6830	0.7472
Baseline+GT	0.8644	0.8332	0.8700
PD+GT	0.8553	0.8312	0.8550
PD+GT+FT	0.8576	0.8060	0.8550
	Path Selection Accuracy		
Baseline	0.9892	0.9913	0.9920
PS	0.9406	0.9147	0.9019
PS+FT	0.9244	0.9475	0.8864

path in Figure 4). It is possible to predict the specific frame number because neurons exhibited activation values proportional to the distance from the set of frames they were trained.

Reactive Control methods did not seem to suffer from significant problems with perceptual aliasing in our tasks, so our analysis focuses on path selection as a more general sub-path selection architecture.

Prediction Error for out of sample Perturbations To ensure robust performance, it is important to examine pose and lighting change perturbations on the path prediction accuracy. In Figure 5, the trained networks show some learned

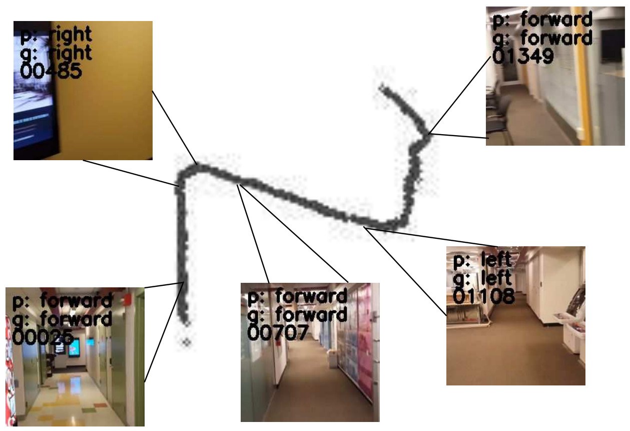


Figure 7: Example visual path captured using a smartphone (Samsung Galaxy Edge S6+) with 46 degree vertical FOV. Two image sequences were captured, one for training and the validation shown above using the same procedure as described in the experiments.

invariance to lighting and pose, except for a sharp performance drop off for orientation changes greater than 20 degrees and path location offsets greater than 0.2 meters. We did not test offset errors greater than 0.7 meters because that was the largest offset that would fit in the hallways in the validation environment. More aggressive data augmentation may be necessary to achieve improved invariance to large angle variation. In practice we imagine that specifying canonical orientations may reduce such problems in structured environments (e.g. canonical orientation in a hallway).

4.4. Path Selection

The path selection experiments evaluate the ability of the proposed method to select between a set of path hypotheses. In a practical system, this would be the first step before routing the input to the correct path decoding network. Being able to select the correct path with a high accuracy enables more complex routing to be implemented.

Path selection is accurate, both the baseline and the Path Selection decoders were able to achieve over 90% in many of the cases. Interestingly, as demonstrated in Figure 4, ambiguities in the network activations indicate low confidence of being on any path at all. This would be expected from such path selection methods, and suggests an interesting direction for future work where this uncertainty could be exploited when moving through partially known environments.

4.5. Real-world Path Following

Finally, we demonstrate path following performance on a real world dataset. Video was captured moving through an office environment similar to that of the simulated dataset. The ground truth was not known for these image sequences, Figure 7 shows the estimated visual odometry along with frames labeled by the decoder. More work is needed to evaluate performance compared to ground truth, the system was able to predict the visual odometry labels with an accuracy of 93.4%. This accuracy demonstrates that the action prediction network is able to recall the labels produced by VO.

5. Conclusion

We introduce a method of solving VT&R tasks for paths that can be described by an observation and corresponding desired direction: forward, left or right. We solve these tasks using small (6.2KB), path specific classifiers trained on embeddings produced by ResNet-18. We introduced another classifier that estimates which of known paths the current observations correspond. All results are shown for validation data. The performance is comparable to a nearest neighbor baseline despite not having access to the original image sequence or associated features and pose graph.

The experiments suggest that neural network models are able to learn path specific reaction control policies to follow an example path for VT&R tasks. Furthermore, similar neural network architectures are able to predict the currently observed sub-path. By combining these predictions with a sequential masking operation, the location along the sequence can be recovered without referring to a frame feature database.

Deep networks have a number of advantages compared to traditional methods: they have fixed memory and time complexity during inference, they benefit from improvements in the deep learning ecosystem (such as better embedding architectures), are easily adapted to other modalities and reduce overall system complexity. The use of visual odometry as an unsupervised labeling scheme suggests new ways to combine traditional and deep learning based methods.

These networks demonstrate some invariance to input variations, but new methods must be addressed to increase the networks ability to tolerate large changes. Finally, our proposed method does not implement visual servoing typically found in VT&R architectures, suggesting future work that may accomplish visual servoing from deep neural network features. Despite these limitations, we hope Deep VT&R may provide new applications that leverage its unique ability to solve path following problems with low computational complexity and memory requirements.

Acknowledgements

We would like to thank the reviewers for their helpful comments. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. (1122374).

References

[1] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[2] I. Armeni, A. Sax, A. R. Zamir, and S. Savarese. Joint 2D-3D-Semantic Data for Indoor Scene Understanding. *ArXiv e-prints*, Feb. 2017.

[3] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2016.

[4] S. Brahmbhatt and J. Hays. Deepnav: Learning to navigate large cities. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[5] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, Dec 2016.

[6] A. Dine, A. Elouardi, B. Vincke, and S. Bouaziz. Graph-based simultaneous localization and mapping: Computational complexity reduction on a multicore heterogeneous architecture. *IEEE Robotics Automation Magazine*, 23(4):160–173, Dec 2016.

[7] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision (ECCV)*, September 2014.

[8] P. Furgale and T. D. Barfoot. Visual teach and repeat for longrange rover autonomy. *Journal of Field Robotics*, 27(5):534–560.

[9] X. Gao and T. Zhang. Loop closure detection for visual slam systems using deep neural networks. In *2015 34th Chinese Control Conference (CCC)*, pages 5851–5856, July 2015.

[10] E. Garcia-Fidalgo and A. Ortiz. Vision-based topological mapping and localization methods: A survey. *Robotics and Autonomous Systems*, 64(Supplement C):1 – 20, 2015.

[11] A. Giusti, J. Guzzi, D. C. Cirean, F. L. He, J. P. Rodrguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. D. Caro, D. Scaramuzza, and L. M. Gambardella. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 1(2):661–667, July 2016.

[12] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive mapping and planning for visual navigation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[13] X. He, R. S. Zemel, and V. Mnih. Topological map learning from outdoor image sequences. *Journal of Field Robotics*, 23(11-12):1091–1104, 2006.

[14] Y. Hou, H. Zhang, and S. Zhou. Convolutional neural network-based image representation for visual loop closure detection. In *2015 IEEE International Conference on Information and Automation*, pages 2238–2245, Aug 2015.

[15] A. Kendall, M. Grimes, and R. Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pages 2938–2946. IEEE, 2015.

[16] K. Konolige, J. Bowman, J. Chen, P. Mihelich, M. Calonder, V. Lepetit, and P. Fua. View-based maps. *Int. J. Rob. Res.*, 29(8):941–957, July 2010.

[17] M. Labb and F. Michaud. Online global loop closure detection for large-scale multi-session graph-based slam. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2661–2666, Sept 2014.

[18] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 239–248. IEEE, 2016.

[19] S. Lowry, N. Snderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford. Visual place recognition: A survey. *IEEE Transactions on Robotics*, 32(1):1–19, Feb 2016.

[20] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, D. Kumaran, and R. Hadsell. Learning to navigate in complex environments. In *ICLR*, volume abs/1611.03673, 2017.

[21] R. Mur-Artal, J. M. M. Montiel, and J. D. Tards. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, Oct 2015.

[22] R. Mur-Artal and J. D. Tards. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, Oct 2017.

[23] T. Nguyen, G. K. I. Mann, R. G. Gosine, and A. Vardy. Appearance-based visual-teach-and-repeat navigation technique for micro aerial vehicle. *Journal of Intelligent & Robotic Systems*, 84(1):217–240, Dec 2016.

[24] D. Nister. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–770, June 2004.

[25] A. Pfrunder, A. P. Schoellig, and T. D. Barfoot. A proof-of-concept demonstration of visual teach and repeat on a quadrocopter using an altitude sensor and a monocular camera. In *2014 Canadian Conference on Computer and Robot Vision*, pages 238–245, May 2014.

[26] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *IEEE International Conference on Computer Vision*, volume 2, pages 1508–1511, October 2005.

[27] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pages 430–443, May 2006.

[28] A. Singh. Vo-mono: github.com/avisingh599/mono-vo, 2015.

[29] S. Song, M. Chandraker, and C. C. Guest. High accuracy monocular sfm and scale correction for autonomous driving. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(4):730–743, April 2016.

[30] B. E. Stenning, C. McManus, and T. D. Barfoot. Planning using a network of reusable paths: A physical embodiment of a rapidly exploring random tree. *Journal of Field Robotics*, 30(6):916–950, 2013.

[31] L. Tai, G. Paolo, and M. Liu. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. *CoRR*, abs/1703.00420, 2017.

[32] C. Tang, O. Wang, and P. Tan. Globalslam: Initialization-robust monocular visual SLAM. *CoRR*, abs/1708.04814, 2017.

[33] K. Tateno, F. Tombari, I. Laina, and N. Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[34] S. Thrun and J. J. Leonard. *Simultaneous Localization and Mapping*, pages 871–889. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[35] S. K. v. Es and T. D. Barfoot. Being in two places at once: Smooth visual path following on globally inconsistent pose graphs. In *2015 12th Conference on Computer and Robot Vision*, pages 54–61, June 2015.

[36] R. Wang, M. Schwörer, and D. Cremers. Stereo dso: Large-scale direct sparse visual odometry with stereo cameras. In *International Conference on Computer Vision (ICCV)*, Venice, Italy, October 2017.

[37] G. Wei, D. Hus, W. S. Lee, S. Shen, and K. Subramanian. Intention-net: Integrating planning and deep learning for goal-directed autonomous navigation. *arXiv preprint arXiv:1710.05627*, 2017.