Visually Localizing in Mcity from Simulated Data

Mollie Bianchi and Steven Waslander

***Abstract*— Localization is a key aspect of any self-driving vehicle but relying solely on GPS can result in offsets from a given semantic map. Visual localization is a cost effective solution that works by registering live images with images stored in a map. This work presents a method for finding the 3DoF relative pose change between real images captured at Mcity and simulated geo-referenced images from an Mcity simulation, and is intended for use in a visual localization pipeline. It also investigates the use of synthetic data during training to increase performance.**

I. INTRODUCTION

An important aspect of self-driving vehicles is their ability to localize within a map. Most often this is achieved using global positioning system (GPS); however, GPS has several limitations. It is noisy, drifts over time, can dropout in tunnels or urban canyons, and can have an offset with respect to the map being used. This was an issue experienced first hand by the University of Toronto's Autodrive team, aUToronto, during the second year of the Autodrive competition held at Mcity. There was an approximately 1m difference between the vehicle's actual position in the semantic map and the position being reported from GPS. To correct for this offset, aUToronto used LIDAR localization to determine their position within the semantic map. LIDAR is a very expensive sensor, so instead this work looks at using cameras for localization.

The RightHook simulation environment [1] is a high fidelity reconstruction based off of the same LIDAR map of Mcity the team used for localization. This simulation allows a camera with adjustable parameters to be placed at any position on the vehicle and will render images as seen from that camera pose. Figure 1 shows an example of two images taken from approximately the same location in both the simulation and the real world. The purpose of this work is to investigate whether images rendered from the simulation can be used for localization.

Fig. 1: On the left is a image rendered from the RightHook Mcity environemnt and on the right is a picture taken at the physical Mcity location.

Visual Teach and Repeat (VT&R)[2] is a method by which a robot is able to repeat a path by localizing current images to images captured along the path during a teach pass. A similar strategy could be employed here. Images could be rendered beforehand along a desired path in simulation. Then as the path is traversed physically, the simulated map images would be used to localize against. In order to do so, the relative pose change between the simulated map image and the real image would need to be computed. In VT&R this is achieved using sparse SURF features[3], but features are unlikely to work in this case due to the two different modalities of the images.

In this work, we present a neural network based method for estimating the 3 Degree of Freedom (DoF) relative pose change, in terms of x, y and θ, between a simulated map image and a real image. This relative pose can then be used to compute the global 3DoF pose where the real image was captured. This assumes that an initial guess of the vehicle's position would be available, either from a dead reckoning method or noisy GPS, which would allow a nearby map image to be selected, and that this localization method would be used within a filter in the correction step.

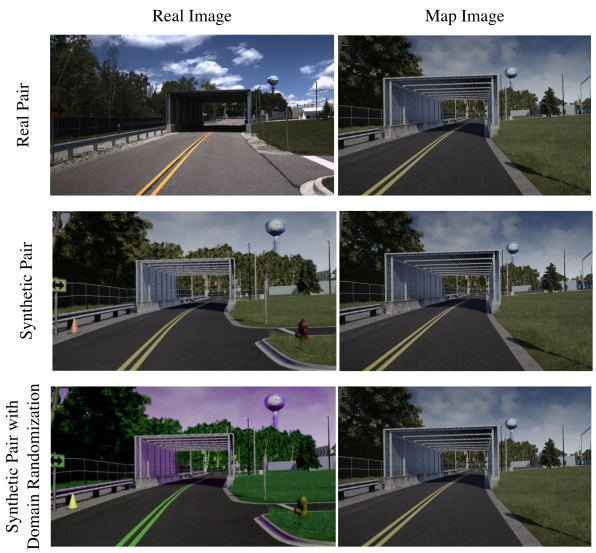


Fig. 2: The three types of pairs used during training: a real pair, a synthetic pair, and a synthetic pair with domain randomization.

Collecting real data is an expensive and time-consuming process, while a theoretically unlimited amount of data could be generated in simulation without ever having to go to Mcity. The other aspect of this work looks at how adding synthetic data during training could help improve performance, specifically by adding synthetic training pairs (see Figure 2) where both images come from the simulation, and experimenting with the combination of real and synthetic

data required.

II. Related Work

A method of localizing monocular camera images to images rendered from a 3D model of the area was presented by Pascoe et al. [4], [5]. They generated a fully textured 3 dimensional mesh of their interest area beforehand from LIDAR scans and camera data. They were then able to generate synthetic images from this model. The live image and the synthetic image were registered by minimizing the Normalized Information Distance (NID) between them.

Neural networks have previously been applied to the relative pose change problem. Melekhov et al. [6] use a convolutional neural network based on AlexNet [7] to estimate the 6DoF relative pose change between two images. Their modified network has a Siamese structure containing a representation component and a regression component. The representation component is comprised of two identical branches with shared weights. An RGB image is passed to each branch, which extracts useful features. The two feature sets are concatenated before passing through the regression part of the network. The regression part consists of two parallel but separate fully connected layers. One of which outputs a 4 dimensional vector to represent the rotation expressed as a quaternion and the second outputs a three dimensional vector for the translation components.

Visual odometry (VO) is a specific application of relative pose estimation and there have been many works applying CNNs to this topic [8]-[11]. [8] uses a supervised method similar to [6]. [9] incorporates an LSTM to exploit the sequential aspect of the VO problem. [10] uses a CNN to learn corrections to drifted, traditional pose estimates. Zhou et al. [11] take an unsupervised approach by training on unlabelled video sequences from the KITTI dataset. This was accomplished by training one network to learn a depth map of each target image while another network learned relative pose between temporally adjacent image frames. Using the depth map and the relative pose output from the network, the adjacent images could be warped onto the target image. A photometric loss is then backpropagated through the network.

The key motivator behind using a neural network based approach for this problem instead of feature based methods, is the network's ability to perform robustly across vastly different experience changes. One example of this is in [12]. Data collected from multi-experience VT&R across three seasons and 24 hours of lighting change was used to train a network based on AlexNet[7] to estimate the 3 DoF relative pose between two images. This relative pose change was then used successfully for VO, localization, and a combination of the two.

An alternate approach is to use a network to transform the images to a canonical form before applying a classical feature method. This was explored in [13]. An encoder decoder structure was used to transform images under a variety of lighting conditions to a canonical lighting condition. The transformed images were then able to yield improved results when performing VO and localization.

Works in classification can face a similar problem of needing to identify objects across image modalities. [14] looks at the case where they want to be able to classify scenes represented in real images, clip art, and sketches. They achieved this by training one network for each modality but forcing the higher level layers to share weights. The idea is to let early layers specialize to modality specific features, while the later layers capture higher level concepts independent of the modality.

There is also much work being done on training networks with synthetic data and generalizing to real data. One technique for accomplishing this is called domain randomization. This takes hundreds of thousands of synthetic images and renders them with random colours, textures, and lighting conditions that may not seem realistic. The idea is that given enough variability in the training data, the network will learn to treat the test data as just another variation. This technique has been applied successfully to problems such as object detection [15], robotic grasping [16], and manipulator pose estimation [17].

III. Methodology

A. System Overview

Our method uses a neural network which takes as input a real image and a simulated map image. The network regresses the 3DoF relative pose from the map image frame to the real image frame. The network has been trained multiple times using a variety of combinations of real pairs, synthetic pairs, and synthetic pairs with domain randomization as summarized in Table I.

B. Data Collection

This project requires both real and simulated images. The real images were collected by aUToronto at Mcity during the competition. The ground truth position data for the images comes from the RTK GPS data corrected with the offset found from LIDAR localization. The transformation from the GPS frame on the car to the camera frame on the car was then applied to get the position of the camera when the image was taken. Images were collected along the five paths shown in Figure 3 using a 5 MP Blackfly BFS-U3-51S5C-C monocular camera. This data collection was undertaken over three days in both sunny and overcast conditions.

The simulated camera was configured to best replicate the physical camera with an image size of (2448×2048), a horizontal field of view of 80°and placement 1.15 m in front of the car's rear axle at a height of 1.715 m. The same five paths were then driven in simulation while rendering and storing the images. The ground truth position of the camera when the image was rendered could be obtained directly from the simulation.

Labelled pairs were generated for every real image by selecting the nearest simulation image. Pairs could have been generated between the real image and all simulation images within a given radius to increase the amount of training data and increase the distance at which the network could learn to localize. However, for this project we have assumed a

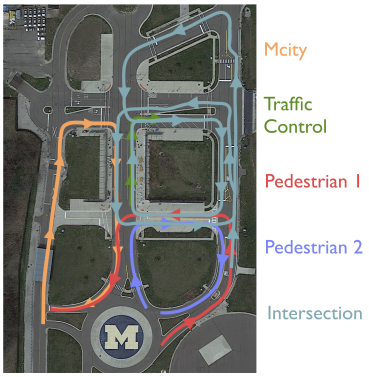


Fig. 3: The five paths along which data was collected.

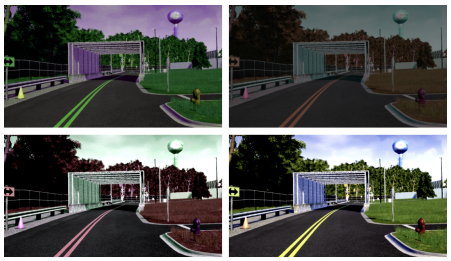


Fig. 4: Sample simulated images after color jitter function has been applied.

good initial guess of the vehicle's position, from odometry or GPS, and the purpose of the network would only be to refine it. The ground truth position in UTM coordinates and the heading of both the real and simulated mages are known. We can then construct a transformation from the real image to the global frame, $\mathbf{T}_{iv}$, and from the simulated map image frame to the global frame, $\mathbf{T}_{im}$. The relative transformation from the simulated map image to the real image frame is then $\mathbf{T}_{vm} = \mathbf{T}_{iv}^{-1}\mathbf{T}_{im}$. See Figure 6 for reference. From $\mathbf{T}_{vm}$ we can extract the values of x, y and θ for the relative pose change. These values were then used as the labels. 9438 real and sim pairs were generated. A random 750 were removed for validation, and an additional 750 for testing. The average value for x, y, and θ are 0.12m, 0.13m, and 0.025 rad. Depending on the path, values for x and y could be as large a 1m. The reason the θ differences are so low is because both the real and simulated collection was done while driving along the path and it was unlikely for either vehicle to have a significant heading error during collection.

The synthetic training data was created by going through each simulated image and creating a pair with any other simulated image whose position was within a 0.75m radius. 25,437 pairs of labeled synthetic data were generated to be used for training.

The effect of domain randomization was also explored. This was done by using PyTorch's random color jitter function with the following parameters: brightness=0.5, contrast=0.8, saturation=0.8, and hue=0.3. Some examples can be seen in Figure 4. Only the simulated real image in each synthetic pair was randomized, since in the both the synthetic pairs and the real pairs the map image comes from simulation, see Figure 2.

C. Network Architecture

The network architecture follows [12], [6] and is based on AlexNet [7]. The network layers that remain unchanged from AlexNet were initialized with the pretrained weights available in PyTorch. Figure 5 shows a detailed breakdown of the whole network. The network takes a real image and a simulated map image each downsized and then cropped to (360 x 640) as inputs to Siamese feature extraction branches. A Spatial Pyramid Pooling Layer [18] is used at the end of the feature extraction branches to account for the increase in input image size as compared to AlexNet. The output of these feature extraction branches are then concatenated and passed to the regression part of the network. This part is primarily composed of fully connected layers. The final layer outputs a three dimensional vector corresponding to the x, y, and θ parameters of the relative pose change. A mean square error loss is computed as $\mathcal{L} = (x-\hat{x})^2+(y-\hat{y})^2+(\theta-\hat{\theta})^2$ where $\hat{x}, \hat{y}$ and $\hat{\theta}$ are the labels associated with the input pair.

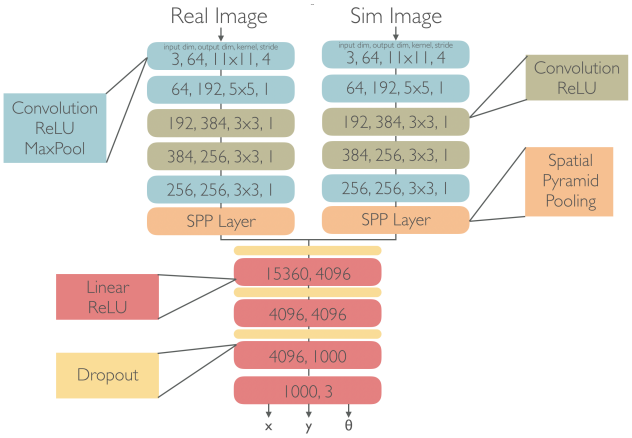


Fig. 5: Detailed network architecture

D. Localizing from Relative Pose

We can let $\underrightarrow{\mathcal{F}}_i$ be the global frame, $\underrightarrow{\mathcal{F}}_m$ be frame at the simulated camera pose when the map image was rendered, and let $\underrightarrow{\mathcal{F}}_v$ the frame at the physical camera pose when the real image was captured. The relative pose change, $\mathbf{T}_{vm}$, is estimated from the network. Since the global pose of the simulated map image, $\mathbf{T}_{im}$, is known, the global pose of the live image can then be obtained by $\mathbf{T}_{iv} = \mathbf{T}_{im}\mathbf{T}_{vm}^{-1}$.

E. Warping Process

For visualizing of the results, the real image was warped onto the simulated map image. This was achieved by using an inverse warping method. The simulation environment is also able to provide a ground truth depth image along with every RGB image. Each pixel location $\mathbf{u} = [u\, v]^T$ with depth d in the simulated map image was transformed to 3D space according to $\mathbf{x} = d\mathbf{K}^{-1}\bar{\mathbf{u}}$ where $\mathbf{K}$ is the camera intrinsic matrix and $\bar{\mathbf{u}} = [u\, v\, 1]^T$. Each of the 3D points was

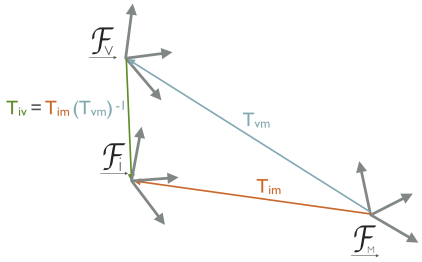


Fig. 6: Coordinate frames under consideration.

transformed to the real camera frame using the estimated relative transform, $\mathbf{x}' = \mathbf{T}_{vm}\mathbf{x}$. They were then projected onto the image plane and transformed to pixel coordinates in the real image according to $\bar{\mathbf{u}}' = \mathbf{K}\mathbf{x}'/z$. The warped image was then constructed by using the value at $\mathbf{u}'$ in the real image at pixel location $\mathbf{u}$.

IV. Results

The network was trained with different combinations of real and synthetic data as outlined in Table: I. The same validation and test sets comprised of only real data were used in all cases. The average root mean square error (RMSE) for each training condition for each of the degrees of freedom is shown in Figures 8a, 8c, and 8e. The cumulative RMSE distribution plots are shown in Figures 8b, 8d, and 8f.

The lowest RMSE for x was 0.0224m when training using only real data. Similar performance was achieved when synthetic data was added to the training set, but training only on synthetic data resulted in significantly worse performance. This can also be seen in the cumulative distribution plot, Figure 8b, where the brown and orange lines representing training only on synthetic data have errors out to 4 or 5 m.

Similar results can be seen in Figure 8c for y. The lowest RMSE was 0.0611m when training with only real data. In the cumulative error plot, Figure 8d, the blue line which shows training only using real data is the furthest to the left. The brown and orange lines, for only training on synthetic data, are the furthest to the right. The other training conditions with various amounts of real and synthetic data all lie close to each other within these.

The performance for θ is much closer together between the training conditions. This is likely because there is not a large variability in the θ values in the training examples, see Section III-B. The lowest RMSE for θ was 0.0137 rad when training with twice as much synthetic data as real data and domain randomization.

Figure 7 shows three examples of performance on the test set from the network trained only on real data. The plots on the left hand side show the position of the simulated camera as a red dot, the ground truth real camera position as a blue dot, and the estimated real camera position as a green dot. On the right side, the two smaller images show the simulated map image and the real image at their respective poses. The large image shows the real image warped onto the simulated image and overlaid with the simulated image.

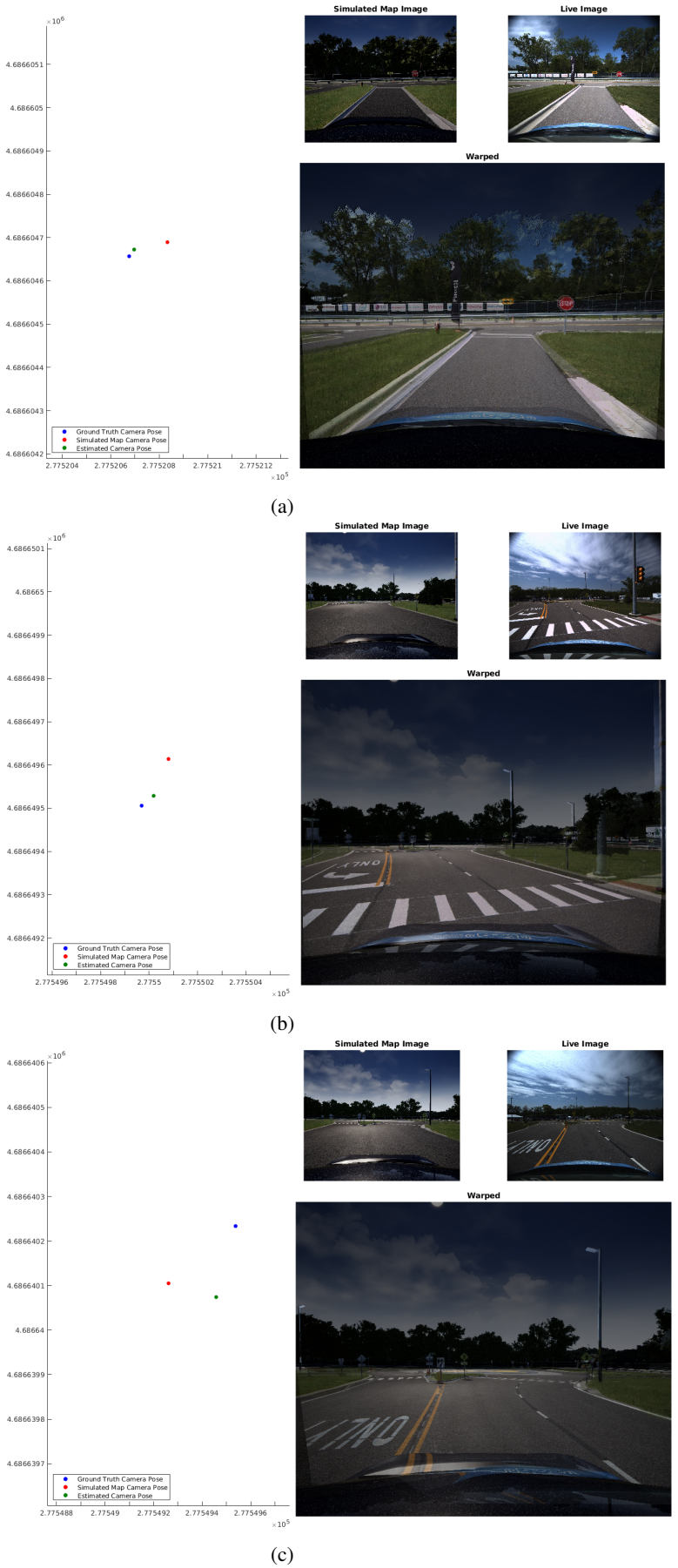
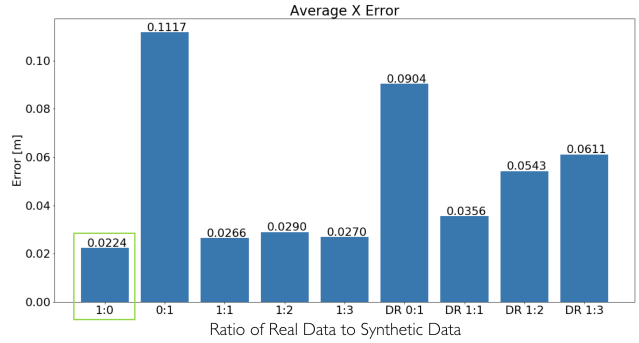
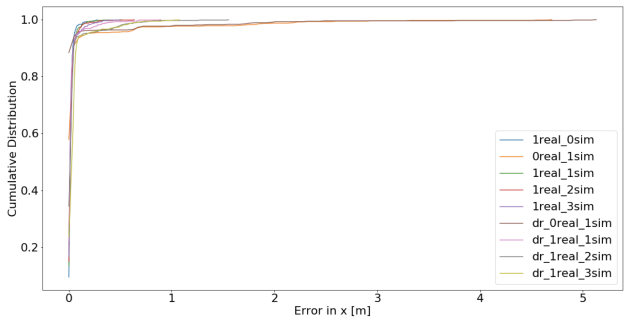


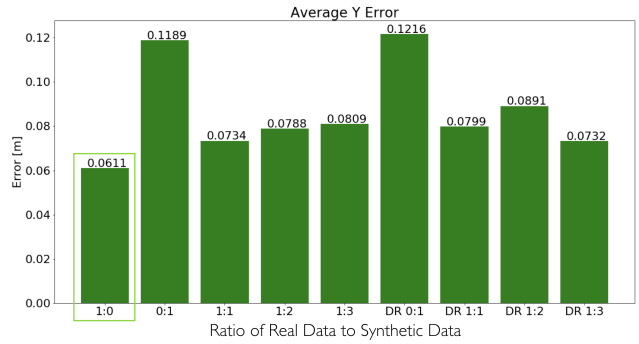
Fig. 7: Performance of the network trained only on real data on random examples from the test set.



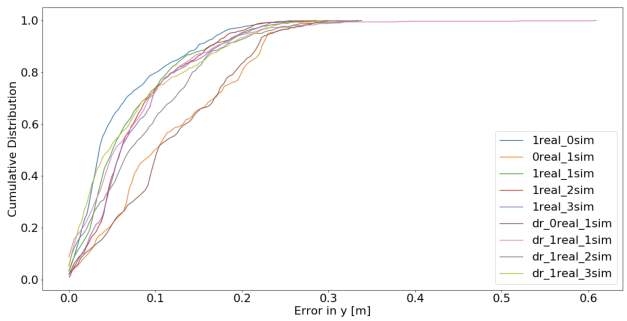
(a) Average x RMSE



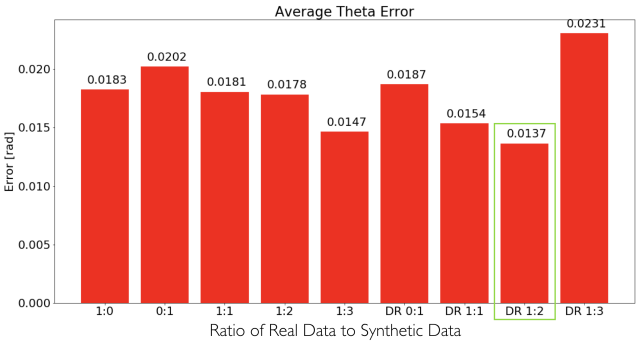
(b) Cumulative x RMSE Distribution



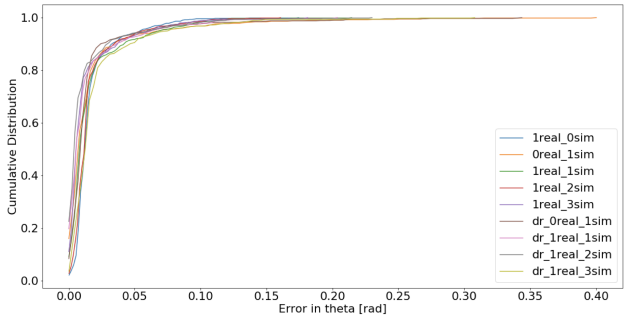
(c) Average y RMSE



(d) Cumulative y RMSE Distribution



(e) Average θ RMSE



(f) Cumulative θ RMSE Distributions

Fig. 8: Plots showing the RMSE for each DoF under different training conditions.

TABLE I: Summary of results and training conditions

ID	Number of Real Pairs	Number of Synthetic Pairs	Domain Randomization	Average RMSE x [m]	Average RMSE y [m]	Average RMS Error θ [rad]
1:0	8688	0	NA	**0.0224**	**0.0611**	0.0183
0:1	0	8688	No	0.1117	0.1189	0.0202
1:1	8688	8688	No	0.0266	0.0734	0.0181
1:2	8688	17376	No	0.0290	0.0788	0.0178
1:3	8688	25437	No	0.0270	0.0809	0.0147
DR 0:1	0	8688	Yes	0.0904	0.1216	0.0187
DR 1:1	8688	8688	Yes	0.0356	0.0799	0.0154
DR 1:2	8688	17376	Yes	0.0543	0.0891	**0.0137**
DR 1:3	8688	25437	Yes	0.0611	0.0732	0.0231

Figure 7a shows an example of decent performance from the network. The signs and curbs in the image overlay are closely aligned. Figure 7c shows a case where the performance is much worse. In the image overlay, we can see an offset in the curb on the middle island and the left side of the road. Interestingly, in Figure 7b the estimate of the real camera pose and the ground truth camera pose are not that far off but in the image overlay there are significant offsets between the curbs. This may suggests some discrepancies between the simulation and the physical world. As well, you can notice lane markings in the real images that are not present in the simulation.

V. Conclusions and Future Work

Overall, it seems that adding synthetic data during training did not improve the performance of the network. This may be because the task is too easy. Future work could look at using training and testing samples with a larger distance between the real image and the map image, instead of just using the closest map image. This would likely lead to increased robustness when integrated with the rest of the localization pipeline. That is, the initial pose estimate could be further away from the true position, which could lead to a poor initial choice of map image, but the network would still be able to compute the relative pose change and correct for this accumulated error. When the network is tasked with this harder problem the synthetic training data may have a positive impact.

Another way the task could be made harder is by separating the data by path. Currently, the test set is created by randomly selecting samples from the entire real dataset. Instead, a specific path could be held out for testing. Each of the paths were captured at different times and large sections of each path do not overlap with the other paths. As a result the network would have few training samples from the area of Mcity being tested. This would likely decrease performance on the test set significantly. It would also give a better indication of how the network would be able to generalize to unseen data. This could be the kind of situation where the addition of synthetic data would actually make a significant impact.

Another possible reason, aside from the task being too easy, that a significant difference when adding synthetic data was not seen is that more data was needed. Most works using domain randomization use hundreds of thousands of examples; this work had less than 26,000 pairs in the largest training set. Also, it may be better to use synthetic examples with a variety of lighting conditions instead of randomly changing the colours. The RightHook simulation has the ability to render samples at any time of day.

This work was focused primarily on computing the relative pose estimate not on how this measurement could be integrated into a full localization pipeline. It would be interesting to see how much this method could correct a trajectory that relies solely on dead reckoning or a noisy GPS. Likely, further effort would be needed to increase the robustness of this method such that it could still perform accurately given a bad initial guess.

There is also area to explore the network architecture. The current architecture was selected based on its use in [6], but that work only explored two images taken at the same time from different poses. Instead, more like [14], a network that does not share weights across the feature extraction branches could be used. This has the potential to let each branch specialize in extracting features from each type of medium, real or simulated.

Lastly, another item to note is that this method is entirely reliant on the existence of a high fidelity simulation environment. Currently, RightHook only has environments created for Mcity and a few neighbourhoods in California. Should this method be used outside those areas, a new simulation environment would need to be created first.

References

[1] RightHook, Inc. www.righthook.io

[2] P. Furgale and T. D. Barfoot, "Visual Teach and Repeat for Long Range Rover Autonomy," Journal of Field Robotics, vol. 27, no. 5, pp. 534–560, 2010.

[3] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features,"in European conference on computer vision. Springer,2006, pp. 404–417.

[4] Geoffrey Pascoe, Will Maddern, Alexander Stewart, and Paul Newman. "FARLAP: Fast robust localisa- tion using appearance priors." volume 2015, 05 2015.

[5] G. Pascoe, W. Maddern, and P. Newman, "Robust Direct Visual Localisation using Normalised Information Distance," in British Machine Vision Conference (BMVC), (Swansea, Wales), pp. 70.1–70.13, 2015

[6] . Melekhov, J. Ylioinas, J. Kannala, and E. Rahtu, "Relative camera pose estimation using convolutional neural networks," in International Conference on Advanced Concepts for Intelligent Vision Systems. Springer, 2017, pp. 675–687.

[7] . Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in neural information processing systems, 2012, pp. 1097–1105.

[8] V. Mohanty, S. Agrawal, S. Datta, A. Ghosh, V. D. Sharma, and D. Chakravarty, "Deepvo: A deep learning approach for monocular visual odometry," ArXiv, vol. abs/1611.06069, 2016.

[9] S.Wang, R.Clark, H.Wen, and N.Trigoni,"End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks," The International Journal of Robotics Research, vol. 37, no. 4-5, pp. 513–542, 2018

[10] V.Peretroukhin andJ.Kelly,"Dpc-net:Deep pose correction for visual localization," IEEE Robotics and Automation Letters, vol. 3, no. 3, pp. 2424–2431, July 2018.

[11] Zhou, T., Brown, M., Snavely, N., and Lowe, D. G., "Unsupervised Learning of Depth and Ego-Motion From Video,"The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[12] Gridseth, M. and Barfoot, T. D., "DeepMEL: Compiling Visual Multi-Experience Local-ization into a Deep Neural Network," 2019.

[13] L. Clement and J. Kelly, "How to train a CAT: Learning canonical appearance transformations for robust direct localization under illumination change," IEEE Robotics and Automation Letters (RA-L), 3(3): 2424-2431, 2018

[14] Y. Aytar, L. Castrejon, C. Vondrick, H. Pirsiavash, and A. Torralba. "Cross-modal scene networks," CoRR, abs/1610.09003, 2016.

[15] Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P., "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World," CoRR, 2017.

[16] Ren, X., Luo, J., Solowjow, E., Ojea, J. A., Gupta, A., Tamar, A., and Abbeel, P., "Domain Randomization for Active Pose Estimation," CoRR, 2019.

[17] James, S., Wohlhart, P., Kalakrishnan, M., Kalashnikov, D., Irpan, A., Ibarz, J., Levine, S., Hadsell, R., and Bousmalis, K., "Sim-to-Real via Sim-to-Sim: Data-efficient Robotic Grasping via Randomized-to-Canonical Adaptation Networks," 2018.

[18] K.He, X.Zhang, S.Ren, and J.Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 37, no. 9, pp. 1904–1916, sep 2015.