GEN-SLAM: Generative Modeling for Monocular Simultaneous Localization and Mapping

Punarjay Chakravarty[1], Praveen Narayanan[1] and Tom Roussel.[2]

***Abstract*— We present a Deep Learning based system for the twin tasks of localization and obstacle avoidance essential to any mobile robot. Our system learns from conventional geometric SLAM, and outputs, using a single camera, the topological pose of the camera in an environment, and the depth map of obstacles around it. We use a CNN to localize in a topological map, and a conditional VAE to output depth for a camera image, conditional on this topological location estimation. We demonstrate the effectiveness of our monocular localization and depth estimation system on simulated and real datasets.**

I. INTRODUCTION

The determination of one's position in an environment and the location of other obstacles around the ego-sensor, are two of the primary tasks for the sensing system of any robot - be it a service robot in a warehouse or an Autonomous Vehicle (AV) in a smart city. The sensing used for this is most commonly some kind of depth sensor - stereoscopic camera or a LIDAR sensor. The computational and dollar costs of operating these sensors are high. LIDAR sensors still cost in the multiple thousands of dollars, depth cameras don't work well in outdoor lighting conditions and depth obtained from stereo is often noisy. In this paper, we present a method that uses monocular sensing for localization and depth map estimation. A depth sensor is used to train a network for a particular environment, after which monocular vision is used to obtain the camera's position in that environment and determine a depth-map of obstacles around it.

We use a generative model to learn the joint distribution $p(x_{rgb}, x_{dep})$ between RGB and depth images instead of the traditional discriminative models that estimate $p(x_{dep}|x_{rgb})$. In what is possibly the first use of generative models for depth estimation in a SLAM context, we use a Variational Autoencoder (VAE) [1] to output a depth map corresponding to an RGB image. Furthermore, we condition this depth generation on the location of the image in the environment. We obtain this location using a standard CNN, trained to output the camera location as one of N topological nodes from the environment.

We are motivated by the underlying assumption that the 3D geometric scene (the depth map) and its 2D projection into the camera (the RGB image) share a common latent subspace. A latent vector sampled from this sub-space, conditioned on location information, should be able to generate both the depth map and its corresponding RGB image for that particular location in the environment.

[1] The authors are with Ford Greenfield Labs, Palo Alto, USA. pchakra5@ford.com,pnaray11@ford.com

[2] Work done as an intern for Ford PA, affiliated with KULeuven, Belgium. tom.roussel@esat.kuleuven.be

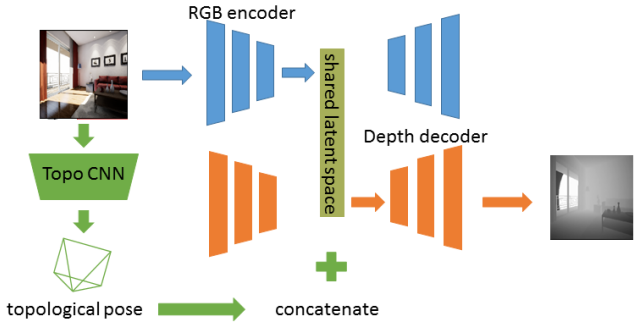

Fig. 1: The GEN-SLAM architecture, a conditional VAE with shared latent space in test configuration. The paired RGB and Depth VAEs are indicated in blue and orange respectively. The RGB image is passed through the Topo-CNN to get topological pose, and simultaneously through the RGB encoder to get the latent vector for this image. This shared latent vector is concatenated with the topological pose, and decoded through the Depth decoder, to get the depth map for the RGB image.

Our Conditional VAE (CVAE) architecture (Figure 1) comprises two encoder-decoder networks, with a shared latent variable, that is conditioned on camera location. The model learns this shared latent representation, with the assumption that latent representations of the RGB image and its paired depth map should coincide. During training, the encoders encode corresponding RGB and depth training samples into a common latent vector, which are then decoded to reconstruct both RGB and depth. Within and cross-domain losses are used to train the network. At test time, the model takes an RGB image, passes it through the RGB encoder, and decodes it using the depth decoder to get the depth map for that image.

Our system trains with a small amount of data. In our experiments, we show that a small number (under 10) laps of a route are enough to train the network to localize in an environment and generate depth maps for each location along its route.

We suggest that this kind of location-aware depth estimation is a useful solution for robots that operate in pre-mapped environments. Topological localization [2] is a useful navigation tool, and when used to condition the generation of depth maps from monocular images, it results in better depth maps than without this prior. There are a lot of applications for robots that operate in environments that do not change too much over time. Shelving robots in a warehouse, surveillance robots in a factory, L4 AVs in a geo-fenced urban environment or vacuum-cleaner robots in

the home, all operate in areas that can be mapped in advance. This mapping can be done with more expensive sensing - a "mapper" vehicle could have a depth camera or a LIDAR, and cheaper, "localizer" vehicles or drones, equipped with monocular vision, can follow pre-mapped routes in that environment and use a system like ours to determine position along that route, as well as get a depth map for local path planning and collision avoidance.

II. RELATED WORK

A. SLAM and Localization

SLAM, initially invented by Leonard et. al. [3] based on earlier work by Smith et. al. [4], is the science of estimating one's position in an incrementally built map. Early work was based on single beam LIDAR and sonar, but vision based techniques [5], [6], [7] were popularized in the last decade. These papers estimate position in continuous space, which is called metric localization. Topological localization [2] refers localization in discrete space, i.e., the localization of a robot closest to a topological node in the environment, and this is done to build more compact maps. When a robot is navigating down a long, featureless corridor for instance, it might not be necessary to know exactly where it is in the corridor, but only that it has reached one of two topological nodes at either end of the corridor. In many practical situations, the robot or the sensor operates in previously known environments, and so the mapping and localization aspects of SLAM are separated from each other. However, even if an environment is completely mapped, changes in the environment and dynamic obstacles often mean that the ability to ascertain the distances to unmapped objects in the vicinity of the robot are essential for planning collision-free paths through an environment.

In this paper, we use visual metric localization (ORB-SLAM2) [7] to train a CNN for topological localization, and also use this topological pose as a conditioning input to our generative model, the CVAE, for the generation of dense depth maps from RGB images.

B. Single image depth

There has been a glut of work in the past 5 years, that takes a Deep Learning approach to the problem of depth perception from a single image [8], [9], [10], [11], [12], [13], [14], [15].

In the supervised case [8], [13], thousands of image-depth map tuples collected from a depth sensor (like the Microsoft Kinect) are used to train a Convolutional Neural Network (CNN) to determine depth from a single image. However, this requires the use of depth cameras, which is not always available.

More recently, there has been work that uses self-supervision to learn depth in the scene. For example, Garg et. al. [9] uses the depth map output from the CNN and the left image of a stereo pair to recreate the right image. The reconstructed right image is then compared with the original right image, and this loss is used to train the network. This obviates the need for actual depth maps (which might be noisy), and uses the left and right images from a stereo camera directly. Godard et. al. [11] improves upon this work by introducing left-right consistency in both directions. SfmLearner, by Zhou et. al. [10] works with monocular sequences for training. It estimates both depth and the camera movement. The depth map and pose output by the network is used to warp an image in a sequence of images coming from a moving camera to its next image in the sequence. The loss between the current image warped to its successor (using the depth map) and the actual successor image is used to train the network.

DeMon [12] and UndeepVO [14] are recent approaches that also determine both the depth and the motion: rotation/translation matrix $[R \mid t]$ of the monocular camera between frames. Both approaches are also able to determine the absolute scale of the depth map estimated because they use ground truth depth and stereo data respectively.

Mat et. al. [15] uses the sparse depth as would be available from a feature based SLAM system (depth at feature points) to generate a dense depth map. Depth at feature points that would be obtained from a system like ORB-SLAM2 [7] are used as an additional input to the CNN, along with the RGB image to generate a dense depth map.

Most of these modern methods use an encoder-decoder architecture for single image depth, and each branch of our twin CVAE uses the same architecture.

C. Deep Generative Models

Deep generative modeling techniques have shown great promise in many application domains in computer vision [16], [17], [18], [19], [20], [21], [22], [23], [24], natural language processing [25], [26], [27], speech [28], [29], [30] and reinforcement learning [31], [32].

Two commonly used classes of Deep Generative models are Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) [1], [33], which are both latent variable models that can construct the data distribution $p(x)$ from a much lower dimensional latent variable distribution $p(z)$. VAEs maximize a variational lower bound for the data distribution, resulting in an encoder-decoder formulation wherein a recognition model (the encoder) maps the data into a lower dimensional latent space, and a reconstruction model or decoder (sometimes called a generator) reconstructs the input, thereby training it as an autoencoder. It differs from an autoencoder in that it one can sample from the latent gaussian prior $z \sim \mathcal{N}(0, I)$, to produce new samples from the data. At test time, the setup becomes a generative model in which data can be produced by sampling from the latent variable distribution $z \sim \mathcal{N}(0, I)$.

GANs approach the problem of generating samples from the data as a two player adversarial (minmax) game between two networks - a generator G whose aim is to produce data from the data distribution, and a discriminator which classifies data produced by G as real or fake. When the process reaches equilibrium, the generator has learned to fool the discriminator, at which point the samples produced by G mimic those coming from the data distribution. Data

samples are produced by sampling a noise variable $\mathcal{N}(0, I)$ and sending it to the generator to obtain $G(z)$.

GANs and VAEs are generative models with somewhat different origins and motivations. VAEs provide a tractable latent variable formulation, are easy to train, but the samples resulting are known to be somewhat blurry. GANs produce excellent, realistic looking samples but can suffer from mode collapse and training instability. In order to get the best of both worlds, hybrid VAE-GAN models [34] have been proposed so as to obtain a tractable latent variable model with the VAE and use GANs as a polishing mechanism in order to produce good, non-blurry samples.

A VAE-GAN framework with shared latent space [35], [22], [23] has recently been used for unsupervised domain adaptation. Images from one domain are transferred into the shared latent space using domain 1's encoder, and decoded into the second domain using domain 2's decoder. Once the domain transfer has been carried out in one direction, it is then carried out in the opposite direction to get back the original image. The difference between the original image in domain 1 its reconstruction, after a cycle to the 2nd domain and back (first introduced by [21] as a Cycle Consistency loss), is used to train the network without paired images between domains.

In this paper, we use paired VAEs with shared latent space to generate depth maps from RGB images. We add localization as an added conditioning constraint, and are able to generate images and depth maps for topological nodes in the environment. The following section describes our model in more detail.

III. GENERATIVE MODELING FOR SLAM

A. Paired Variational Auto Encoders (VAEs) for RGB and Depth

Our model has an encoder-decoder architecture with shared latent space for the VAE, as shown in Figure 4. This paired VAE has 2 encoders and 2 decoders, one each for the RGB and depth images respectively. The encoders encode input RGB images and depth maps into a shared latent space representation, which forces a shared reduced representation of both domains. This latent representation is then decoded through the RGB and depth decoders to give the RGB and depth reconstructions respectively.

We employ four reconstruction losses to train the network: within-domain reconstruction losses, $\mathcal{L}_{x_{rgb}\rightarrow x_{rgb}}$ & $\mathcal{L}_{x_{dep}\rightarrow x_{dep}}$ and the cross-domain reconstruction losses, $\mathcal{L}_{x_{rgb}\rightarrow x_{dep}}$ & $\mathcal{L}_{x_{dep}\rightarrow x_{rgb}}$ shown below:

$$\mathcal{L}_{x_{rgb}\rightarrow x_{rgb}} = \| x_{rgb} - G_{rgb}(E_{rgb}(x_{rgb})) \|_{L2} \quad (1)$$

$$\mathcal{L}_{x_{dep}\rightarrow x_{dep}} = \| x_{dep} - G_{dep}(E_{dep}(x_{dep})) \|_{L2} \quad (2)$$

$$\mathcal{L}_{x_{rgb}\rightarrow x_{dep}} = \| x_{dep} - G_{dep}(E_{rgb}(x_{rgb})) \|_{L2} \quad (3)$$

$$\mathcal{L}_{x_{dep}\rightarrow x_{rgb}} = \| x_{rgb} - G_{rgb}(E_{dep}(x_{dep})) \|_{L2} \quad (4)$$

where x_{rgb} and x_{dep} are paired RGB and depth maps used for training and E and G are the Encoders and the decoders (Generators) respectively.

Just using the above losses, we would get a standard AutoEncoder (AE). The VAE enforces the latent distribution $q(z)$ to be Gaussian, so that we can sample from it and generate new RGB and depth maps from a trained network. It uses a Kullback-Leibler (KL) distance (and loss) to make the input-conditioned latent distribution $q(z|x)$ to be as close as possible to a 0 mean, unit variance Normal distribution p_n.

The VAE formulation for the within and cross-domain losses are given as follows:

$$\mathcal{L}_{x_{rgb}\rightarrow x_{rgb}} = KL(q_{rgb}(z_{rgb}|x_{rgb}) \parallel p_n(z)) - \mathbb{E}_{z_{rgb}\sim q_{rgb}(z_{rgb}|x_{rgb})}[log\ p_{G_{rgb}}(x_{rgb}|z_{rgb})] \quad (5)$$

$$\mathcal{L}_{x_{dep}\rightarrow x_{dep}} = KL(q_{dep}(z_{dep}|x_{dep}) \parallel p_n(z)) - \mathbb{E}_{z_{dep}\sim q_{dep}(z_{dep}|x_{dep})}[log\ p_{G_{dep}}(x_{dep}|z_{dep})] \quad (6)$$

$$\mathcal{L}_{x_{rgb}\rightarrow x_{dep}} = KL(q_{rgb}(z_{rgb}|x_{rgb}) \parallel p_n(z)) - \mathbb{E}_{z_{rgb}\sim q_{rgb}(z_{rgb}|x_{rgb})}[log\ p_{G_{dep}}(x_{dep}|z_{rgb})] \quad (7)$$

$$\mathcal{L}_{x_{dep}\rightarrow x_{rgb}} = KL(q_{dep}(z_{dep}|x_{dep}) \parallel p_n(z)) - \mathbb{E}_{z_{dep}\sim q_{dep}(z_{dep}|x_{dep})}[log\ p_{G_{rgb}}(x_{rgb}|z_{dep})] \quad (8)$$

where the first terms for each loss value are the KL-divergence terms enforcing the Gaussian constraint on the latent distribution $q(z|x)$ and the second terms are the within and cross domain reconstruction terms. For VAEs, encoding of an input into the latent representation is non-deterministic. For the RGB domain, this is represented by the distribution $q_{rgb}(z_{rgb}|x_{rgb})$, which can be sampled to give a z_{rgb}. This is then passed through the RGB generator G_{rgb} to give a distribution $p_{G_{rgb}}(x_{rgb}|z_{rgb})$. The expected log value of this distribution over multiple samples is the same as the reconstruction RGB reconstruction loss $\mathcal{L}_{x_{rgb}\rightarrow x_{rgb}}$ in equation 1, and the same holds for the other within and cross domain reconstruction losses. For details of this derivation, see [1]. Equations 1-4 are used to motivate our discussion on auto-encoders, but we finally use the VAE versions 5-8 to train our networks. Using these loss functions, we solve the following joint optimization problem in the RGB and depth domains:

$$\min_{E_{rgb},G_{rgb},E_{dep},G_{dep}} \mathcal{L}_{x_{rgb}\rightarrow x_{rgb}}(E_{rgb}, G_{rgb}) + \mathcal{L}_{x_{dep}\rightarrow x_{dep}}(E_{dep}, G_{dep}) + \mathcal{L}_{x_{rgb}\rightarrow x_{dep}}(E_{rgb}, G_{dep}) + \mathcal{L}_{x_{dep}\rightarrow x_{rgb}}(E_{dep}, G_{rgb}) \quad (9)$$

B. Conditional VAE (CVAE)

The paired VAE for RGB and depth domains is extended with a conditional formulation [16] that can be used when the data has additional labels that would be beneficial to the problem. We use the topological node associated with each image as the conditioning input, the assumption being that rgb to depth conversion conditioned on the location information is better than without. This is done in practice by passing the input (RGB or depth) through the encoder to generate the latent code and concatenating the latent

code with a one-hot encoding of the topological node. This latent vector, concatenated with location information is then decoded through the rgb and depth decoders as described earlier. This conditioning also allows us to generate new RGB images and depth maps for the environment, given a particular location. This allows us to hallucinate images for a given topological node, or location in an environment (Figure 7).

IV. IMPLEMENTATION DETAILS

A. Data Collection

We generate a synthetic dataset comprising of RGB and depth images using the Unreal gaming engine UE4 [36], and a plugin, UnrealCV [37]. UnrealCV allows us to use OpenCV to programmatically communicate with Unreal Engine and fly the camera through pre-defined trajectories and save RGB images, depth maps and camera poses. We use a living room model from the Unreal Marketplace, and call this the "Living Room" dataset.

For real imagery, we desire data where there are not too many dynamic obstacles, and where we can repeat the same route multiple times. We generate our own datasets by using the StereoZed camera [38], mounting it on a trolley and driving it around the indoor environment of our office. We collect data by doing loops around a lab and a longer loop around the office, and call these the "Lab" and "Corridor" environments respectively.

B. Topological map Generation

We use ORB-SLAM2 [7] as the conventional SLAM algorithm that supervises the training of our Deep networks. ORB-SLAM2 operates on the stereo image pairs generated by the StereoZed camera and outputs a camera trajectory. To ensure route repeatability, we had to save maps generated by ORBSLAM2 and re-load them for localization for another sequence captured at a different time, along the same route. The original ORBSLAM2 code [39] does not come with the ability to save maps. We extended it to allow serializing the map to disk, followed by map re-load and localizing a new sequence of images in the previously saved map. A topological node is set to be at every 1.5m along the metric route measured by ORB-SLAM2.

C. Network details for the CVAE-model

We use the PyTorch framework for coding our CVAE, which is made location-aware by concatenating the one-hot encoded topological node associated with each image to the latent vector for that image. We use an additional CNN that operates on the input RGB image, and outputs the topological node as a classification output. We use Alex-net, pre-trained on ImageNet, downloaded from the pytorch model zoo. We remove the last classifier layer, and substitute it with our own, with the number of outputs equal to the number of topological nodes. This pre-trained network is then fine-tuned with (RGB image, topological node) pairs. Our CVAE uses convolutional and de-convolutional layers (Figure 2) with instance norm (INS), leaky RELU and residual (Res) layer

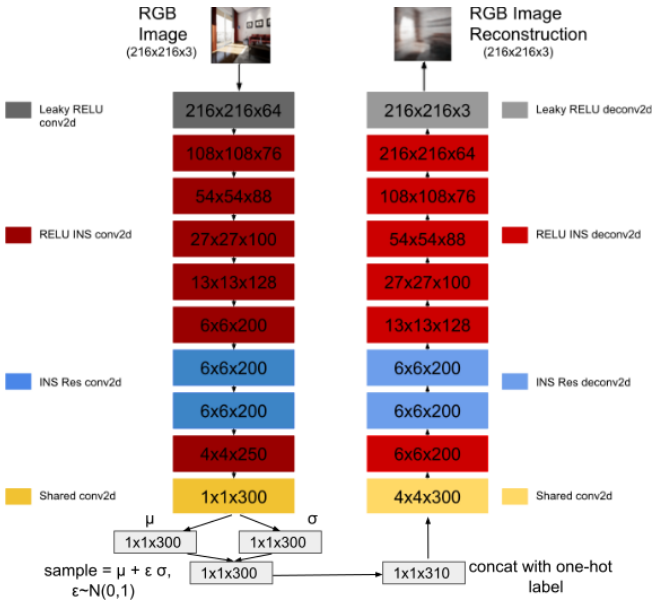

Fig. 2: Output tensor sizes for each layer of the RGB VAE. Layers are coloured according to type, with encoder layers (first column) being a darker shade compared to the corresponding decoder layers (second column). The Depth VAE (not shown here) is identical and shares the yellow convolutional layers with the RGB VAE. The topological node in the map the image belongs to is concatenated as a one-hot vector with the sample from the encoder.

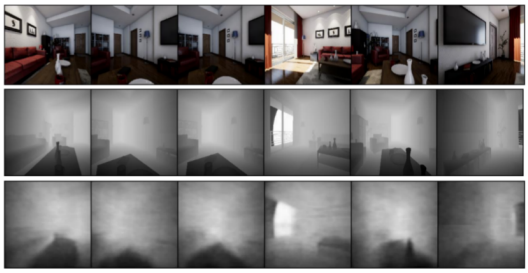

Fig. 3: Ground truth (middle row) and estimated (bottom row) depth maps for RGB images (top row) from the Living Room dataset.

modifications. Training of the CVAE is done with (RGB image, depth image, topological node) tuples.

V. EXPERIMENTAL RESULTS

We test our proposed method on the data we collected as described previously. For the simulated "Living Room" dataset, we generate data by letting the camera travel in trajectories with increments of 0.5 metre offsets + random noise on either side of the reference camera trajectory. For the "Lab" and "Corridor" datasets we collect data by driving laps of the same loop (12 laps for Lab and 7 laps for Corridor), but with the environment slightly altered (e.g. moving chairs) at different times of the day. We divide these loops into train and test sets using a 90/10 split divided evenly across topological nodes.

Since we use stereo to generate depth maps for Lab and Corridor, there are holes in them, which can negatively impact the training procedure. To address this we use a hole

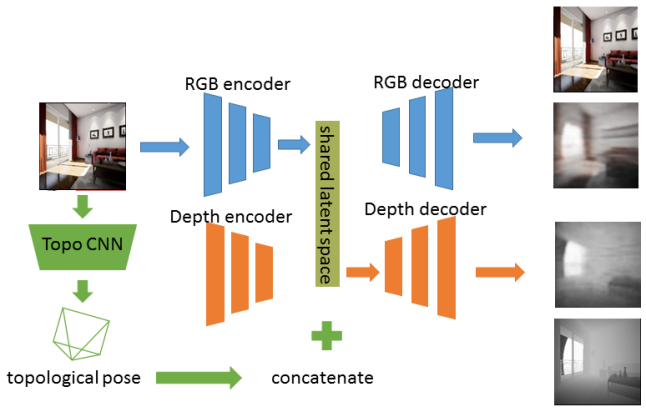

Fig. 4: The GEN-SLAM model, with RGB and Depth encoders and decoders, and a shared latent space. It is trained with tuples of rgb, depth and topological pose.

filling algorithm [40] to create dense depth maps.

Figures 5 and 6 show topological localization and depth map generation for Lab and Corridor. Left to right along each of the 4 rows: input RGB image, ground truth depth and estimated depth. Centre: topological nodes in green, metric location (for illustrative purposes, from ORBSLAM2) in red and numbers next to nodes correspond to the 4 rows of RGB $\rightarrow$ depth reconstruction images.

We show how well our method performs using metrics from [9], [10], [11], the current standard for evaluating depth estimation. These metrics are: RMSE, log RMSE, absolute relative difference, squared relative difference and accuracy with different thresholds. The absolute relative distance is defined as: $\frac{1}{N}\sum_d |d_{estim} - d_{gt}|/d_{gt}$ and the squared relative difference is defined as: $\frac{1}{N}\sum_d \|d_{estim} - d_{gt}\|^2/d_{gt}$. Accuracy in this context is the ratio of 'correct' depth pixels and the total number of depth pixels. A depth pixel is considered correct if: $max(\frac{d_{gt}}{d_{estim}}, \frac{d_{estim}}{d_{gt}}) < \delta$, with δ being a selected threshold. If the threshold is higher, the metric is less strict.

We show how well our Topological Pose CNN performs using 2 metrics: accuracy and off-by-one accuracy. Accuracy is simply the classification accuracy of our method. The off-by-one accuracy counts the assignment of an image to a node that is adjacent to its actual node as correct. We do this because from a visual standpoint, there is some ambiguity of where one node stops and the other begins.

The performance of our depth estimator is shown in quantitatively in Table I. Aside from the metrics we add the mean ground truth depth values for each dataset, to give some context to the scale of the different environments. Corridor has a larger mean depth compared to Lab, and hence will naturally result in larger RMSE values when estimating depth, though the other metrics are less sensitive to this. The depth values in our simulated dataset Living Room, are not in meters, which partly the reason both RMSE and log RMSE are significantly lower than the other datasets. This environment is also much less complex than our real environments, resulting in more accurate depth estimates. All our non-simulated metrics are computed in meters.

The topological pose CNN accuracy (Table II) is nearly 100% across datasets.

We have a video demonstration of the experiments at https://youtu.be/WGuB1cO0mCY

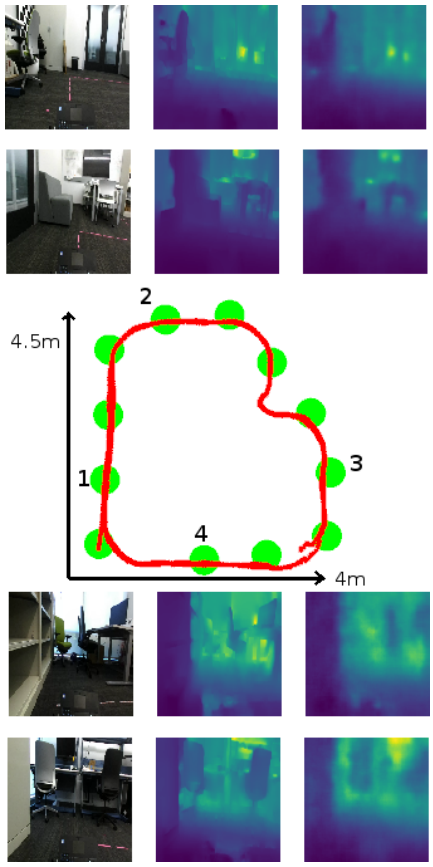

Fig. 5: Topological localization and depth map estimation in the Lab dataset. Along rows: rgb, ground truth and estimated depths for 4 topological nodes in map. Metric localization path (red) is obtained from ORBSLAM2.

VI. DISCUSSION AND FUTURE WORK

One might ask why we use a VAE and not a regular encoder-decoder architecture that has so far been used for single image depth. The VAE forces the latent vector to conform to a distribution, a Gaussian. In our case, both the 3D geometry of the scene (depth map) and its 2D projection (RGB image) conditioned on topological pose, are forced to belong to the same distribution, and the network internalizes the connection between location in an environment, its geometry and appearance, which is embedded in the latent space manifold that the network learns. Figure 7 shows the results of sampling RGB images and depth maps from the CVAE for the Living Room dataset. These sampled images are created by sampling z from random noise, concatenating with the topological label, and then passing $[z|label]$ to the RGB and depth decoders. This visualization of sampling from the latent space manifold shows that the network has learnt the appearance and underlying geometry of each topological node.

The twin VAE architecture allows us to use weak supervision to train our networks. Here, we have used noisy

TABLE I: Depth estimation metrics across datasets: Living Room, Lab and Corridor.

		Lower is better				Higher is better		
						Accuracies		
Dataset	Mean depth	RMSE	log RMSE	Abs. Rel.	Sq. Rel.	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Living room	0.69	0.031	0.083	0.039	0.002	0.968	0.994	0.998
Lab [m]	1.85	0.473	0.159	0.091	0.067	0.922	0.980	0.992
Corridor [m]	4	0.908	0.211	0.140	0.183	0.844	0.956	0.980

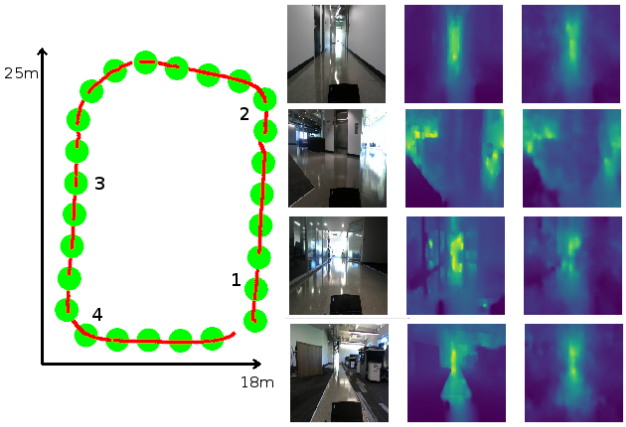

Fig. 6: Topological localization and depth map estimation at 4 nodes in the Corridor dataset.

TABLE II: Topological CNN localization accuracy across datasets

Dataset	Accuracy	Off-by-one accuracy
Living Room	94.35%	100%
Lab	85.77%	99.72%
Corridor	97%	100%

depth maps obtained from the stereoZed camera to train our networks. using both within domain and cross domain losses, as opposed to the single cross domain loss that would be used in a regular encoder-decoder architecture that goes from RGB $\rightarrow$ depth domains.

Instead of training using the noisy stereo depth maps, we could use the left and right images of the stereo pair, and train with reconstruction of the right image, given the depth map (produced by the network) [11]. We could also train with reconstruction of images over time, as the camera moves through its environment [10]. Cycle Consistency losses [21] could also be used to go from RGB $\rightarrow$ depth $\rightarrow$ RGB, and its converse, depth $\rightarrow$ RGB $\rightarrow$ depth. These additional loss functions will be examined in future work and should produce better training supervision and allow us to generate better reconstructions of depth in the absence of good training data.

The topological pose conditioning in this paper could be augmented with additional priors like object detection and lighting for better depth estimation across dynamic environments and different lighting conditions.

Here, our generation of topological maps is relatively simplistic, and operates on single-looped paths. Future work will examine topological map generation for more complicated routes and multi-edged junctions.

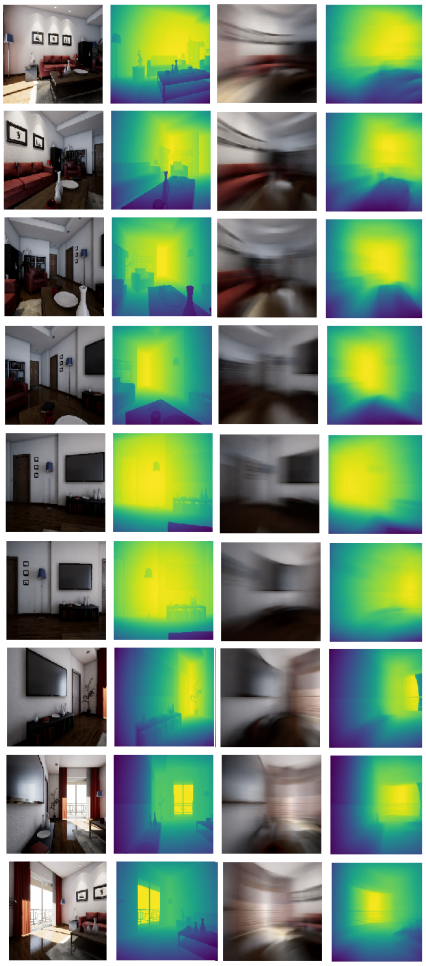

Fig. 7: RGB, depth, sampled RGB, sampled depth, for each topological node in the Living Room dataset.

VII. CONCLUSIONS

Mobile robots need to solve the twin tasks of localization and estimating the distance to various obstacles around them to plan collision free paths through an environment. We present a Deep Learning based model that solves both tasks using only monocular vision. Our Deep Generative model is trained using conventional visual SLAM, and at test time, takes in an RGB image, outputs its topological pose and also generates the depth map for that image, conditioned on that pose. Such a system could be used in an environment where there is a requirement for the operation of multiple robots with cheap sensing. A factory environment, say a fulfillment centre for an e-commerce giant, with hundreds of thousands of square metres of shelving and inventory, and robots to move boxes around, could be mapped once, using depth sensing and conventional geometric SLAM, and this map could be used to train our model. Subsequently, hundreds

of robots with single low-cost cameras as sensors could use such a trained model to sense and navigate their way around the factory.

REFERENCES

[1] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[2] I. Ulrich and I. Nourbakhsh, "Appearance-based place recognition for topological localization," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 2. Ieee, 2000, pp. 1023–1029.

[3] J. J. Leonard and H. F. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *IEEE Transactions on robotics and Automation*, vol. 7, no. 3, pp. 376–382, 1991.

[4] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," in *Machine intelligence and pattern recognition*. Elsevier, 1988, vol. 5, pp. 435–461.

[5] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2320–2327.

[6] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European Conference on Computer Vision*. Springer, 2014, pp. 834–849.

[7] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[8] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Advances in neural information processing systems*, 2014, pp. 2366–2374.

[9] R. Garg, V. K. BG, G. Carneiro, and I. Reid, "Unsupervised cnn for single view depth estimation: Geometry to the rescue," in *European Conference on Computer Vision*. Springer, 2016, pp. 740–756.

[10] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *CVPR*, vol. 2, no. 6, 2017, p. 7.

[11] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *CVPR*, vol. 2, no. 6, 2017, p. 7.

[12] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox, "Demon: Depth and motion network for learning monocular stereo," in *IEEE Conference on computer vision and pattern recognition (CVPR)*, vol. 5, 2017, p. 6.

[13] P. Chakravarty, K. Kelchtermans, T. Roussel, S. Wellens, T. Tuytelaars, and L. Van Eycken, "Cnn-based single image obstacle avoidance on a quadrotor," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 6369–6374.

[14] R. Li, S. Wang, Z. Long, and D. Gu, "Undeepvo: Monocular visual odometry through unsupervised deep learning," *arXiv preprint arXiv:1709.06841*, 2017.

[15] F. Ma and S. Karaman, "Sparse-to-dense: depth prediction from sparse depth samples and a single image," *arXiv preprint arXiv:1709.07492*, 2017.

[16] J. Walker, C. Doersch, A. Gupta, and H. Martial, "An uncertain future: Forecasting from static images using variational autoencoders," *arXiv preprint arXiv:1606.07873*, 2016.

[17] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015.

[18] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, "Draw, a recurrent neural network for image generation," *arXiv preprint arXiv:1502.04623*, 2015.

[19] S. M. Eslami, Ali, J. Rezende, F. Besse, F. Viola, A. S. Morcos, M. Garnelo, A. Ruderman, A. A. Rusu, I. Denihelka, K. Gregor, D. P. Reichert, L. Buesing, T. Weber, O. Vinyals, D. Rosenbaum, N. Rabinowitz, H. King, C. Hillier, M. Botvinick, D. Wierstra, K. Kavukcuoglu, and D. Hassabis, "Neural scene representation and rendering," *Science*, vol. 360, no. 6394, pp. 1204–1210, 2018.

[20] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," 2016.

[21] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," 2017.

[22] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 700–708.

[23] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," *arXiv preprint arXiv:1804.04732*, 2018.

[24] D. Ha and D. Eck, "A neural representation of sketch drawings," *arXiv preprint arXiv:1704.03477*, 2017.

[25] R. Bowman, Samuel, L. Vilnis, O. Vinyals, A. M. Dai, R. Josefowics, and S. Bengio, "Generating sentences from a continious space," *arXiv preprint arXiv:1511.06349*, 2015.

[26] S. Semeniuta, A. Severyn, and E. Barth, "A hybrid convolutional variational autoencoder for text generation," *arXiv preprint arXiv:1702.02390*, 2017.

[27] Y. Zhang, Z. Gan, K. Fan, Z. Chen, R. Henao, D. Shen, and L. Carin, "Adversarial feature matching for text generation."

[28] C.-C. Hsu, H.-T. Hwang, Y.-C. Wu, Y. Tsao, and H.-M. Wang, "Voice conversion from non-parallel corpora using variational auto-encoder," *arXiv preprint arXiv:1610.04019*, 2016.

[29] W.-N. Hsu, Y. Zhang, and J. Glass, "Unsupervised learning of disentangled and interpretable representations from sequential data," *arXiv preprint arXiv:1709.07902*, 2017.

[30] A. Van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," *arXiv preprint arXiv:1711.00937*, 2017.

[31] D. Ha and J. Schmidhüber, "World models," *arXiv preprint arXiv:1803.10122*, 2018.

[32] I. Higgins, A. Pal, A. A. Rusu, L. Matthey, C. P. Burgess, A. Pritzel, M. Botvinick, C. Blundell, and A. Lerchner, "Darla: Improving zero-shot transfer in reinforcement learning," *arXiv preprint arXiv:1707.08475*, 2017.

[33] D. P. Rezende and S. Mohamed, "Stochastic backpropagation and approximate inference in deep generative models," *arXiv preprint arXiv:1401.4082*, 2014.

[34] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," *arXiv preprint arXiv:1512.09300*, 2015.

[35] C. Wan, T. Probst, L. Van Gool, and A. Yao, "Crossing nets: Combining gans and vaes with a shared latent space for hand pose estimation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.

[36] "Unreal engine 4," https://www.unrealengine.com/en-US/what-is-unreal-engine-4.

[37] W. Qiu, F. Zhong, Y. Zhang, S. Qiao, Z. Xiao, T. S. Kim, Y. Wang, and A. Yuille, "Unrealcv: Virtual worlds for computer vision," *ACM Multimedia Open Source Software Competition*, 2017.

[38] "Stereo zed camera," https://www.stereolabs.com/.

[39] "Orbslam 2 code," https://github.com/raulmur/ORB_SLAM2.

[40] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," in *ACM transactions on graphics (tog)*, vol. 23, no. 3. ACM, 2004, pp. 689–694.