arXiv:2404.13288v3 [cs.RO] 7 May 2024

PoseINN: Realtime Visual-based Pose Regression and Localization with Invertible Neural Networks

Zirui Zang, Ahmad Amine, Rahul Mangharam

***Abstract*—Estimating ego-pose from cameras is an important problem in robotics with applications ranging from mobile robotics to augmented reality. While SOTA models are becoming increasingly accurate, they can still be unwieldy due to high computational costs. In this paper, we propose to solve the problem by using invertible neural networks (INN) to find the mapping between the latent space of images and poses for a given scene. Our model achieves similar performance to the SOTA while being faster to train and only requiring offline rendering of low-resolution synthetic data. By using normalizing flows, the proposed method also provides uncertainty estimation for the output. We also demonstrated the efficiency of this method by deploying the model on a mobile robot.**

I. Introduction

Visual pose regression is the task of finding camera poses of images within a trained environment. The matured geometric-based pipeline [1]–[3] can lead to expensive computation and long latency. On the other hand, learning-based pose regression [4]–[8] has improved efficiency but can be cumbersome to deploy due to their low accuracy and long training time. Recently, with aid from neural radiance fields (NeRF) [9], learning-based pose regression methods have greatly improved their accuracy [10], [11]. Direct feature-matching with online-rendered images and synthetic training data generation are two ways people use NeRF to improve pose regression. Despite that, these efforts either need online rendering with NeRF or long-time synthetic data preparation.

To address these limitations, we propose to use NeRF to render a large number of low-resolution images and view the problem as finding a mapping between the distributions of camera poses and images with normalizing flows. NeRF enabled us to conveniently sample in the image space and fully utilize the 3D spatial information embedded in the training dataset. During the evaluation, we can find the full posterior distribution of poses given the images by sampling the latent space of the INN. We summarize our contributions as the following:

1) We extend Local_INN [12] from LiDAR to cameras, which expands the usability for real robots. The method is tested on common benchmark datasets and the performance is on par with state-of-the-art.
2) We realize a fast data preparation pipeline with NeRF [9], [13], which further lowers the deployment burden.
3) We demonstrate the balance of performance and efficiency of the proposed method by deploying it on a real mobile robot.

All authors are with the University of Pennsylvania, Department of Electrical and Systems Engineering, 19104, Philadelphia, PA, USA. Emails: {zzang, aminea, rahulm}@seas.upenn.edu

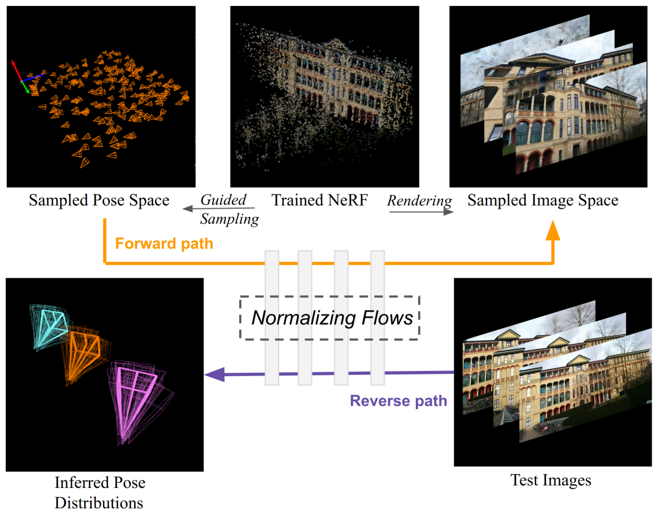


Fig. 1. We propose to learn a mapping between the latent space of the images and camera poses in an environment with an invertible neural network. We use NeRF to guide camera pose sampling and render synthetic images. Evaluating the reverse path of the INN outputs the full posterior distribution of camera poses given a test image.

II. Related Work

A. Visual Pose Regression

The pioneering work in pose regression by PoseNet [4] used simple CNN + average pooling layers to regress camera poses. Since that, the state-of-the-art (SOTA) was yearly refreshed by people trying more complex neural network architectures, such as using separate outputs for position and orientation [5], translational invariant layers [6], [8], or LSTM [14], auto-encoders [15], transformers [16], etc. To show the effectiveness of our method, we are using an encoder that is also simply CNN + average pooling and yet still performs on par with the SOTAs.

Recently, pose regression tasks benefited from NeRF's ability to render photo-realistic images from novel camera poses. LENS [10] augments the training data by rendering synthetic images with a trained NeRF-W from a grid-based novel pose sampling. The limitation of LENS is the days-long training time and high-resolution image rendering time. On the other hand, Direct-PoseNet [17] uses a photometric loss to compare the test images with NeRF-rendered images at test poses. DFNet [11] improved that with direct feature-matching in the feature space instead of pixel-value space. However, these methods require expensive online rendering from NeRF. Different from the SOTA's complex approach, we claim that offline rendering of many low-resolution images is enough to perform the localization. Given a test image, our method also produces the posterior distribution of camera poses, which can be used as uncertainty estimations [6], [18], [19] to improve robustness and deployability.

B. Normalizing Flows

Normalizing flows use a series of bijective transformations to map a source distribution to a target distribution. They provide efficient density estimation [20], [21] and sampling of the target distribution. Ardizonne et al. [22]–[24] proposed a framework for using normalizing flows to solve ambiguous inverse problems. The use of INNs in solving inverse problems has been applied to various fields [22], [25]–[27]. Recently, Local_INN [12] has shown the effectiveness of INNs in performing robot localization, which is naturally an ambiguous inverse problem. However, [12] uses LiDAR ranges, which can be simulated with high fidelity given an occupancy map of the environment. Although LiDAR data provide reliable distance measurements, the sensor is expensive and lacks color information about the world. We extend that framework for visual 6DoF pose regression which is a more common problem. For that we developed a synthetic pose sampling policy with NeRF guidance.

III. Method

Our approach to visual pose regression is to view it as finding a mapping from the distribution of the image to that of camera poses. Effectively sampling enough corresponding data points in both distributions is the key to finding such mapping. We train a Neural Radiance Fields (NeRF) model of the environment and use it to render images at randomly sampled novel camera poses as in Fig. 2. We propose a random camera pose sampling and synthetic image rendering pipeline that is fundamental to the final pose regression result.

Once we have generated enough image samples, based on [12], we use normalizing flows combined with variational autoencoder (VAE) to learn the mapping from pose to images. To reduce the dimensionality of image data, we use a VAE to encode images into a latent space. Then, we use coupling-based normalizing flows to learn the mapping from encoded images to poses. The latent space of the normalizing flows is sampled according to a normal distribution during training. During the evaluation, we evaluate only the encoder of the VAE and the reverse path of the normalizing flows with repeatedly sampled INN latent space to reveal the full posterior distribution of the poses given an input image.

A. Generate Synthetic Views with NeRF

A NeRF model stores 3D spatial information of an environment implicitly within two neural networks: A density MLP and a color MLP, which can be queried for any point in the continuous 3D space. Images can be rendered by tracing rays from the environment to the image plane, integrating the density and color information provided by the two MLPs. We train a NeRF model with a set of images with known camera poses, optimizing the rendering loss. However, if the learned density and color information is noisy or missing, the rendered images will contain artifacts or be a complete mess. Therefore, selecting suitable rendering poses while sufficiently sampling the wanted 3D space is challenging.

We used nerfacto [13] as our NeRF model. After training the model, we output a sparse point cloud from NeRF by thresholding the density of the environment. To generate novel camera poses, we first uniformly randomly sample positions in the region. The orientations of these sampled camera poses are given as $R_{\text{noise}}R^{\text{rand}}_{\text{training}}$, where $R^{\text{rand}}_{\text{training}}$ is a randomly picked camera orientation from the training set and R_{noise} is an added perturbation. We generated random rotation R_{noise} for up to 3.6 degrees using [28].

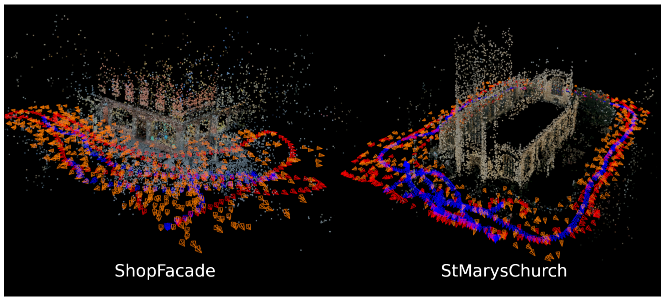


Fig. 2. Sampling of Novel Camera Poses. Point clouds represent high-density points in the environment. Small pyramids represent training poses, testing poses, and sampled poses.

For each sampled camera pose, we verify that we have sufficient spatial information by finding a subset $\mathcal{P}_{\text{in-view}}$ of the NeRF point cloud that is within the field of view (in-view) of the sample camera. We want every sampled camera pose to have enough $\mathcal{P}_{\text{in-view}}$, i.e. enough density information for rendering, and not blocked by a very close point in $\mathcal{P}_{\text{in-view}}$. We then filter out the sampled camera poses according to the following three rules:

- The distance δ_{training} from the sampled pose to the nearest pose in the training set cannot be larger than 0.5 meters.
- For $N_{\text{in-view}} = |\mathcal{P}_{\text{in-view}}|$, we first find the range of $N_{\text{in-view}}$ of the poses in training set. Then we limit the $N_{\text{in-view}}$ of sampled poses according to that range.
- The distance $\delta_{\text{in-view}}$ from the sampled pose to the nearest point in $\mathcal{P}_{\text{in-view}}$ is also limited with the range of $\delta_{\text{in-view}}$ of the poses in training set.

Because we use a sparse point cloud, we can sample 50k poses within minutes. Synthetic images at the sampled poses are then rendered with the trained NeRF model.

B. Learning the Pose-Image Mapping

Normalizing flows are a series of transformations that are mathematically invertible and with learnable parameters. Fig. 3, shows the structure of the network. The normalizing flows side of the network is identical to [12], please refer to that paper for details. We use Real-NVP [20], [21] for its efficiency, which uses affine coupling blocks to achieve invertibility. $\mathbf{c}$ is the optional conditional input [23]. For a fair comparison with other methods, we don't use $\mathbf{c}$ for the absolute pose regression experiments. It's only for real robot localization experiments. Normalizing flows require the input and output to have the same dimension due to their invertibility. The 6DoF camera poses, $\mathbf{x} = [x, y, z, \theta_z, \theta_x, \theta_y]$

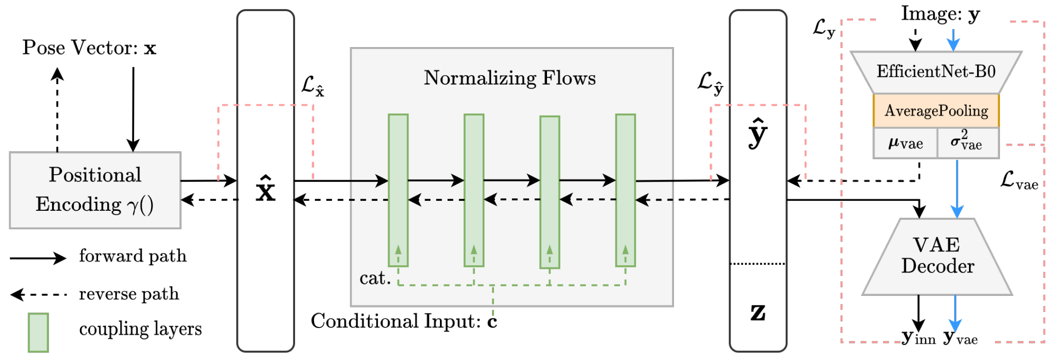


Fig. 3. Network Structure of the PoseINN. The forward path (solid) is from pose to image. The reverse path (dashed) is from image to pose.

are augmented with Positional Encoding [29] [9] from $\mathbb{R}^6$ to $\mathbb{R}^{12L}$.

$$\gamma(p) = (\sin(2^0\pi p), \cos(2^0\pi p), \dots, \sin(2^{L-1}\pi p), \cos(2^{L-1}\pi p)). \tag{1}$$

We use $L = 5$ for camera poses and the output is concatenated with the original 6-dimensional pose to form an input $\hat{\mathbf{x}} \in \mathbb{R}^{12L+6}$ for the INN. On the image side, we use a VAE to encode the image $\mathbf{y}$ into $\hat{\mathbf{y}} \in \mathbb{R}^{12L}$, which is concatenated with a 6-dimensional latent vector $\mathbf{z} \sim \mathcal{N}(0, 1)$ to form the output of the INN. Different from [12], in the VAE encoder, we use a pre-trained EfficientNet-B0 backbone [30] connected with an average-pooling layer to output one number for each feature channel. At test time, we can sample the latent vector to reveal the full posterior distribution of the pose given an image [23].

We train the network the same way as in [12], where with each batch of data, we evaluate both the forward and reverse paths of the network and losses are added together before an optimizer step. To handle the 6DoF poses more efficiently, we used the geodesic distance [31] $\mathcal{L}_{\text{geo}}$ between two rotations:

$$\mathcal{L}_{\text{geo}} = \cos^{-1}((tr(M_{\text{pred}}M_{\text{gt}}^{-1}) - 1)/2). \tag{2}$$

The EfficientNet backbone in the VAE is loaded with pre-trained weights when initialized and also optimized with the rest of the network in training.

IV. Experiments

We validated our method with two types of tasks. To directly compare it with other pose regression methods, we tested on public absolute pose regression datasets. We also deployed a sequential version on a mobile robot to show the performance of our method on an embedded platform.

A. Camera Pose Regression on Public Dataset

TABLE I
Data Generation Strategy Comparison
(Error Data from 7Scene)

Model (backbone)	Pose Error (m/°)	Synthetic Resolution	Rendering Cost	Generation Mode
LENS(EB3)	0.08/3.00	High	Expensive	Offline
DFNet(EB0)	**0.08**/3.47	Low	Cheap	Online
Ours(EB0)	0.09/**2.65**	**Low**	**Cheap**	**Offline**

With the 7scene [32] dataset, we trained the nerfacto model for 50k epochs, which takes about 20 mins on our setup with an NVIDIA A6000 GPU. Then 50k synthetic camera-pose images are rendered for each scene, which takes about 40 mins. The rendering resolution for the 7scene dataset is 160x120. The original training set images are then mixed with the rendered images and resized to 128x128 for training the INN. We trained the network for 300 epochs with batch size 200 and a learning rate of 5e-4 exponentially decaying to 5e-5, which takes around 8 hours. Table I shows a comparison of the data generation strategy with LENS [10] and DFNet [11]. The inputs for the other two methods are from [11]. Our strategy is the most efficient while outputting on-par results.

B. Visual Localization on Real-world Mobile Robot

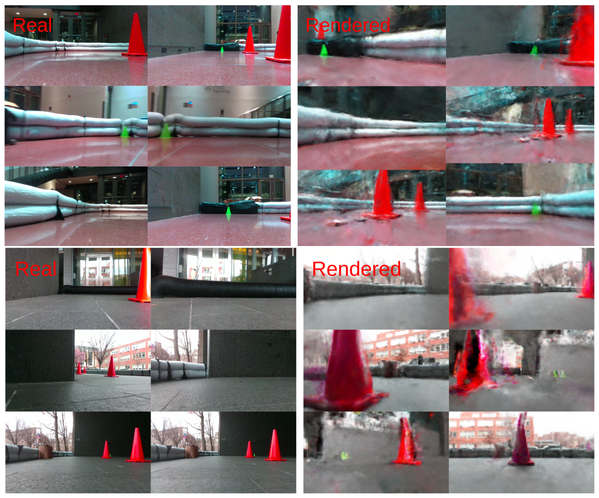
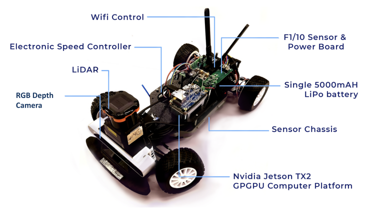
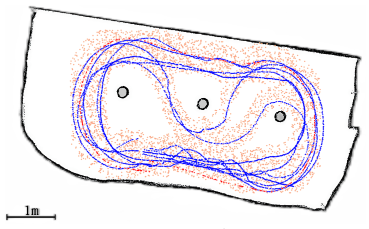
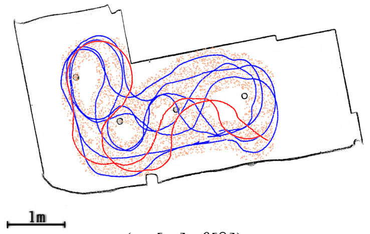


Fig. 4. Examples of training and rendered images in real-world testing (Up: Indoor, Down: Outdoor)

With a small network size, PoseINN is suitable for embedded platforms. To demonstrate that, we deployed PoseINN on an F1TENTH racecar [33], which is a 1/10 scale autonomous racing car equipped with a Hokuyo 30LX LiDAR, an RGB camera, and an NVIDIA Jetson Xavier NX. We used LiDAR to collect ground truth poses for training images and used the camera for localization tests. For the 2D localization experiment, we train for 3 degrees of freedom: xy positions and the car's heading. Similar to [12], the network architecture

TABLE II
MEDIAN LOCALIZATION ERRORS WITH 2D LIDAR VS. CAMERA

Experiment Platform		Indoor	Outdoor
 Wifi Control Electronic Speed Controller LiDAR RGB Depth Camera F1/10 Sensor & Power Board Single 5000mAH LiPo battery Sensor Chassis Nvidia Jetson TX2 GPGPU Computer Platform	train trajectory test trajectory sampled points	 (xy[m], θ[°])	 (xy[m], θ[°])
	Online PF (45Hz)	$\mathbf{0.01}, 0.23$	$\mathbf{0.02}, \mathbf{0.36}$
	PoseINN (**154Hz**)	$0.02, 0.31$	$0.12, 0.72$
	PoseINN + EKF	$0.02, \mathbf{0.22}$	$0.10, 0.65$

we used for the 2D localization experiments takes a rounded previous state of the mobile robot as conditional input $\mathbf{c}$, which is encoded by a separate MLP. This one-step historical information makes the inverse problem easier and it's used in traditional robot localization methods like particle filters [34], [35].

We set up an indoor and an outdoor experiment. The maps shown in the Table II are captured with LiDAR scan using ROS SLAM toolbox. We use an offline particle filter [36] with an infinite computation budget for ground truth poses and training data for NeRF. An online version of the particle filter with fewer particles is used as the baseline comparison. Training and testing trajectories are also shown on the map. We capture RGB images as the car navigates along the trajectories. Fig. 4 shows the training and rendered images. We can see even without the super-accurate image renderings, the trained model is still able to provide localization.

The translation and rotation error results in Table II show that when the training data sufficiently cover the test trajectory, this method can provide localization comparable to LiDAR-based PF. When the test trajectory moves outside the sampled zone, then the performance drops. As for runtime on the Jetson Xavier NX, PoseINN runs at 154Hz while evaluating batches with 50 randomly sampled $\mathbf{z}$ for uncertainty estimation, whereas the compared online particle filter runs at 45Hz.

C. Uncertainty Estimation

TABLE III
OUTPUT FILTERING WITH UNCERTAINTY ESTIMATION
(ERROR DATA FROM CAMBRIDGE [4])

(xy[m], θ[°])	Raw Mean Error	With Filtering
Kings	$0.93, 1.02$	**0.58**, **0.96**
Hospital	$0.87, 1.14$	**0.64**, **0.97**
Shop	$0.59, 5.00$	**0.20**, **1.04**
Church	$0.81, 2.43$	**0.52**, **1.21**
Average	$0.84, 2.55$	**0.68**, **1.87**

We can then calculate the variance of the output distribution as uncertainty estimations. To demonstrate the effectiveness, we use the covariance of the 2D localization results with an Extended Kalman Filter (EKF) to fuse the output with odometry data from the mobile robot, which improves the accuracy. For the 3D pose regression experiments, we show that filtering the inferred poses with their variance reduces noise levels. In Table III, we show the *average error* of the raw outputs of PoseINN on the left. We then filter out outputs with variance values larger than the median variance value of the testing set. The average errors of outputs after filtering are in the right column. Because average values can be influenced by extreme values, a large improvement shows the output is more robust.

V. DISCUSSION & LIMITATIONS

Using NeRF to efficiently sample camera poses and RGB images in an environment, we reduce the problem of pose regression into learning a mapping between two distributions. Results in Table I show that with a large amount of lower-resolution rendering, we can achieve the same performance without using more complex methods or higher resolutions as in the compared methods. Results in Table II show the proposed method is very efficient and can provide accurate localization if proper training data is provided. The uncertainty estimation that naturally comes with the normalizing flows also makes it suitable for deployment on robot platforms.

Some limitations remain in this work. First, we didn't deal with the domain gap between NeRF-rendered images and the real images that change dramatically with the weather, camera parameters, etc. We tried to have the VAE reconstruct rendered images from real images, but the effect was not prominent. Second, we can see from our experiment that better-covered training data is crucial for the final results. Although our camera pose sampling pipeline, reduced instances of bad renderings, having a more deeply related rendering pipeline will be very helpful.

VI. CONCLUSION

We showcase how this [12] invertible neural network architecture can be used for image-based localization at SOTA performance by only changing an image encoder. To achieve that, we used NeRF as a camera simulator to efficiently sample images within an environment. The efficiency and robustness of the model are illustrated by deploying it on an embedded mobile robot.

References

[1] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4104–4113.

[2] A. Fisher, R. Cannizzaro, M. Cochrane, C. Nagahawatte, and J. L. Palmer, "Colmap: A memory-efficient occupancy grid mapping framework," *Robotics and Autonomous Systems*, vol. 142, p. 103755, 2021.

[3] T. Sattler, B. Leibe, and L. Kobbelt, "Improving image-based localization by active correspondence search," in *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part I 12*. Springer, 2012, pp. 752–765.

[4] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2938–2946.

[5] J. Wu, L. Ma, and X. Hu, "Delving deeper into convolutional neural networks for camera relocalization," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5644–5651.

[6] A. Moreau, N. Piasco, D. Tsishkou, B. Stanciulescu, and A. de La Fortelle, "Coordinet: uncertainty-aware pose regressor for reliable vehicle localization," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022, pp. 2229–2238.

[7] I. Melekhov, J. Ylioinas, J. Kannala, and E. Rahtu, "Image-based localization using hourglass networks," in *Proceedings of the IEEE international conference on computer vision workshops*, 2017, pp. 879–886.

[8] R. Liu, J. Lehman, P. Molino, F. Petroski Such, E. Frank, A. Sergeev, and J. Yosinski, "An intriguing failing of convolutional neural networks and the coordconv solution," *Advances in neural information processing systems*, vol. 31, 2018.

[9] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[10] A. Moreau, N. Piasco, D. Tsishkou, B. Stanciulescu, and A. de La Fortelle, "Lens: Localization enhanced by nerf synthesis," in *Conference on Robot Learning*. PMLR, 2022, pp. 1347–1356.

[11] S. Chen, X. Li, Z. Wang, and V. A. Prisacariu, "Dfnet: Enhance absolute pose regression with direct feature matching," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part X*. Springer, 2022, pp. 1–17.

[12] Z. Zang, H. Zheng, J. Betz, and R. Mangharam, "Local_inn: Implicit map representation and localization with invertible neural networks," *arXiv preprint arXiv:2209.11925*, 2022.

[13] M. Tancik, E. Weber, E. Ng, R. Li, B. Yi, J. Kerr, T. Wang, A. Kristoffersen, J. Austin, K. Salahi, A. Ahuja, D. McAllister, and A. Kanazawa, "Nerfstudio: A modular framework for neural radiance field development," *arXiv preprint arXiv:2302.04264*, 2023.

[14] B. Wang, C. Chen, C. X. Lu, P. Zhao, N. Trigoni, and A. Markham, "Atloc: Attention guided camera localization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 06, 2020, pp. 10 393–10 401.

[15] Y. Shavit and Y. Keller, "Camera pose auto-encoders for improving pose regression," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part X*. Springer, 2022, pp. 140–157.

[16] Y. Shavit, R. Ferens, and Y. Keller, "Learning multi-scene absolute pose regression with transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 2733–2742.

[17] S. Chen, Z. Wang, and V. Prisacariu, "Direct-posenet: absolute pose regression with photometric consistency," in *2021 International Conference on 3D Vision (3DV)*. IEEE, 2021, pp. 1175–1185.

[18] A. Kendall and R. Cipolla, "Modelling uncertainty in deep learning for camera relocalization," in *2016 IEEE international conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4762–4769.

[19] F. Zangeneh, L. Bruns, A. Dekel, A. Pieropan, and P. Jensfelt, "A probabilistic framework for visual localization in ambiguous scenes," *arXiv preprint arXiv:2301.02086*, 2023.

[20] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," *arXiv preprint arXiv:1605.08803*, 2016.

[21] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," *Advances in neural information processing systems*, vol. 31, 2018.

[22] L. Ardizzone, J. Kruse, S. Wirkert, D. Rahner, E. W. Pellegrini, R. S. Klessen, L. Maier-Hein, C. Rother, and U. Köthe, "Analyzing inverse problems with invertible neural networks," *arXiv preprint arXiv:1808.04730*, 2018.

[23] L. Ardizzone, C. Lüth, J. Kruse, C. Rother, and U. Köthe, "Guided image generation with conditional invertible neural networks," *arXiv preprint arXiv:1907.02392*, 2019.

[24] C. Winkler, D. Worrall, E. Hoogeboom, and M. Welling, "Learning likelihoods with conditional normalizing flows," *arXiv preprint arXiv:1912.00042*, 2019.

[25] T. J. Adler, L. Ardizzone, A. Vemuri, L. Ayala, J. Gröhl, T. Kirchner, S. Wirkert, J. Kruse, C. Rother, U. Köthe *et al.*, "Uncertainty-aware performance assessment of optical imaging modalities with invertible neural networks," *International journal of computer assisted radiology and surgery*, vol. 14, no. 6, pp. 997–1007, 2019.

[26] T. Wehrbein, M. Rudolph, B. Rosenhahn, and B. Wandt, "Probabilistic monocular 3d human pose estimation with normalizing flows," in *International Conference on Computer Vision (ICCV)*, Oct. 2021.

[27] R. Zhao, T. Liu, J. Xiao, D. P. Lun, and K.-M. Lam, "Invertible image decolorization," *IEEE Transactions on Image Processing*, vol. 30, pp. 6081–6095, 2021.

[28] J. Arvo, "Fast random rotation matrices," in *Graphics gems III (IBM version)*. Elsevier, 1992, pp. 117–120.

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[30] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

[31] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5745–5753.

[32] B. Glocker, S. Izadi, J. Shotton, and A. Criminisi, "Real-time rgb-d camera relocalization," in *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 2013, pp. 173–179.

[33] M. O'Kelly, H. Zheng, D. Karthik, and R. Mangharam, "F1tenth: An open-source evaluation environment for continuous control and reinforcement learning," in *Proceedings of the NeurIPS 2019 Competition and Demonstration Track*, ser. Proceedings of Machine Learning Research, H. J. Escalante and R. Hadsell, Eds., vol. 123. PMLR, 08–14 Dec 2020, pp. 77–89.

[34] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Proceedings 1999 IEEE international conference on robotics and automation (Cat. No. 99CH36288C)*, vol. 2. IEEE, 1999, pp. 1322–1328.

[35] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, 2002.

[36] C. Walsh and S. Karaman, "Cddt: Fast approximate 2d ray casting for accelerated localization," vol. abs/1705.01167, 2017. [Online]. Available: http://arxiv.org/abs/1705.01167