Bayesian Information Recovery from CNN for Probabilistic Inference

Dmitry Kopitkov and Vadim Indelman

***Abstract*— Typical inference approaches that work with high-dimensional visual measurements use hand-engineered image features (e.g. SIFT) that require combinatorial data association, or predict only hidden state mean without considering its uncertainty and multi-modality aspects. We develop a novel approach to infer system hidden state from visual observations via CNN features which are outputs of a CNN classifier. To that end, at pre-deployment stage we use neural networks to learn a generative viewpoint-dependent model of CNN features given the robot pose and approximate this model by a spatially-varying Gaussian distribution. Further, at deployment this model is utilized within a Bayesian framework for probabilistic inference, considering a robot localization problem. Our method does not involve data association and provides uncertainty covariance of the final estimation. Moreover, we show empirically that the CNN feature likelihood is unimodal which simplifies the inference task. We test our method in a simulated Unreal Engine environment, where we succeed to retrieve high-level state information from CNN features and produce trajectory estimation with high accuracy. Additionally, we analyze robustness of our approach to different light conditions.**

I. Introduction

Inferring a system state from multiple measurements, possibly captured by different sensors, is a fundamental problem in robotics. Bayesian inference for system identification is one of the main building blocks on which modern real-world robotic applications rely, such as autonomous navigation and simultaneous localization and mapping (SLAM). The inference task is challenging considering the stochastic nature of the captured measurements. Moreover, retrieving state information from measurements becomes even harder when the measurements are high-dimensional (e.g. images).

A common approach to deal with images is by producing from them visual features - points in a picture that have very recognizable/special local appearance (e.g. SIFT). Further, such landmarks are tracked along the trajectory and help to localize the robot pose via triangulation and multi-view geometry. However, such geometric methods have significant weaknesses. The landmarks are not always robustly detected w.r.t. changing environment conditions, such as weather, day-night changes, spatial occlusions and moving objects. Further, landmark-based techniques require determining data association, i.e. matching different landmarks in different images. This process is tough and prone to mistakes that introduce measurement outliers into estimation and damage the inference accuracy.

D. Kopitkov is with the Technion Autonomous Systems Program (TASP), Technion - Israel Institute of Technology, Haifa 32000, Israel, dimkak@technion.ac.il. V. Indelman is with the Department of Aerospace Engineering, Technion - Israel Institute of Technology, Haifa 32000, Israel, vadim.indelman@technion.ac.il.

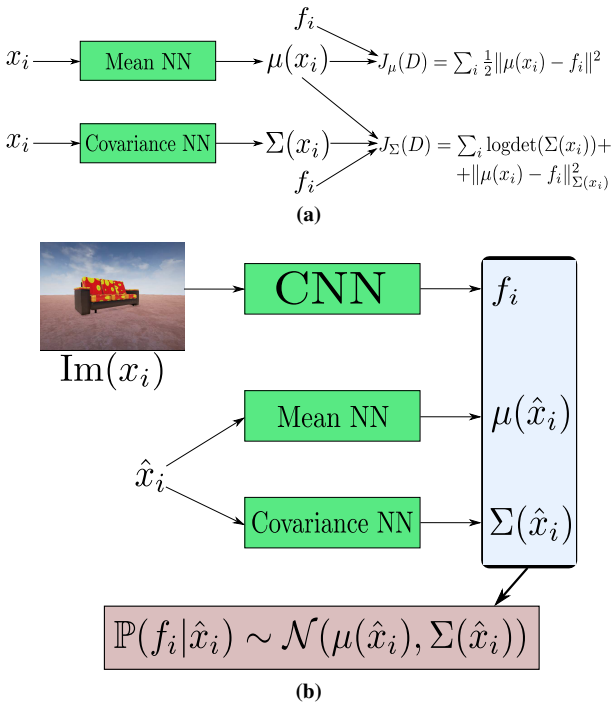


Fig. 1: Approach overview. In this paper we use CNN features for robot's state inference within a Bayesian framework. An image captured from robot pose x_i is passed to a CNN classifier which produces a features vector f_i that represents the image. (a) During the pre-deployment stage we learn *spatially-varying* CNN probability likelihood $\mathbb{P}(f_i|x_i)$ approximated by $\mathcal{N}(\mu(x_i), \Sigma(x_i))$. Two neural networks produce *viewpoint-dependent* mean and covariance functions of f_i given x_i, and are trained through losses J_μ and J_Σ respectively. (b) During the deployment stage the likelihood of f_i is evaluated at x_i's estimation $\hat{x}_i$. Such *spatially-varying* probability likelihood $\mathbb{P}(f_i|\hat{x}_i)$ turns to be very informative during the Bayesian inference optimization where we estimate robot's entire trajectory.

In this paper we present an approach that instead uses short descriptors of image high-dimensional measurements produced by a convolutional neural network (CNN), the CNN feature vectors. We will show that by using these descriptors as the measurements we can recover essential information for the state estimation task.

In recent years the usage of convolutional neural networks (CNN) in computer vision and robotics became very popular. CNN is applied for both passive and active autonomous tasks, such as occlusion detection, place recognition, visual odometry and camera localization [5], [8], [9], [11], [13], [17]. The corresponding approaches typically use the output from different layers of a CNN, also known as the CNN feature vectors, as noisy measurements and utilize these to regress the required hidden state of the system (e.g. camera pose). However, in many CNN-based works [5], [11] only the mean of state estimation is found and the estimation uncertainty is ignored. Accounting for such uncertainty is of prime importance in robotics applications, as ignoring it can cause severe operation malfunction. Only recently, deep

machine learning approaches started to focus on estimation error uncertainty as well [8], [17]. Moreover, as we discuss in this paper, in many cases the distribution of the hidden state (e.g. robot pose) given the image measurement is multimodal due to perceptual aliasing. If this is the case, the estimation mean may not adequately represent the real hidden state of the system.

In contrast, in this paper we present an approach that learns the opposite distribution - of a CNN feature vector given camera pose. We approximate this distribution as a Gaussian with mean vector and covariance matrix that are functions of the camera pose, i.e. viewpoint-dependent. During pre-deployment we learn these functions using two neural networks, and produce a viewpoint-dependent measurement model of a CNN feature vector. At the deployment stage we use this measurement model to infer state information from CNN feature vectors produced from images that were acquired along the robot trajectory (see Figure 1). Specifically, we perform robot localization by utilizing the fact that CNN feature vectors are viewpoint-dependent (see Figure 3). For this purpose, we express the trajectory inference problem via Bayesian formulation where both mean and covariance of CNN feature vectors are exploited. Moreover, we speculate in this paper that the feature distribution conditioned on pose is unimodal since given specific pose the feature values are typically oscillating around one single value, as we will see further. Learning such a unimodal distribution and using it for state inference is a significantly simpler task than learning a distribution with several modes and using it for state estimation. This is an additional motivation for the research direction of this paper.

To summarize, our main contributions in this paper are as follows: (a) we learn a spatial measurement likelihood of CNN features; (b) we exploit the viewpoint-dependency of CNN features for camera pose estimation; (c) we solve Bayesian inference with spatially varying measurement likelihood, in particular using tensorflow/computation graphs; (d) for inference we use image measurements without data association requirement; and (e) we analyze robustness of CNN features to different light conditions via night scenario simulation.

II. Related Work

In many CNN-based techniques a neural network provides an image descriptor that contains high-level semantic information and is used to infer the hidden system state [11], [13]–[15], [20]. Specifically, solving camera pose estimation and localization problems by using CNN descriptors got a big attention in the last years. Particularly, the PoseNet network [11] was introduced to regress a 6-DOF camera pose from a CNN feature vector given an image. However, such a regression was producing only the mean of the pose conditioned on CNN features, since the CNN feature vector has an innate sensor noise and the connection between camera pose and CNN feature vector is stochastic in nature.

In particular, there are two main reasons that cause the regressed pose to be stochastic. First is the mentioned above image noise affected for example by specific light conditions and weather. The second is the learning process itself, where a specific chosen CNN architecture, learning hyper-parameters and training dataset affect the final regression outputs of PoseNet. The error uncertainty of pose estimation is affected by these two stochastic sources, and accounting for such uncertainty is typically required in order to perform robust estimation, e.g. using Bayesian fusion.

In later works [7], [8] the authors deal with the second source mentioned above, by performing dropout during regression, i.e. at test time, and averaging over several regression samples. Further, in [10] both uncertainty sources were termed as *aleatoric* (measurement noise) and *epistemic* (model noise), and the first one was dealt through measurement heteroscedastic covariance regression in an image segmentation problem. Later, in [3] the approach was extended to regress an entire trajectory from video clips. To deal with multi-modality of pose estimation the authors infer camera pose distribution through Gaussian mixture approximation. In contrast, in our work we explore the opposite distribution of CNN feature vector given pose which we argue to be unimodal and simpler to learn.

In another related work [13], Li et al. convert CNN features into a low-dimensional space by locally-inverse transformations using pose-feature pairs from past experiments. Further, the measurement model of the new features is approximated via data perturbation and used in Bayesian inference to estimate robot trajectory. Yet, such a technique requires past measurements and their localized poses from the entire environment which is typically problematic to acquire in real-world applications. In contrast, in this paper we first learn the measurement model of CNN features and at the deployment stage use this model without any need for old observations.

Lately, uncertainty regression through neural networks became more popular [3], [10], [17], though the idea is not new. In 1996, Williams at el. [18] introduced an efficient representation to parametrize a positive-definite covariance matrix of a Gaussian distribution through a neural network. In our work we use this representation to regress the heteroscedastic covariance matrix of a CNN feature vector.

III. Problem Formulation

We consider a Bayesian inference framework and use a typical smoothing formulation, where the entire robot trajectory is inferred. Denote the robot pose at time step i as x_i, and combine all such positions till current time step k as state vector $X_k = \{x_0, \dots, x_k\}$. Also, consider factors $F = \{f^1(X^1), \dots, f^{n_f}(X^{n_f})\}$ added to the inference system till current time. Each such factor $f^j(X^j)$ represents a measurement model, prior or odometry, with $X^j \subset X_k$ being a subset of the involved state variables.

The probability density function (pdf) of X_k can be represented as

$$\mathbb{P}(X_k|history) \propto \prod_{j=1}^{n_f} f^j(X^j), \tag{1}$$

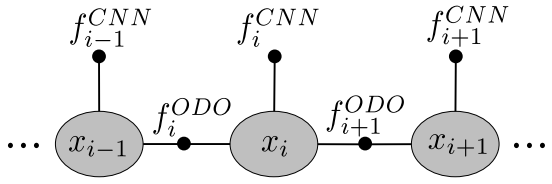


Fig. 2: Illustration of a factor graph that represents this paper's trajectory inference problem. It contains odometry factors f_i^{ODO} for each pose pair $\{x_{i-1}, x_i\}$ and CNN factors f_i^{CNN} for each pose x_i.

where *history* represents all information gathered till current time.

Typically in the robotics domain, factors are represented as Gaussian functions:

$$f^j(X^j) \propto \det(2\pi\Sigma^j)^{-\frac{1}{2}} \cdot \exp(-\frac{1}{2}\|h^j(X^j, z^j)\|^2_{\Sigma^j}), \quad (2)$$

with an appropriate model

$$v^j = h^j(X^j, z^j), \quad v^j \sim \mathcal{N}(0, \Sigma^j), \quad (3)$$

where h^j is a known nonlinear function with state subset $X^j \subseteq X$ and measurement z^j as parameters, and v^j is a Gaussian white noise with covariance Σ^j.

The maximum a posteriori (MAP) estimation of trajectory X_k can be calculated through the optimization:

$$X_k^* = \arg\min_{X_k} \sum_{j=1}^{n_f} \text{logdet}(\Sigma^j) + \sum_{j=1}^{n_f} \|h^j(X^j, z^j)\|^2_{\Sigma^j}. \quad (4)$$

In this paper we infer robot trajectory within known environment considering odometry and camera measurements. Specifically, we will use odometry factors involving two subsequent poses each ($X^j \equiv \{x_{i-1}, x_i\}$). Further, typically in SLAM and localization approaches, human engineered visual features (e.g. SIFT features) are recovered from the camera images and used to provide better trajectory estimation. In contrast, in our work we will use CNN features - outputs from a CNN classifier that receives images as input. In a pre-deployment stage we learn a viewpoint-dependent measurement model of these features and use this model to formulate CNN factors for trajectory inference; each CNN factor will involve a single robot pose (see Figure 2). These CNN factors are one of the paper's main contributions and we will show in Section V how these factors can improve accuracy of the trajectory inference.

State of the art SLAM and localization approaches typically assume the covariance Σ^j of the model noise to be position-independent. In such a case *logdet* terms in Eq. (4) can be ignored. In the same way, the covariance in Mahalanobis terms is typically considered fixed, reducing the problem in Eq. (4) to a weighted least squares optimization, which can be solved efficiently (see e.g. [6]). In contrast, CNN factors have a state-dependent noise. To see that, recall that the measurement used within CNN factors is the output of a CNN classifier given a camera-captured image. Such a measurement is a feature vector taken from one of the layers in CNN, and both its expected value and its variability changes between different areas of the environment (see Figure 4), thus making it position-dependent. Therefore, the noise covariance within measurement model of CNN

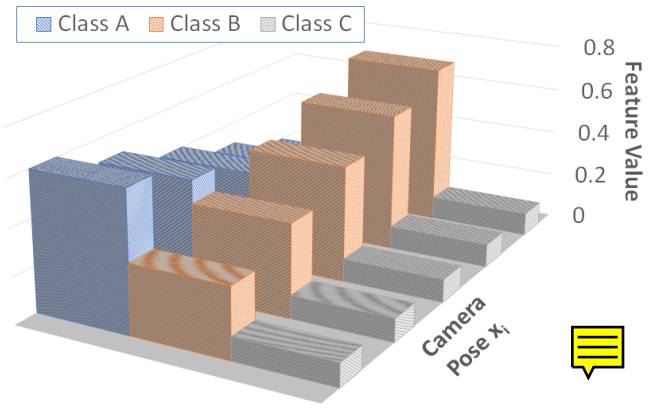


Fig. 3: Illustration of a CNN feature vector f_i changing along camera pose x_i. In this example, a CNN classifier provides $f_i = \{p_A, p_B, p_C\}$ for each pose x_i, with f_i representing probabilities of seeing objects of Class A, B and C. Learning how f_i behaves in our environment, we can use the classifier output in real-time to improve accuracy of robot localization. See also Figure 4 for actual CNN features in UE environment.

factors is spatially-varying, $\Sigma^j(x_i)$, making the optimization problem in Eq. (4) more complicated. We discuss different methods to deal with *logdet* terms and the covariance-varying Mahalanobis distance in Section IV.

IV. Approach

In this paper, rather than using hand-engineered features, we exploit high-level/CNN features and their viewpoint dependence, which we learn, and develop a corresponding Bayesian inference formulation. CNN features behave differently in different map areas (see Figures 3 and 4). Learning their behavior as a function of robot pose x allows us to utilize this information later at the deployment stage for better localization estimation. For that purpose, we use pre-trained Inception-v3 [16] that provides a vector of class probabilities f given an image captured at each time step. The weights from the Inception-v3 network are not modified, keeping their pre-trained values. During the pre-deployment stage we learn heteroscedastic viewpoint-dependent measurement model $\mathbb{P}(f|x)$ of this vector f in a simulation environment, and incorporate this learned model within probabilistic inference during a deployment. This results in an improved accuracy of robot localization due to the spatial dependence of CNN features. In such a way we recover high-level state-dependent information from trained CNN networks.

Importantly, however, our localization inference does not rely on the quality of classifier object predictions since f within $\mathbb{P}(f|x)$ represents only classifier outputs and not necessarily the real probabilities to see specific objects. In other words, when a classifier fails to recognize objects in the area and returns incorrect probabilities inside f, the trajectory estimation will still be accurate since the learned likelihood $\mathbb{P}(f|x)$ will also contain information about such classification inconsistency. That is, as long as our learned likelihood captures the viewpoint-dependency of the classifier output it will be informative during the trajectory inference.

CNN features behave differently in different environment areas depending on what objects are observed from each viewpoint. Both the expected value and the variability of the

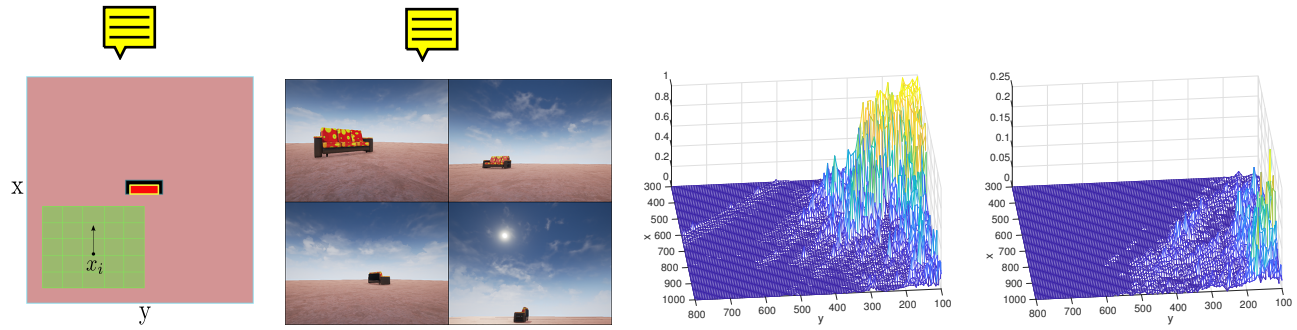


Fig. 4: Environment simulated in this paper. (a)-(b) Map and image examples of the environment simulated through UE. The environment contains a single object of a sofa at the center, daily lighting from sun, sky clouds and brown terrain. Note that both floor and sky do not have many edge features, making it hard to use SIFT-like feature-based approaches for visual odometry. (c)-(d) CNN produced probabilities of observing a park bench (b) and a rocking chair (c) from grid-sampled camera pose x_i. The $x_i = \{\text{x, y, yaw, pitch}\}$ is positioned in area marked by green in (a), with constant orientation $\{\text{yaw} = 180°, \text{pitch} = 0°\}$ and with $\{\text{x, y}\}$ representing grid intersections of the green area. Note as the probability to see a park bench (b) and its variability get bigger when camera gets closer to the sofa. In this paper we exploit such viewpoint-dependency of CNN feature distribution to infer robot state (see also Figure 3).

specific feature f vary spatially, making the distribution of the feature to be position-dependent (see Figure 4). Moreover, we speculate that such feature distribution $\mathbb{P}(f|x)$ conditioned on a position x will have only a single mode - given a specific position, the CNN feature (e.g. probability to see a chair in the captured image) will be spread around a specific single value. This can be seen empirically from Figures 4c-4d. In contrast, many existing techniques learn the opposite distribution $\mathbb{P}(x|f)$ of pose given the feature/measurement, or its mean $\mathbb{E}(x|f)$ (e.g. Posenet in [11]). Such a conditional distribution typically will have multiple modes - there can be multiple objects of the same type in the area; thus, the same feature can be produced at different positions. We argue that learning a unimodal distribution is by far an easier task compared to learning a multimodal one, further adding motivation to our research direction.

Below, in Section IV-A we will describe the main intuition behind our approach. In Section IV-B we describe how to learn the CNN measurement model using deep learning (DL) techniques. Further, in Section IV-C we discuss Bayesian inference optimization that involves factors with a spatially-varying noise.

A. General Idea

We use a designated neural network to approximate the CNN measurement model. Specifically, define a sample dataset $D = \{x_i, f_i\}$ where pose x_i is taken uniformly from the scenario's environment and where f_i is a CNN feature vector provided for image captured at pose x_i (see Figure 1). Given such a training dataset we learn a conditional pdf $\mathbb{P}(f_i|x_i)$, representing the probability likelihood of getting a specific feature vector from a specific pose. In this paper we approximate this pdf by a spatially varying Gaussian distribution $\mathcal{N}(\mu(x_i), \Sigma(x_i))$. Given input x_i, our neural network returns a mean vector $\mu(x_i)$ and covariance matrix $\Sigma(x_i)$ of likelihood distribution $f_i|x_i$.

In this paper we simulate our environment using Unreal Engine [2] (UE) which can be viewed as a deterministically controlled world (see Figure 4). Thus, unlike the real-world scenario, an image $I_i = Image(x_i)$ captured at pose x_i is deterministic, i.e. given pose the generated image is always the same. Therefore, the CNN feature $f_i = classifier(I_i)$ will be also deterministic since the classifier weights are fixed. However, due to the complexity of functions $Image(\cdot)$ and $classifier(\cdot)$ the CNN feature f_i can be thought as *pseudo*-stochastic, produced by some distribution $f_i|x_i$ which we want to learn. Such a perspective is supported by Figures 4c and 4d where CNN features seem to have a stochastic nature. Another explanation for looking for $\mu(x_i)$ and $\Sigma(x_i)$, while the feature vector $f_i(x_i)$ is a deterministic function of x_i is as follows. We aim to approximate $f_i(x_i)$ via $\mu(x_i)$. However, since this function is complicated, $\mu(x_i)$ can approximate it only up to a specific resolution, producing an approximation error $err(f_i, x_i) = f_i - \mu(x_i)$. This error would be very different in various areas of x_i since in some areas the function $f_i(\cdot)$ can be easily approximated (e.g. areas where it is zero), while in other areas the approximation would be very rough (e.g. areas where it is spiky). Since one of the requirements of Bayesian inference and least-squares regression (see Section IV-C) is for errors to have a unified magnitude, the $\Sigma(x_i)$ is responsible for canceling the difference between approximation errors. Thus, for a large error $err(f_i, x_i)$, $\Sigma(x_i)$ will be high on average, while for small error values, it will be small.

Note that when applying our method in a real-world scenario, the CNN features f_i would become entirely stochastic since the images are affected by other factors such as light conditions and pedestrians. In this setting learning a conditional distribution $\mathbb{P}(f_i|x_i)$ of a random variable f_i would become even more crucial and beneficial.

In the next section we discuss how DL can be used to learn the required $\mathbb{P}(f_i|x_i)$.

B. Measurement Model Learning via DL

To represent $\mathcal{N}(\mu(x_i), \Sigma(x_i))$ via neural networks we use a Gaussian parametrization introduced in [18]. Specifically, we use two separate networks to learn $\mu(x_i)$ and $\Sigma(x_i)$. Both *mean* and *covariance* networks have a similar architecture and the separation was done for a more stable learning process (see Figure 1).

In this paper we address a 4DOF case scenario where each pose $x_i = \{\text{x, y, yaw, pitch}\}$ has 4 degrees of freedom. Each network gets poses x_i as input. It contains several fully connected (FC) layers (we examined different architectures with varying number of the layers), where the output of the final layer, o_i, is transformed into an appropriate function

as follows. In the *mean* network, o_i has the dimension of $m \equiv |f_i|$ and the learned function is:

$$\mu(x_i) = sigmoid(o_i). \tag{5}$$

In this paper we use class probabilities vector f_i for a specific class subset produced by a CNN classifier. The f_i's entries are bounded by $[0, 1]$ and bounding entries of the function $\mu(x_i)$ to this range through the $sigmoid$ helped stabilize the learning. Note that we do not constrain the sum of the μ's entries to be 1 since we do not use all CNN class probabilities but only a small subset of thereof.

We represent the conditional covariance of f_i through a positive-definite upper-triangular square-root information matrix $R(x_i)$ with $\Sigma(x_i) = (R(x_i)^T \cdot R(x_i))^{-1}$. The dimension of the final layer output o_i of the *covariance* network is $m \cdot (m+1)$, which corresponds to the number of non-zero entries in $R(x_i)$. $R(x_i)$ is constructed as:

$$\begin{aligned} \text{diag}(R(x_i)) &= \exp(o_i(1:m)) \\ \text{utri}(R(x_i)) &= o_i(m+1:end) \\ \text{ltri}(R(x_i)) &= 0 \end{aligned} \tag{6}$$

where $utri(\cdot)$ and $ltri(\cdot)$ represent matrix entries above and below the main diagonal, respectively, and where we ensure the positive-definiteness of $R(x_i)$ by forcing its diagonal entries to be positive.

First, we train the *mean* network with an $L2$ regression loss:

$$\min_{\theta_\mu} J_\mu(D) = \min_{\theta_\mu} \sum_i \frac{1}{2} \|\mu(x_i) - f_i\|^2. \tag{7}$$

After the training converges, we use it for the *covariance* network learning through a maximum likelihood loss:

$$\begin{aligned} &\max_{\theta_\Sigma} \prod_i \mathbb{P}(f_i|x_i) = \min_{\theta_\Sigma} J_\Sigma(D) = \\ &= \min_{\theta_\Sigma} \sum_i \text{logdet}(\Sigma(x_i)) + \|\mu(x_i) - f_i\|^2_{\Sigma(x_i)}, \text{where} \end{aligned} \tag{8}$$

$$\text{logdet}(\Sigma(x_i)) = -2 \cdot \text{logdet}(R(x_i)) = -2 \cdot \sum_{j=1}^{m} o_j, \tag{9}$$

$$\|\mu(x_i) - f_i\|^2_{\Sigma(x_i)} = \|R(x_i) \cdot (\mu(x_i) - f_i)\|^2. \tag{10}$$

Note that the loss J_Σ is optimized only with respect to the *covariance* network weights θ_Σ, since the *mean* network weights θ_μ were already optimized.

The training process considering the loss J_Σ can be complicated by the fact that logarithm terms are in general not bounded from below as opposed to Mahalanobis terms. In fact, the $logdet(\Sigma(x_i))$ has significantly low values w.r.t. other terms, specifically in the areas where variability of CNN features is low. In such areas the term $det(\Sigma(x_i))$ approaches 0^+ and $logdet(\Sigma(x_i))$ goes to $-\infty$. Thereby, the learning process is busy to approximate covariance function in areas where covariance is almost zero, and puts only little effort to approximate it in other areas. To tackle this problem and stabilize the learning, we defined a minimal covariance threshold $\delta_\Sigma = 0.005$ and clipped the $\text{logdet}(R(x_i))$ from above by $-\frac{m}{2}\log(\delta_\Sigma)$.

To better approximate $\mathbb{P}(f_i|x_i)$ we examined different architectures. The architecture of each network was parametrized by the number of FC layers N_F and number of hidden-units within each layer N_H. Additionally, after each FC layer, dropout was performed with probability P_D. Networks with a range of different N_F, N_H and P_D were trained and the best networks were chosen by evaluating the loss on the testing dataset.

Although theoretically $\mu(x_i)$ and $R(x_i)$ can be learned together in a single network, we empirically found that the training separation (first learning mean and then covariance) provides better approximations to the real distribution of the learned CNN feature vector. Nevertheless, it is possible that other regularization methods, besides dropout, can be performed to make the unified neural network a better alternative. This will be explored in our future work.

C. MAP Inference via Spatially-Varying Models

After learning $\mu(x_i)$ and $\Sigma(x_i)$, the CNN factor from Section III can be represented as:

$$\begin{aligned} &f^{CNN}(x_i) \doteq \mathbb{P}(f_i|x_i) = \\ &= \det(2\pi\Sigma(x_i))^{-\frac{1}{2}} \cdot \exp(-\frac{1}{2}\|\mu(x_i) - f_i\|^2_{\Sigma(x_i)}), \end{aligned} \tag{11}$$

where x_i is robot pose at time step i and f_i is the CNN measurement (output of a CNN classifier). Note that due to the spatial variability of $\Sigma(x_i)$, the $det(\cdot)$ term and the weight matrix within the Mahalanobis norm in Eq. (11) are state-dependent, in contrast to the usual setting in SLAM, which typically considers both terms to be constant.

The above factor will introduce two terms into the MAP inference (4):

$$X^*_k = \arg\min_{X_k} \left[\ldots + \text{logdet}(\Sigma(x_i)) + \|\mu(x_i) - f_i\|^2_{\Sigma(x_i)} \right] \tag{12}$$

where both terms can be calculated from neural networks via Eq. (9) and Eq. (10), and i is timestep index. The above Bayesian optimization can be efficiently solved using incremental Gauss-Newton [6] given that it has a least-squares form.

Unfortunately, the first term, *logdet*, is not a square function. One possibility to deal with it is by taking its root and then raising it in the power of two, $(\sqrt{\lambda + \text{logdet}(\Sigma(x_i))})^2$, where the constant λ assures the inner root expression to be positive. Replacing the $\text{logdet}(\Sigma(x_i))$ with such a least-squares term in Eq. (12) will not change the MAP solution X^*_k. However, in our empirical evaluation (see Section V) this squared term did not change much the MAP solution compared to the solution where we use only Mahalanobis terms of Eq. (12). Thus, this suggests that there is not much information about state X_k inside *logdet* terms. Moreover, we optimized Eq. (12) using general optimizers (Gradient Descent, Adam and BFGS) that do not require a least-squares form, and found also there that *logdet* terms do not affect optimization solution in any significant way.

The Mahalanobis distance term can be represented as a simple Euclidean-square norm via Eq. (10). Typical least-squares optimizers (e.g. [4], [6]) require, for each such term, the residual res and the Jacobian A, defined as follows.

The residual of a CNN factor can be calculated via a neural network as:

$$res(x_i, f_i) \doteq R(x_i) \cdot (\mu(x_i) - f_i). \tag{13}$$

The Jacobian A of res w.r.t. x_i can be calculated by exploiting the automatic differentiation capability of modern neural network libraries. Note that in case we ignore the covariance part of Mahalanobis distance during the differentiation and compute the Jacobian as $A = R(x_i) \cdot \frac{\partial}{\partial x_i}(\mu(x_i) - f_i)$, an incorrect Jacobian will be obtained and will negatively affect accuracy of our MAP estimation, as shown in Section V.

V. Results

In this section we evaluate our approach considering a robot localization problem. Specifically, we use CNN features in order to improve estimation accuracy of a robot trajectory. In Section V-A we provide details about the learning process of *mean* and *covariance* networks. Section V-B demonstrates how the information extracted from CNN features influences the trajectory inference. Finally, in Section V-C we show how different light conditions alter the conditional likelihood $\mathbb{P}(f_i|x_i)$ and examine the corresponding effect on our approach performance.

A. Learning Process of $\mu(x_i)$ and $\Sigma(x_i)$ Networks

The networks for $\mu(x_i)$ and $\Sigma(x_i)$ were trained according to the scheme described in Section IV-B using the TensorFlow library [1]. We use Unreal Engine to simulate an environment with a single object (sofa, see Figure 4), daily lighting from sun, sky clouds and brown terrain. We sampled x_i uniformly in this environment and captured an image I_i and CNN output $\hat{f}_i$ for each sample, producing a dataset $D = \{x_i, \hat{f}_i\}$ with 200000 samples where each $\hat{f}_i$ contains 1008 class probability values. Further, this dataset was divided 75%-25% to training and testing sets respectively. We selected $f_i \subset \hat{f}_i$ to include 10 class probabilities for which CNN classifier returned high values on average for samples from D. The chosen classes are: park bench, sandbar, beacon, seashore, wing, patio, forklift, trailer truck, harvester and rocking chair. Note that most of the objects *do not really appear* in our environment and the CNN classifier produces high probabilities for them due to perceptual aliasing. Yet, as was explained in Section IV, as long as f_i within learned $\mathbb{P}(f_i|x_i)$ represents a classifier output (i.e. not necessarily the real probability to see object of a specific class), the learned likelihood can be used within the proposed Bayesian inference framework.

Network architecture parameters N_F, N_H and P_D were sampled from ranges $[2, 8]$, $[100, 1000]$ and $[0.7, 1]$, respectively. In the first stage we trained multiple *mean* networks for different architecture configurations and picked the *optimal* one that produces minimal loss on the testing set. In the second stage we used the *optimal mean* network with fixed weights to train multiple *covariance* networks according to Eq. (8). Finally, we picked 5 best *mean* networks and 5 best *covariance* networks (according to their testing loss) and combined them into 25 mixes, each one representing a different learned likelihood $\{\mathbb{P}^j(f_i|x_i)\}_{j=1}^{25}$.

In Figures 5a-5d we show the output of one of these likelihood models for a specific environment area marked in green in Figure 4a. Specifically, we present mean and standard deviation for two specific CNN feature entries of a park bench and a rocking chair whose real values are displayed in Figures 4c-4d. As can be seen, the learned measurement models resemble the real distribution of the values, though not in the best possible way since the real distribution is complex and hard to approximate. Yet, in the next section we show that each one of the 25 learned sub-optimal likelihoods holds an essential information about the robot state and can improve localization accuracy.

B. Bayesian Inference

In our localization problem the robot moves in a "star"-pattern trajectory (see Figure 6a) where at the star center the sofa object is located. At each time step, the robot receives an odometry measurement and an image from UE, which is converted into CNN feature vector by a CNN classifier. Using appropriate likelihoods for each measurement, the trajectory inference problem can be represented as a factor graph, as illustrated in Figure 2. The inference optimization was solved via GTSAM library [4], [6].

Initially, we solved the problem in Eq. (4) using general gradient-based optimizers. We optimized both versions, with and without *logdet* terms, and saw empirically that these terms have negligible impact on the solution and thus can be ignored. Further, after discarding *logdet* terms and after whitening residuals with state-dependent covariance via Eq. (13), the Eq. (4) becomes ordinary least-squares problem which we optimized using an incremental least-squares optimizer ISAM2 [6]. We evaluate estimation through Absolute Trajectory Error (ATE) performance measure which is the mean of norm of Cartesian error between estimated and ground truth poses (see Figure 6b).

First, we solved the problem using *emulated* CNN measurements f_i. That is, for each ground truth pose x_i we sampled a CNN feature vector f_i from the likelihood $\mathbb{P}(f_i|x_i)$ and used these sampled features in our trajectory inference. Such a setting can be viewed as the scenario where we succeeded to learn a precise likelihood $\mathbb{P}(f_i|x_i)$. As can be seen from Figure 6c (blue line) where all 25 models are considered, in such case the trajectory estimation turns to be very accurate compared to the odometry-only scenario ($\sim$15 ATE vs 283 ATE). This indicates that if we can perfectly learn a CNN measurement model, it becomes very informative for the system inference problem.

Further, we performed the same scenario but with a *real* CNN features that were produced by a CNN classifier from Unreal Engine synthesized images. In Figure 6c (red line) we can see that estimation error is around 50-150 ATE which is still much lower compared to the odometry drift. Yet, it is higher than the samples-emulated scenario, suggesting that the CNN likelihood $\mathbb{P}(f_i|x_i)$ was not perfectly learned. Nonetheless, it is apparently close enough to the real one since it is still very informative in our inference solution.

Additionally, we evaluate inference where Jacobian A

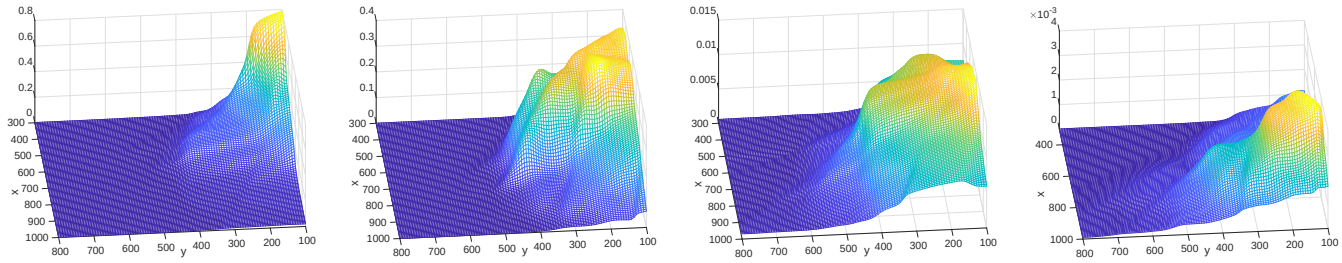
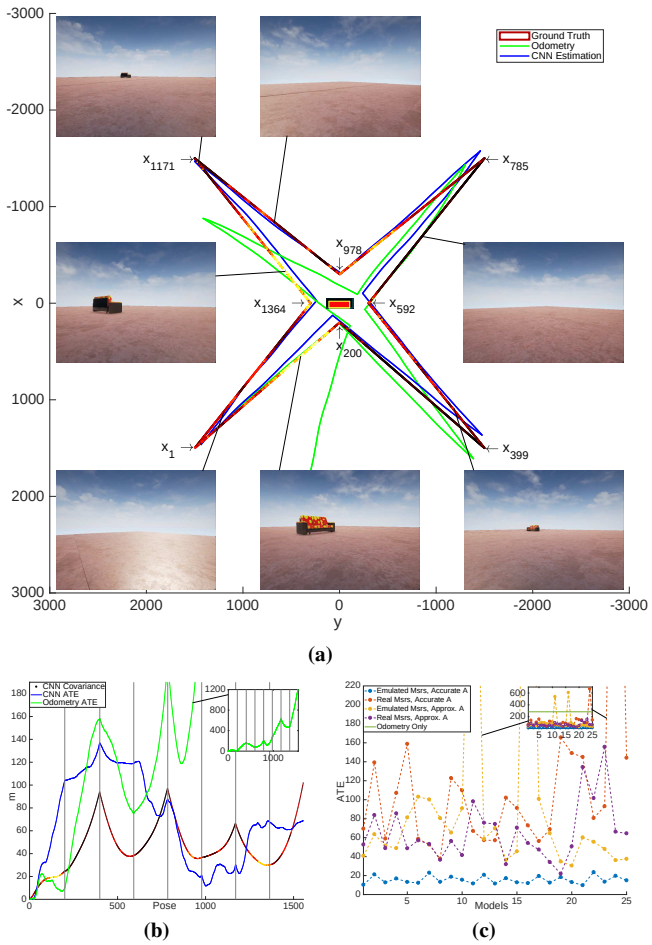


Fig. 5: Learned likelihood $\mathbb{P}^1(f_i|x_i)$ in area marked green in Figure 4a. (a)-(b) are mean and standard deviation of "park bench" feature. (c)-(d) are mean and standard deviation of "rocking chair" feature.



(a)

(b) **(c)**

Fig. 6: Robot trajectory inference using CNN and odometry factors for the first measurement model $\mathbb{P}^1(f_i|x_i)$. (a) Ground truth trajectory colored-coded with information gain of CNN factors, odometry-only estimation (green) and estimation using both odometry and CNN factors (blue). (b) ATE error and uncertainty covariance of CNN estimation and odometry versus pose index. The vertical lines separate 8 sections of star trajectory. (c) Performance ATE error for all considered 25 models for different inference configurations.

from Section IV-C is calculated through $A = R(x_i) \cdot \frac{\partial}{\partial x_i}(\mu(x_i) - f_i)$. Such calculation does not account for the first-order derivative of the covariance function, making the optimizer to progress in a slightly different direction. In samples-emulated setting the new optimization direction results in considerably reduced performance (yellow line in Figure 6c) - around 101 ATE vs around 15 ATE for A's proper calculation. However, in real-measurement setting both approximate and accurate A matrices (purple and red lines in Figure 6c) provide similar performance. This suggests that accurate A can be useful only when our measurement model is accurate. When this is not the case, also roughly-calculated A will provide the same level of performance.

Furthermore, we analyzed in detail the performance of the first model $\mathbb{P}^1(f_i|x_i)$, in a setting with real measurements and accurate A. In Figure 6a, the final (smoothed) trajectory estimation is shown to be very close to the ground truth. Additionally, we colored the ground truth path according to the information gain a CNN factor provides at each robot pose, and we use it to assess the learned model quality (IG-LMQ). It is calculated according to [12] as

$$IG\text{–}LMQ \doteq \frac{1}{2} \ln \left| I_m + A \cdot S \cdot A^T \right|, \qquad (14)$$

where S is a constant scale matrix that accounts for different scales of pose coordinates. We calculated IG-LMQ over ground truth states and real measurements; in such a setting S replaces the prior covariance matrix of camera pose (see [12]). IG-LMQ can be thought as a metric of novelty that a factor/measurement contains. Since we calculate it over ground truth, i.e. the Jacobian A of res w.r.t. x_i (see Eq. (13)) is obtained using the ground truth value for x_i as the linearization point, there should be no novelty and IG-LMQ just reflects how good the model $\mathbb{P}^1(f_i|x_i)$ was learned at a specific pose. That is, high IG-LMQ values indicate that real model of measurements is far from our learned approximation $\mathbb{P}^1(f_i|x_i)$. As can be seen from Figure 6a, the highest IG-LMQ values are in areas where the sofa object is visible and the robot is getting closer to it. One possible reason to explain this is that CNN classifier outputs f_i behave more dynamically (not constant) around the object and learning their behavior via $\mathbb{P}(f_i|x_i)$ is more difficult w.r.t. other areas where object is not observed and where the viewpoint-dependence of CNN features is less significant (f_i's values are lower and more stable w.r.t. x_i). Nevertheless, as seen in Figure 6b, there are significant improvement in estimation accuracy when the object is observed.

In Figure 6b the uncertainty covariance of pose estimation and the ATE error are shown as a function of trajectory pose. As can be seen, pattern of covariance is opposite to IG-LMQ - it gets bigger when robot does not see the sofa object and gets smaller when sofa is captured in images.

Finally, we applied kernels from robust estimation to improve the inference performance. Since we saw that learned likelihoods $\{\mathbb{P}^j(f_i|x_i)\}_{j=1}^{25}$ are not very accurate, some of the

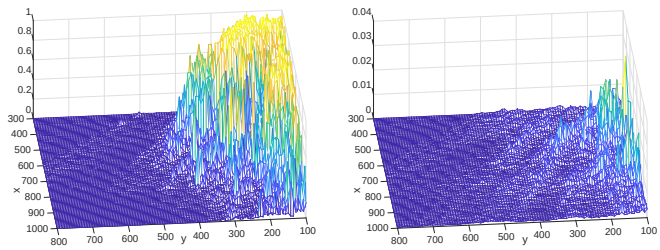


Fig. 7: CNN produced probabilities of observing a park bench (a) and a rocking chair (b) in area marked green in Figure 4a. Used environmental light is considerably darker than in Figures 4c-4d.

measurements can be seen as outliers. Thus we tried different robust kernels [19] which are typically used to deal with measurement outliers. In scenario with real measurements and accurate A we succeeded to reduce average ATE from 117 to 85 via "Fair" and "Cauchy" kernels.

C. Light Conditions Effect

In order to test robustness of our approach to different light conditions, we have constructed a similar "sofa" UE environment but with a significantly darker illumination. In Figure 7 we can see how some CNN features behave in this environment. Compared with the same features in the environment with usual illumination (Figures 4c-4d) we can see that the feature values (and the appropriate distribution $\mathbb{P}(f|x)$) were changed, meaning that the used CNN features f are *not* robust to different light conditions.

The inference task in dark environment using the learned likelihoods $\{\mathbb{P}^j(f_i|x_i)\}_{j=1}^{25}$ produced estimation that was only a little better than odometry drift (226 average ATE vs 283 ATE). The robust kernels did not succeed to improve it. Obviously, the reason for such results is that changing light conditions increased the distance between the real feature distribution and the one approximated through neural networks. When such distance becomes big enough, the approximated likelihood $\mathbb{P}(f_i|x_i)$ is not consistent anymore and cannot be used for inference. In future we will try to robustify the $\mathbb{P}(f_i|x_i)$, e.g. by training on dataset with changing light conditions.

VI. Conclusions

In this paper we developed a Bayesian approach to recover high-level state information from features learned by a CNN classifier and use this information for more accurate state inference. We showed that the CNN features are highly viewpoint-dependent and their generative model can be successfully learned via neural network probability density function (pdf) approximations. Further, we demonstrated how such model can be used to localize robot trajectory as Gaussian factor in Bayesian estimation. Importantly, our approach does not require to solve the challenging data association problem. Moreover, the learned conditional pdf of CNN features' subset given state pose was shown empirically to be a unimodal distribution, unlike the opposite pdf of pose given CNN features which is multimodal due to possible perceptual aliasing. As a consequence, using initial pose values from odometry sensors, the learned unimodal likelihood does not involve dealing with multiple inference hypotheses, unlike the inference process involving multimodal pdfs. In future we will explore ways to improve model accuracy, to use information from the entire CNN feature vector, and to make model more robust to different light conditions.

References

[1] TensorFlow. www.tensorflow.org.

[2] Unreal Engine. www.unrealengine.com.

[3] Ronald Clark, Sen Wang, Andrew Markham, Niki Trigoni, and Hongkai Wen. Vidloc: A deep spatio-temporal model for 6-dof video-clip relocalization. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 3, 2017.

[4] F. Dellaert. Factor graphs and GTSAM: A hands-on introduction. Technical Report GT-RIM-CP&R-2012-002, Georgia Institute of Technology, September 2012.

[5] Sourav Garg, Niko Suenderhauf, and Michael Milford. Don't look back: Robustifying place categorization for viewpoint-and condition-invariant place recognition. *arXiv preprint arXiv:1801.05078*, 2018.

[6] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *Intl. J. of Robotics Research*, 31:217–236, Feb 2012.

[7] Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv:1511.02680*, 2015.

[8] Alex Kendall and Roberto Cipolla. Modelling uncertainty in deep learning for camera relocalization. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 4762–4769. IEEE, 2016.

[9] Alex Kendall and Roberto Cipolla. Geometric loss functions for camera pose regression with deep learning. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 3, page 8, 2017.

[10] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *arXiv preprint arXiv:1703.04977*, 2017.

[11] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: Convolutional networks for real-time 6-dof camera relocalization. In *Intl. Conf. on Computer Vision (ICCV)*, 2015.

[12] D. Kopitkov and V. Indelman. No belief propagation required: Belief space planning in high-dimensional state spaces via factor graphs, matrix determinant lemma and re-use of calculation. *Intl. J. of Robotics Research*, 36(10):1088–1130, August 2017.

[13] Jie Li, Paul Ozog, Jacob Abernethy, Ryan M Eustice, and Matthew Johnson-Roberson. Utilizing high-dimensional features for real-time robotic applications: Reducing the curse of dimensionality for recursive bayesian estimation. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1230–1237. IEEE, 2016.

[14] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*, pages 512–519. IEEE, 2014.

[15] Niko Sunderhauf, Sareh Shirazi, Adam Jacobson, Feras Dayoub, Edward Pepperell, Ben Upcroft, and Michael Milford. Place recognition with convnet landmarks: Viewpoint-robust, condition-robust, training-free. *Robotics: Science and Systems (RSS)*, 2015.

[16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[17] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks. *Intl. J. of Robotics Research*, page 0278364917734298, 2017.

[18] Peter M Williams. Using neural networks to model conditional multivariate densities. *Neural Computation*, 8(4):843–854, 1996.

[19] Zhengyou Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. *Image and Vision Computing*, 15(1):59–76, 1997.

[20] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems (NIPS)*, pages 487–495, 2014.