ViPR: Visual-Odometry-aided Pose Regression for 6DoF Camera Localization

Felix Ott¹, Tobias Feigl¹², Christoffer Lößfler¹², and Christopher Mutschler¹³

¹Fraunhofer Institute for Integrated Circuits IIS, Nuremberg, Germany
²Department of Computer Science, FAU Erlangen-Nuremberg, Germany
³Department of Statistics, Ludwig-Maximilians-University (LMU), Munich, Germany

{felix.ott | tobias.feigl | christoffer.loeffler | christopher.mutschler}@iis.fraunhofer.de

Abstract

Visual Odometry (VO) accumulates a positional drift in long-term robot navigation tasks. Although Convolutional Neural Networks (CNNs) improve VO in various aspects, VO still suffers from moving obstacles, discontinuous observation of features, and poor textures or visual information. While recent approaches estimate a 6DoF pose either directly from (a series of) images or by merging depth maps with optical flow (OF), research that combines absolute pose regression with OF is limited.

We propose ViPR, a novel modular architecture for long-term 6DoF VO that leverages temporal information and synergies between absolute pose estimates (from PoseNet-like modules) and relative pose estimates (from FlowNet-based modules) by combining both through recurrent layers. Experiments on known datasets and on our own Industry dataset show that our modular design outperforms state of the art in long-term navigation tasks.

1. Introduction

Real-time tracking of mobile objects (e.g., forklifts in industrial areas) allows to monitor and optimize workflows and tracks goods for automated inventory management. Such environments typically include large warehouses or factory buildings, and localization solutions often use a combination of radio-, LiDAR- or radar-based systems, etc.

However, these solutions often require infrastructure or they are costly in their operation. An alternative approach is a (mobile) optical pose estimation based on ego-motion. Such approaches are usually based on SLAM (Simultaneous Localization and Mapping), meet the requirements of exact real-time localization, and are also cost-efficient.

Available pose estimation approaches are categorized into three groups: classical, hybrid, and deep learning (DL)-based methods. Classical methods often require an infrastructure that includes either synthetic (i.e., installed in the environment) or natural (e.g., walls and edges) markers. The accuracy of the pose estimation depends on a large extent on suitable invariance properties of the available features such that they can be reliably recognized. However, to reliably detect features, we have to invest a lot of expensive computing time [38, 27]. Additional sensors (e.g., inertial sensors, depth cameras, etc.) or additional context (e.g., 3D models of the environment, prerecorded landmark databases, etc.) may increase the accuracy but also increase system complexity and costs [44]. Hybrid methods [66, 7, 6, 23, 74] combine geometric and machine learning (ML) approaches. For instance, ML predicts the 3D position of each pixel in world coordinates, from which geometry-based methods infer the camera pose [16].

Recent DL approaches partly address the above mentioned issues of complexity and cost, and also aim for high positioning accuracy, e.g., regression forests [51, 74] learn a mapping of images to positions based on 3D models of the environment. Absolute pose regression (APR) uses DL [63] as a cascade of convolution operators to learn poses only from 2D images. The pioneer PoseNet [33] has been extended by Bayesian approaches [31], long short-term memories (LSTMs) [77] and others [50, 26, 36, 11]. Recent APR methods such as VLocNet [72, 59] and DGRNets [42] introduce relative pose regression (RPR) to address the APR-

Figure 1: Our pose estimation pipeline solves the APR- and RPR-tasks in parallel, and recurrent layers estimate the final 6DoF pose.
task. While APR needs to be trained for a particular scene, RPR may be trained for multiple scenes [63]. However, RPR alone does not solve the navigation task.

For applications such as indoor positioning, existing approaches are not yet mature, i.e., in terms of robustness and accuracy to handle real-world challenges such as changing environment geometries, lighting conditions, and camera (motion) artifacts. This paper proposes a modular fusion technique for 6DoF pose estimation based on a PoseNet-like module and predictions of a relative module for VO. Our novel relative module uses the flow of image pixels between successive images computed by FlowNet2.0 [25] to capture time dependencies in the camera movement in the recurrent layers, see Fig. 1. Our model reduces the positioning error using this multitasking approach, which learns both the absolute poses based on monocular (2D) imaging and the relative motion for the task of estimating VO.

We evaluate our approach first on the small-scale 7-Scenes [66] dataset. As other datasets are unsuitable to evaluate continuous navigation tasks we also release a dataset that can be used to evaluate various problems arising from real industrial scenarios such as inconsistent lighting, occlusion, dynamic environments, etc. We benchmark our approach on both datasets against existing approaches [33, 77] and show that we consistently outperform the accuracy of their pose estimates.

The rest of the paper is structured as follows. Section 2 discusses related work. Section 3 provides details about our architecture. We discuss available datasets and introduce our novel Industry dataset in Section 4. We present experimental results in Section 5 before Section 6 concludes.

2. Related Work

SLAM-driven 3D point registration methods enable precise self-localization even in unknown environments. Although VO has made remarkable progress over the last decade, it still suffers greatly from scaling errors of real and estimated maps [43, 69, 49, 29, 34, 35, 40, 54, 4, 39]. With more computing power, Visual Inertial SLAM combines VO with Inertial Measurement Unit (IMU) sensors to partly resolve the scale ambiguity, to provide motion cues without visual features [43, 70, 29], to process more features, and to make tracking more robust [69, 34]. Multiple works combine global localization in a scene with SLAM/(Inertial) VO [46, 17, 55, 64, 22, 52, 28]. However, recent SLAM methods do not yet meet industry-strength with respect to accuracy and reliability [57, 18] as they need undamaged, clean and undisguised markers [39, 30] and as they still suffer from long-term stability and the effects of movement, sudden acceleration and occlusion [75]. SIFT-like point-based features [45] for the localization from landmarks [3, 24, 41, 78] require efficient retrieval methods, use VLAD encodings such as DenseVLAD [71], use anchor points such as AnchorNet [60], or use RANSAC-based optimization such as DSAC [6] and ActiveSearch [61].

VO primarily addresses the problem of separating ego-motion from feature-motion and suffers from area constraints, poorly textured environments, scale drift, a lack of an initial position, and thus inconsistent camera trajectories [10]. Instead, PoseNet-like architectures (see Sec. 2.1) that estimate absolute poses on single-shot images are more robust, less compute-intensive, and can be trained in advance on application data. Unlike VO, they do not suffer from a lack of initial poses and do not require access to camera parameters, good initialization, and handcrafted features [65]. Although the joint estimation of relative poses may contribute to increasing accuracy (see Sec. 2.2), such hybrid approaches still suffer from dynamic environments, as they are often trained offline in quasi-rigid environments. While optical flow (see Sec. 2.3) addresses these challenges it has not yet been combined with APR for 6DoF self-localization.

2.1. Absolute Pose Regression (APR)

Methods that derive a 6DoF pose directly from images have been studied for decades. Therefore, there are currently many classic methods whose complex components are replaced by machine learning (ML) or DL. For instance, RelocNet [2] learns metrics continuously from global image features through a camera frustum overlap loss. CamNet [15] is a coarse (image-based)-to-fine (pose-based) retrieval-based model that includes relative pose regression to get close to the best database entry that contains extracted features of images. NNet [37] queries a database for similar images to predict the relative pose between images and a RANSAC [67] solves the triangulation to provide a position. While those classic approaches have already been extended with DL-components their pipelines are expensive as they embed feature matching and projection and/or manage a database. Most recent (and simple) DL-based also outperform their accuracies.

The key idea of PoseNet [33] and its variants [32, 31, 20, 77, 76, 79, 58, 65, 56, 66] among others such as BranchNet [56] and Hourglass [66] is to use a CNN for camera (re-)localization. PoseNet works with scene elements of different scales and is partially insensitive to light changes, occlusions and motion blur. However, while Dense PoseNet [33] crops subimages, PoseNet2 [32] jointly learns network and loss function parameters, [31] links a Bernoulli function and applies variational inference [20] to improve the positioning accuracy. However, those variants work with single images, and hence, do not use the temporal context (which is available in continuous navigation tasks), that could help to increase accuracy.

In addition to PoseNet+LSTM [77], there are also similar approaches that exploit time-context that is inherently given by consecutive images (i.e., DeepVO [79],
ContextualNet [58], and VidLoc [12]). Here, the key idea is to identify temporal connections in-between the feature vectors (extracted from images) with LSTM-units and to only track feature correlations that contribute the most to the pose estimation. However, there are hardly any long-term dependencies between successive images, and therefore LSTMs give worse or equal accuracy to, for example, simple averaging over successively estimated poses [65]. Instead, we combine estimated poses from time-distributed CNNs with estimates of the OF to maintain the required temporal context in the features of image series.

2.2. APR/RPR-Hybrids

In addition to approaches that derive a 6DoF pose directly from an image there are hybrid methods that combine them with VO to increase the accuracy. VLocNet [72] is closely related to our approach as it estimates a global pose and combines it with VO (but it does not use OF). To further improve the (re-)localization accuracy VLocNet++ [59] uses features from a semantic segmentation. However, we use different networks and do not need to share weights between VO and the global pose estimation. DGRNets [42] estimates both the absolute and relative poses, concatenates them, and uses recurrent CNNs to extract temporal relations between consecutive images. This is similar to our approach but we estimate the relative motion with OF, which allows us to train in advance on large datasets, making the model more robust. MapNet [8] learns a map representation from input data, combines it with GPS, inertial data, and unlabeled images, and uses pose graph optimization (PGO) to combine absolute and relative pose predictions. However, compared to all other methods the most accurate extension of it, MapNet+PGO, does not work on purely visual information, but exploits additional sensors.

2.3. Optical Flow

Typically, VO uses OF to extract features from image sequences. Motion fields, see Fig. 2 (middle), are used to estimate trajectories of pixels in a series of images. For instance, Flowdometry [53] and LS-VO [13] estimate displacements and rotations from OF. [48] proposed a VO-based dead reckoning system that uses OF to match features. [80] combined two CNNs to estimate the VO-motion: FlowNet2-ss [25] estimates the OF and PCNN [14] links two images to process global and local pose information. However, to the best of our knowledge, we are the first to propose an OF-based architecture that estimates the relative camera movement through RNNs, using OF [25].

3. Proposed Model

After a data preprocessing that crops subimages of size \(224 \times 224 \times 3\) from a sequence of four images, our pose regression pipeline consists of three parts (see Fig. 3): an APR-network, a RPR-network, and a 6DoF pose estimation (PE) network. PE uses the outputs of the APR- and RPR-networks to provide the final 6DoF pose.

3.1. Absolute Pose Regression (APR) Network

Our APR-network predicts the 6DoF camera pose from three input images based on the original PoseNet [33] model (i.e., essentially a modified GoogLeNet [68] with a regression head instead of a softmax) to train and predict the absolute positions \(p \in \mathbb{R}^3\) in the Euclidean space and the absolute orientations \(q \in \mathbb{R}^4\) as quaternions. From a single monocular image \(I\) the model predicts the pose as approximations to the actual \(p\) and \(q\). As the original model learns the image context, based on shape and appearance of the environment, but does not exploit the time context and relation between consecutive images [32], we adapted the model to a time-distributed variant. Hence, instead of a single image the new model receives three (consecutive) input images (at timesteps \(t_{n-1}, t_n,\) and \(t_{n+1}\)), see top part of Fig. 3, uses three separate dense layers (one for each pose) with 2,048 neurons each, and each of the dense layers yields a pose. The middle pose yields the most accurate position for the image at time step \(t_n\).

3.2. Relative Pose Regression (RPR) Network

Our RPR-network uses FlowNet2.0 [25] on each consecutive pairs of the four input images to compute an approximation of the OF (see Fig. 2) and to predict three relative poses for later use. As displacements of similar length but from different camera viewing directions result in different OFs, the displacement and rotation of the camera between pairwise images must be relative to the camera’s viewing direction of the first image. Therefore, we transform each camera’s global coordinate systems \((x_n, y_n, z_n)\) to the same local coordinate system \((\tilde{x}_n, \tilde{y}_n, \tilde{z}_n)\) by

\[
\begin{pmatrix}
\tilde{x}_n \\
\tilde{y}_n \\
\tilde{z}_n
\end{pmatrix}
= R
\begin{pmatrix}
x_n \\
y_n \\
z_n
\end{pmatrix},
\]

with the rotation matrix \(R\). The displacement \(\Delta \tilde{x}_n, \Delta \tilde{y}_n, \Delta \tilde{z}_n\) is the difference between the transformed coordinate systems. The displacement in global coordinates
is obtained by a back-transformation of the predicted displacement, such that
\[ R^T = R^{-1} \] and \[ RR^T = I. \] (3)

Fig. 4 shows the structure of the RPR-network. Similar to the APR-network, the RPR-network also uses a stack of images, i.e., three OF-fields from the four input images of the timesteps \( t_{n-1}, \ldots, t_{n+2} \), to include more time context.

In a preliminary study, we found that our recurrent units struggle to remember temporal features when the direct input of the OF is too large (raw size \( 224 \times 224 \times 3 \, px \)). This is in line with findings from Walch et al. [77]. Hence, we split the OF in zones and compute the mean value for each the \( u \)- and \( v \)-direction. We reshape \( 16 \times 16 \) number of zones in both directions to the size \( 2 \times 256 \). The final concatenation results in a smaller total size of \( 3 \times 512 \). The LSTM-output is forwarded to 2 FC-layers that regress both the displacement (size \( 3 \times 3 \)) and rotation (size \( 3 \times 4 \)).

Figure 3: Pipeline of the ViPR-architecture. Data preprocessing (grey): Four consecutive input images \( (t_{n-1}, \ldots, t_{n+2}) \) are center cropped. For the absolute network the mean is subtracted. For the relative network the OF is precomputed by FlowNet2.0 [25]. The absolute poses are predicted by our time-distributed APR-network (yellow). The RPR-network (purple) predicts the transformed relative displacements and rotations on reshaped mean vectors of the OF with (stacked) LSTM-RNNs. The PE modules (green) concatenates the absolute and relative modules and predicts the absolute 6DoF poses with stacked LSTM-RNNs.

The 2 FC-layers use the following loss function to predict the relative transposed poses \( \Delta \hat{p}^T \) and \( \Delta q \):
\[
L = \alpha_2 \| \Delta \hat{p}^T - \Delta p^T \|_2 + \beta_2 \| \Delta \hat{q} - \Delta q \|_2. \] (4)

The first term accounts for the predicted and transformed displacement \( \Delta \hat{p}^T \) to the ground truth displacement \( \Delta p^T \) with an \( L^2 \)-norm. The second term quantifies the error of the predicted rotation to the normalized ground truth rotation using an \( L^2 \)-norm. Both terms are weighted by the hyperparameters \( \alpha_2 \) and \( \beta_2 \). A preliminary grid search with a fixed \( \alpha_2 = 1 \) revealed an optimal value for \( \beta_2 \) that depends on the scaling of the environment.

3.3. 6DoF Pose Estimation (PE) Network

Our PE-network predicts absolute 6DoF poses from the outputs of both the APR- and RPR-networks, see Fig. 5. The PE-network takes as input the absolute position \( p_i = (x_i, y_i, z_i) \), the absolute orientation \( q_i = (w_i, p_i, q_i, r_i) \), the relative displacement \( \Delta p_i = (\Delta x_i, \Delta y_i, \Delta z_i) \), and the rotation change \( \Delta q_i = (\Delta w_i, \Delta p_i, \Delta q_i, \Delta r_i) \). As we feed poses from three sequential timesteps \( t_{n-1}, t_n, \) and \( t_{n+1} \) as input to the model it is implicitly time-distributed. The 2 stacked LSTM-layers and the 2 FC-layers return a 3DoF absolute position \( p \in \mathbb{R}^3 \) and a 3DoF orientation \( q \in \mathbb{R}^4 \) using the following loss:
\[
L(p, \Delta P) = \alpha_3 \| \hat{p} - p \|_2 + \beta_3 \| q - \hat{q} \|_2. \] (5)
As in a preliminary grid search we chose $L_2$-norms with a fixed $\beta_3 = 1$ that revealed an optimal value for $\alpha_3$.

### 4. Evaluation Datasets

To train our network we need two different types of image data: (1) images annotated with their absolute poses for the APR-network, and (2) images of OF, annotated with their relative poses for the RPR-network.

**Datasets to evaluate APR.** Publicly available datasets for absolute pose regression (Cambridge Landmarks [33] and TUM-LSI [77]) either lack accurate ground truth labels or the proximity between consecutive images is too large to embed meaningful temporal context. The Aalto University [37], Oxford RobotCar [47], DeepLoc [59] and CMU Seasons [62] datasets solve the small-scale issue of the 7-Scenes [66] dataset, but are barely used for evaluation of state-of-the-art techniques or consider only automotive-driving scenarios. The 12-Scenes [73] dataset is only used by DSAC++ [5]. For our industrial application these datasets are insufficient. 7-Scenes [66] only embeds scenes with less training data and only enables small scene-wise evolutions, but is mainly used for evaluation. Hence, to compare ViPR with recent techniques we use the 7-Scenes [66] dataset. Furthermore, we recorded the Industry dataset (see Sec. 4.1) that embeds three different industrial-like scenarios to allow a comprehensive and detailed evaluation with different movement patterns (such as slow motion and fast rotation).

**Datasets to evaluate RPR.** To evaluate the performance of the RPR and its contribution to ViPR, we also need a dataset with a close proximity between consecutive images. This is key to calculate the relative movement with OF. However, most publicly available datasets (Middlebury [1], MPI Sintel [9], KITTI Vision [21], and FlyingChairs [19]) either do not meet this requirement or the OF pixel velocities do not match those of real-world applications. Hence, we directly calculate the OF from images with FlowNet2.0 [25] to train the RPR on it. Our novel Industry dataset allows this, while retaining a large, diverse environment with hard real-world conditions, as described in the following.

### 4.1. Industry Dataset

We designed the Industry dataset to suite the requirements of both the APR- and the RPR-network and published the data \(^1\) at large-scale (1,320 m\(^2\)) using a high-precision (< 1 mm) laser-based reference system. Each scenario presents different challenges (such as dynamic ego-motion with motion blur), various environmental characteristics (such as different geometric scales, light changes, i.e., artificial and natural light), and ambiguously structured elements, see Fig. 6.

**Industry Scenario #1** [44] has been recorded with 8 cameras (approx. 60° field-of-view (FoV) each) mounted on a stable apparatus to cover 360° (with overlaps) that has been moved automatically at a constant velocity of approx. 0.3 m/s. The height of the cameras is at 1.7 m. The scenario contains 521,256 images ($640 \times 480$ px) and densely covers an area of 1,320 m\(^2\). The environment imitates a typical warehouse scenario under realistic conditions. Besides well-structured elements such as high-level racks with goods, there are also very ambiguous and homogeneously textured elements (e.g., blanket white or dark black walls). Both natural and artificial light illuminates volatile structures such as mobile work benches. While the training dataset is composed of a horizontal and vertical zig-zag movement of the apparatus the test datasets movements vary to cover different properties for a detailed evaluation, e.g., different environmental scalings (i.e., scale transition, cross, large scale, and small scale), network generalization (i.e., generalize open, generalize racks, and cross), fast rotations (i.e., motion artifacts was recorded on a fork lift at 2.26 m height) and volatile objects (i.e., volatility).

**Industry Scenario #2** uses three 170° cameras (with overlaps) on the same apparatus the same height. The recorded 11,859 training images (1$280 \times 720$ px) represent a horizontal zig-zag movement (see Fig. 7a) and 3,096 test images represent a diagonal movement (see Fig. 7b). Compared to Scenario #1 this scenario has more variation in its velocities (between $0 \text{ m/s}$ and $0.3 \text{ m/s}$, SD $0.05 \text{ m/s}$).

**Industry Scenario #3** uses four 170° cameras (with overlaps) on a forklift truck at a height of 2.26 m. Both the training and test datasets represents camera movements at varying, faster, and dynamic speeds (between $0 \text{ m/s}$ and $1.5 \text{ m/s}$, SD $0.51 \text{ m/s}$). This makes the scenario the most challenging one. The training trajectory (see Fig. 7c) consists of 4,166 images and the test trajectory (see Fig. 7d) consists of 1,687 images. In contrast to the Scenarios #1 and #2 we train and test a typical industry scenario on dynamic movements of a forklift truck. However, one of cameras’ images were corrupted in the test dataset, and thus, not used in the evaluation.

\(^1\) Industry dataset available at: [https://www.iis.fraunhofer.de/warehouse](https://www.iis.fraunhofer.de/warehouse).

Provided are raw images and corresponding labels: $p$ and $q$.  

---

\(\text{Figure 5: Pipeline of the 6DoF PE-architecture. The input tensor (}3 \times 14\text{) contains absolute positions and orientations and relative displacements and rotations at timesteps } t_{n-1}, t_n, t_{n+1}. \text{2 stacked LSTMs process the tensor and 2 FC-layers return the pose.}\)
5. Experimental Results

To compare ViPR with state-of-the-art results, we first briefly describe our parameterization of PoseNet [33] and PoseNet+LSTM [77] in Sec. 5.1. Next, Sec. 5.2 presents our results. We highlight the performance of ViPR’s sub-networks (APR, APR+LSTM) individually, and investigate both the impact of RPR and PE on the final pose estimation accuracy of ViPR. Sec. 5.3 shows results of the RPR-network. Finally, we discuss general findings and show run-times of our models in Sec. 5.4.

For all experiments we used an AMD Ryzen 7 2700 CPU 3.2 GHz equipped with one NVidia GeForce RTX 2070 with 8 GB GDDR6 VRAM. Tab. 1 shows the median error of the position in m and the orientation in degrees. The second column reports the spatial extends of the datasets. The last column reports the improvement in position accuracy of ViPR (in %) over APR-only.

5.1. Baselines

As a baseline we report the initially described results on 7-Scenes of PoseNet [33] and PoseNet+LSTM [77] (in italic). We further re-implemented the initial variant of PoseNet and trained it from scratch with \( \alpha_1 = 1, \beta_1 = 30 \) (thus optimizing for positional accuracy at the expense of orientation accuracy). Tab. 1 (cols. 3 and 4) shows our implementation’s results next to the initially reported ones (on 7-Scenes). We see that (as expected) the results of the PoseNet implementations differ due to changed values for \( \alpha_1 \) and \( \beta_1 \) in our implementation.

5.2. Evaluation of the ViPR-Network

In the following, we evaluate our method in multiple scenarios with different distinct challenges for the pose estimation task. 7-Scenes focuses on difficult motion blur conditions of typical human motion. We then use the Industry Scenario #1 to investigate various challenges at a larger scale, but with mostly constant velocities. Industry Scenarios #2 and #3 then focus on dynamic, fast ego-motion of a moving forklift truck at large-scale.

7-Scenes [66]. For both architectures (PoseNet and ViPR), we optimized \( \beta \) to weight the impact of position and orientation such that it yields the smallest total median error. Both APR+LSTM and ViPR return a slightly lower pose estimation error of 0.33 m and 0.32 m than PoseNet+LSTM with 0.34 m. ViPR yields an average improvement of the position accuracy of 3.18 % even in strong motion blur situations. The results indicate that ViPR relies on a plausible optical flow component to achieve performance that is superior to the baseline. In situations of negligible motion between frames the median only improves by 0.02 m. However, the average accuracy gain still shows that ViPR performs en par or better than the baselines.

Stable Motion Evaluation. For the Industry Scenario #1 dataset, we train the models on the zig-zag trajectories, and test them on specific sub-trajectories with individual challenges, but at almost constant velocity. In total, ViPR improves the position accuracy by 12.27% on average (min.: 4.03%; max.: 25.31 %) while the orientation error is similar for most of the architectures and test sets.

In environments with volatile features, i.e., objects that are only present in the test dataset, we found that ViPR (with optical flow) is significantly (6.41 %) better compared to APR-only. However, the high angular error of 77.54° indicates an irrecoverable degeneration of the APR-part. In tests with different scaling of the environment, we think that ViPR learns an interpretation of relative and absolute position regression, that works both in small and large proximity to environmental features, as ViPR improves by 15.52 % (scale trans.) and 14.41 % (small scale) or 10.68 % (large scale). When the test trajectories are located within areas that embed only few or no training samples (gener. racks and open), ViPR still improves over other methods with 4.03-11.75 %. The highly dynamic test on a forklift truck (motion artifacts) is exceptional here as only the test dataset contains dynamics and blur, and hence, challenges ViPR most. However, ViPR still improves by 10.01 % over APR-only, despite the data dynamic’s absolute novelty.

In summary, ViPR decreases the position median significantly by about 2.53 m than only APR+LSTM (4.89 m). This and the other findings are strong indicators that the relative component RPR significantly supports the final pose estimation of ViPR.
Table 1: Pose estimation results (position and orientation median error in meters (m) and degrees (°)) and total improvement of PE in % on the 7-Scenes [66] and Industry datasets. The best results are bold and underlined ones are additionally referenced in the text.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Spatial extend (m)</th>
<th>PoseNet [33] (original/our param.)</th>
<th>PoseNet+LSTM [77]</th>
<th>APR-only</th>
<th>APR+LSTM (our param.)</th>
<th>ViPR*</th>
<th>Improv. ViPR (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>chess</td>
<td>3.0 × 2.0 × 1.0</td>
<td>0.32/0.24 / 4.06/7.79</td>
<td>0.24 / 5.77</td>
<td>0.23 / 7.96</td>
<td>0.27 / 9.66</td>
<td>0.22 / 7.89</td>
<td>+1.74</td>
</tr>
<tr>
<td>fire</td>
<td>2.5 × 1.0 × 1.0</td>
<td>0.47 / 0.39 / 4.4 / 12.40</td>
<td>0.34 / 11.9</td>
<td>0.39 / 12.85</td>
<td>0.50 / 15.70</td>
<td>0.38 / 12.74</td>
<td>+2.56</td>
</tr>
<tr>
<td>heads</td>
<td>2.0 × 0.5 × 1.0</td>
<td>0.29 / 0.21 / 6.00 / 16.46</td>
<td>0.21 / 13.7</td>
<td>0.22 / 16.48</td>
<td>0.23 / 16.91</td>
<td>0.21 / 16.41</td>
<td>+3.64</td>
</tr>
<tr>
<td>office</td>
<td>2.5 × 2.0 × 1.5</td>
<td>0.48/0.33 / 3.84/10.08</td>
<td>0.30 / 8.08</td>
<td>0.36 / 10.11</td>
<td>0.37 / 10.83</td>
<td>0.35 / 9.59</td>
<td>+4.01</td>
</tr>
<tr>
<td>pumpkin</td>
<td>2.5 × 1.0 × 1.0</td>
<td>0.47 / 0.45 / 8.42 / 8.70</td>
<td>0.43 / 7.00</td>
<td>0.39 / 8.57</td>
<td>0.86 / 49.46</td>
<td>0.37 / 8.45</td>
<td>+5.12</td>
</tr>
<tr>
<td>red kitchen</td>
<td>4.0 × 3.0 × 1.5</td>
<td>0.59 / 0.41 / 8.64 / 9.09</td>
<td>0.37 / 8.83</td>
<td>0.42 / 9.33</td>
<td>1.06 / 50.67</td>
<td>0.40 / 9.32</td>
<td>+4.76</td>
</tr>
<tr>
<td>stairs</td>
<td>2.5 × 2.0 × 1.5</td>
<td>0.47 / 0.36 / 6.93 / 13.69</td>
<td>0.40 / 13.7</td>
<td>0.31 / 12.49</td>
<td>0.42 / 13.50</td>
<td>0.31 / 12.65</td>
<td>+0.46</td>
</tr>
<tr>
<td>☞ total</td>
<td>0.44 / 0.34 / 7.47 / 11.17</td>
<td>0.31 / 9.85</td>
<td>0.33 / 11.11</td>
<td>0.53 / 23.82</td>
<td>0.32 / 11.01</td>
<td>+3.18</td>
<td></td>
</tr>
</tbody>
</table>

Industry Scenario #2 is designed to evaluate for unknown trajectories. Hence, training trajectories represent an orthogonal grid, and test trajectories are diagonal. In total, ViPR improves the position accuracy by 30.2% on average (min.: 17.87%; max.: 46.49%). Surprisingly, the orientation error is comparable for all architectures, except ViPR. We think that this is because ViPR learns to optimize its position based on the APR- and RPR- orientations, and hence, exploits these orientations to improve its position estimate, that we prioritized in the loss function. APR-only yields an average position accuracy of 0.27 m, while the pure PoseNet yields position errors of 0.36 m on average, but APR+LSTM results in an even worse accuracy of 1.69 m. Instead, the novel ViPR outperforms all significantly with 0.2 m. Compared to our APR+LSTM approach, we think that ViPR on the one hand interprets and compensates the (long-term) drift of RPR and on the other hand smooths the short-term errors of APR, as PE counteracts the accumulation of RPR’s scaling errors with APR’s absolute estimates. Here, the synergies of the networks in ViPR are particularly effective. This is also visualized in Fig. 8a: the green (ViPR) trajectory aligns more smoothly to the blue baseline when the movement direction changes. This also indicates that the RPR component is necessary to generalize to unknown trajectories and to compensate scaling errors.

Dynamic motion evaluation. In contrast to the other datasets, the Industry Scenario #3 includes fast, large-scale, and high dynamic ego-motion in both training and test datasets. However, all estimators result in similar findings as Scenario #2 as both scenarios embed motion dynamics and unknown trajectory shapes. Accordingly, ViPR again improves the position accuracy by 17.41 % on average (min.: 11.47%; max.: 20.64 %), but this time exhibits very similar orientation errors. Improved orientation accuracy compared to Scenario #2 is likely due to diverse orientations available in this dataset’s training.

Fig. 8b shows exemplary results that visualize how ViPR handles especially motion changes and motion dynamics (see the abrupt direction change between $x \in [8 - 9] m$ and $y \in [14 - 16] m$). The results also indicate that ViPR predicts the smoothest and most accurate trajectories on unknown trajectory shapes (compare the trajectory segments between $x \in [11 - 12] m$ and $y \in [14 - 16] m$). We think the reason why ViPR significantly outperforms APR
by 20.13\% here is because of the synergy of APR, RPR, and PE. RPR contributes most in fast motion-change, i.e., in motion blur situation. The success of RPR may also indicate that RPR differentiates between ego- and feature-motion to more robustly estimate a pose.

5.3. Evaluation of the RPR-Network

We use the smaller FlowNet2-s [25] variant of FlowNet2.0 as this has faster runtimes (140 Hz), and use it pretrained on the FlyingChairs [19], ChairsSDHom and FlyingThings3D datasets. To highlight that RPR contributes to the accuracy of the final pose estimates of ViPR, we explicitly test it on the Industry Scenario #3 that embeds dynamic motion of both ego- and feature-motion. The distance between consecutive images is up to 20 cm, see Fig. 9. This results in a median error of 2.49 cm in x- and 4.09 cm in y-direction on average (i.e., the error is between 12.5\% and 20.5\%). This shows that the RPR yields meaningful results for relative position regression in a highly dynamic and difficult setting. It furthermore appears to be relatively robust in its predictions despite both ego- and feature-motion.

5.4. Discussion

6DoF Pose Regression with LSTMs. APR-only increases the positional accuracy over PoseNet for all datasets, see Tab. 1. We found that the position errors increase when we use methods with independent and single-layer LSTM-extensions [77, 79, 58, 65] on both the 7-Scenes and the Industry datasets, by 0.04 m up to 2.07 m. This motivated us to investigate stacked LSTM-layers only for the RPR- and PE-networks. We support the statement of Seifi et al. [65] that the motion between consecutive frames is too small, and thus, naive CNNs are already unable to embed them. Hence, additionally connected LSTMs are also unable to discover and track meaningful temporal and contextual relations between the features.

Influence of RPR to ViPR. To figure out the information gain of the RPR-network we also constructed ViPR in a closed end-to-end architecture through direct concatenation of the CNN-encoder-output (APR) and the LSTM-output (RPR). For a smaller OF-input (3 x 3) of the RPR-model the accuracy of the 7-Scenes [66] dataset increases, but decreases for the Industry dataset. This stems from the fact that the relative movements of the 7-Scenes dataset are too small (< 2 cm) compared to the Industry dataset (approx. 20 cm). Hence, ViPR’s contribution is limited here.

Comparison of ViPR to state-of-the-art methods. VLocNet++ [59] currently achieves the best results on 7-Scenes [66], but due to the small relative movement and the high ground truth error compared to VLocNet’s results a plausible evaluation is not possible regarding industrial applications. MapNet [8] achieves (on average) better results than ViPR on the 7-Scenes dataset, but results in a similar error, e.g., 0.30 m and 12.08° on the stairs set against ViPR’s 0.31 m and 12.65°. MapNet has an improvement of 8.7\% over PoseNet2 [32] and achieves 41.4 m and 12.5° on the RobotCar [47] dataset. However, a fair evaluation on this dataset with state-of-the-art methods requires results and code from VLocNet [72, 59].

Runtimes. The training of the APR takes 0.86 s per iteration for a batch size of 50 (GoogLeNet [68]) on our hardware setup. The training of the RPR and PE is faster (0.065 s) even at a higher batch size of 100, as these models are smaller (214,605, resp. 55,239, parameters). Hence, it is possible to retrain the PE-network quickly upon environment changes. The inference time of ViPR is between 7 ms and 9 ms per sample (PoseNet: avg. 5 ms, FlowNet2-s: avg. 9 ms). In addition, ViPR does not require domain knowledge to provide scenario-dependent applicability, nor does it need a compute-intensive matcher like brute force or RANSAC [67, 6]. However, instead of PoseNet, ViPR can also use such classical approaches in its modular process pipeline. DenseVLAD [71] and classical approaches are 10x (200-350 ms/sample) more computationally intensive than today’s deep pose regression variants.

6. Conclusion

In this paper, we addressed typical challenges of learning-based visual self-localization of a monocular camera. We introduced a novel DL-architecture that makes use of three modules: an absolute and a relative pose regressor module, and a final regressor that predicts a 6DoF pose by concatenating the predictions of the two former modularities. To show that our novel architecture improves the absolute pose estimates, we compared it with a publicly available dataset and proposed novel Industry datasets that enable a more detailed evaluation of different (dynamic) movement patterns, generalization, and scale transitions.

Acknowledgements

This work was supported by the Bavarian Ministry for Economic Affairs, Infrastructure, Transport and Technology through the Center for Analytics-Data-Applications (ADA-Center) within the framework of "BAYERN DIGITAL II".
References


[50] Iaroslav Melekhov, Juho Ylisonas, Juho Kannala, and Esa Rahtu. "Image-Based Localization Using Hourglass Networks". In Intl. Conf. on Computer Vision Workshop (ICCVW), pages 870–877, Venice, Italy, 2017. 1

[51] Lili Meng, Jianhui Chen, Frederick Tung, James J. Little, Julien Valentin, and Clarence W. da Silva. "Backtracking


[64] Soroush Seifi and Tinne Tuytelaars. “How to improve CNN-based 6-DoF Camera Pose Estimation”. In *Intl. Conf. on Computer Vision (ICCV)*, 2019. 2, 3, 8


[75] Florian Walch, Caner Hazirbas, Laura Leal-Taixé, Torsten Sattler, Sebastian Hilsenbeck, and Daniel Cremers. "Image-Based Localization Using LSTMs for Structured Feature

