Exploring Representation Learning With CNNs for Frame-to-Frame Ego-Motion Estimation

Gabriele Costante, Michele Mancini, Paolo Valigi, and Thomas A. Ciarfuglia

***Abstract*—Visual ego-motion estimation, or briefly visual odometry (VO), is one of the key building blocks of modern SLAM systems. In the last decade, impressive results have been demonstrated in the context of visual navigation, reaching very high localization performance. However, all ego-motion estimation systems require careful parameter tuning procedures for the specific environment they have to work in. Furthermore, even in ideal scenarios, most state-of-the-art approaches fail to handle image anomalies and imperfections, which results in less robust estimates. VO systems that rely on geometrical approaches extract sparse or dense features and match them to perform frame-to-frame (F2F) motion estimation. However, images contain much more information that can be used to further improve the F2F estimation. To learn new feature representation, a very successful approach is to use deep convolutional neural networks. Inspired by recent advances in deep networks and by previous work on learning methods applied to VO, we explore the use of convolutional neural networks to learn *both* the best visual features and the best estimator for the task of visual ego-motion estimation. With experiments on publicly available datasets, we show that our approach is robust with respect to blur, luminance, and contrast anomalies and outperforms most state-of-the-art approaches even in nominal conditions.**

***Index Terms*—Visual Learning, Visual-Based Navigation.**

I. Introduction

EGO-MOTION estimation is a fundamental building block of any robotic system, needed for localization and route planning, and for the more complex task of mapping an unknown environment. When vision comes into play, the task of estimating the ego-motion of cameras is referred to as Visual Odometry (VO). In recent years, impressive results have been shown in the context of monocular visual odometry [1], [2], [3], [4].

Most visual odometry approaches are grounded on the estimate of the camera motion between pairs of consecutive frames. This frame to frame (F2F) estimate is in most cases computed with geometric methods, *i.e.* through the use of projective geometry relations between 3D points of the scene and their projection on the image plane, or by minimizing the gradient of the pixel intensities across consecutive images [5]. The initial F2F estimate is then refined with different strategies, such as bundle adjustment on a sliding window of previous frames [6], or loop closure detection [7].

However, the initial feature extraction process is critical to the whole estimation process and, because of that, while in structured and controlled environments (*e.g.*, with a large amount of texture and without dynamic objects) these standard approaches provide good results, their performance drops quickly when facing challenging and unpredicted scenarios. These uncertainties can have various causes: (i) illumination changes over time and scenes; (ii) presence of dynamic objects; (iii) different camera calibrations; (iv) low-textured environments, noise and blur. Strengthening the estimation process against these issues requires a twofold action. On one side more informative and robust features are needed, on the other the estimation algorithms are required to better handle noise and unpredicted input anomalies. As far as the estimation aspect goes, new approaches have recently been proposed, that start from the perspective of a statistical learning problem [8], [9], and show many desirable properties. Learning an estimator from data requires good labelled datasets, but when these are available, the learned estimator is robust to illumination changes, noise and blur. From the point of view of what features to use for robust ego-motion estimation, recent advances in Deep Networks applied to representation learning have shown a lot of potential [10]. The core of these approaches in Computer Vision is to use deep Convolutional Neural Networks (CNNs) to learn the best convolutional filters to apply to input image for a given task.

Guided by the previous considerations, in this work we explore a different strategy for performing visual ego-motion estimation. We do not assume any pre-defined procedure to compute the transformation between frames. Instead, following the deep network learning paradigm, we allow the system to autonomously select both the portion of input data that is crucial to achieving robust F2F motion estimation and the strategy for computing it. In particular, inspired by the results achieved with deep architectures, we propose a novel F2F estimation strategy that predicts the camera motion using a CNN (see Figure 1). By learning the CNN, our approach is able to autonomously select the *most important visual cues* and the *best strategy* to compute F2F estimates that are *robust to blur, luminance and constrast anomalies*.

II. Related Work

To our knowledge, algorithms that compute VO can be divided into different categories, according to the kind of processing used for computing ego-motion: Geometric Methods,

Manuscript received August 29, 2015; accepted November 18, 2015. Date of publication December 4, 2015; date of current version December 28, 2015. This work was supported by the NVIDIA Corporation with the donation of the Tesla K40 GPU. This paper [illegible] recommended for publication by Associate Editor B. Caputo and Editor [illegible]ecka upon evaluation of the reviewers' comments.

The authors are with the Department of Engineering, University of Perugia, Perugia 06125, Italy (e-mail: thomas.ciarfuglia@unipg.it; paolo.valigi@unipg.it; gabriele.costante@studenti.unipg.it; michele.mancini1@studenti.unipg.it).

Digital Object Identifier 10.1109/LRA.2015.2505717

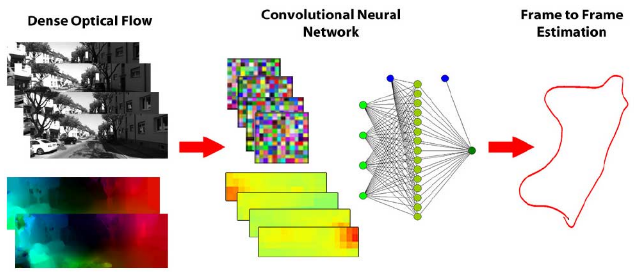

Fig. 1. Overview of the proposed Convolutional Neural Network VO estimator. First, dense optical flow is extracted from consecutive images and then fed into a chain of convolutional filters that extract more visual information. Finally, the new features are used by a fully connected neural network to estimate the F2F camera motion.

that are further divided in Feature-based and Direct Methods, and Learning methods, that learn the estimation function from data.

In the following, the related works for each kind of approach are presented.

A. Geometric Methods

1) Feature-Based Methods: Feature-based methods rely on the extraction of visual salient points from each image in the stream, tracking them frame after frame until they disappear from view. Typically, the process matches 2D point features across two or more frames, then reconstructs their 3D position using triangulation. Examples of these approaches are [11], [12], [13] for stereo and [14], [15] for mono.

To address the problem of scale estimation a number of solutions were proposed. In [16] the scale is recovered using the extra information from an IMU, while in [1] and, more recently, in [2] an optimization approach using loop closing information is able to recover scale errors. In most cases, to address the problem of scale uncertainty, extra-information is needed in the form of an extra sensor, loop closing data or metric set up information.

2) Dense Methods: The main limit of feature based VO estimators is the extraction of the features. Feature-based methods are fast but can easily fail in contexts where feature extraction is difficult, such as low textured or blurred images. Dense methods try to use the whole image instead. The process is similar to dense optical flow extraction [17], but instead of extracting the motion of each pixel, it extracts the underlying camera motion. Clearly, dense methods can achieve better accuracy than feature-based ones, but are more computationally intense, and only recently some have reached real-time operation on common hand-held devices or embedded systems, enabling their use on Micro Aerial Vehicle (MAV) platforms [2], [3], [4].

B. Learning Methods

While geometric methods mainly make strong assumptions about what to extract from images and how to use this information to compute motion estimation, learning methods try to infer them from data. The first examples of learning-based VO are [18] and [19], where the authors divide each frame into cells and compute an average optical flow for each block, then they train a K-Nearest Neighbor (KNN) regressor in the first work and an Expectation Maximization (EM) algorithm in the second. In [9], [20] and [8] a similar feature parametrization of optical flow is used together with Coupled Gaussian Processes (CGP) as a regression algorithm.

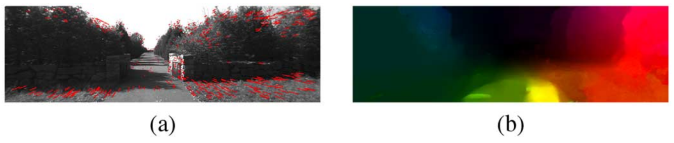

Fig. 2. Optical flow fields a) feature based (sparse) b) intensity based (dense).

The common aspect of these previous works is the use of quantized sparse optical flow, such as Histogram of Optical Flow (HOF). As far as we know, no previous work on this problem has proposed learning the representation from data. Since the recent rise of deep architectures has shown that it is possible to give to the learning algorithm the plain input and let it learn the correct representation for the task with impressive results [21], [22], [23], we propose to apply the same techniques to the feature extraction part of ego-motion estimation problem. In order to do so there is the need to stack many stages of successive filter banks [24], whose coefficients are learned using unsupervised or supervised methods. In this work, we use Convolutional Neural Networks (CNN), applying them to dense optical flow, to learn new visual features at the same time with a non-parametric motion estimator. The proposed CNN architectures outperform the state-of-the-art F2F estimation approaches and guarantee robustness with respect to image anomalies (*e.g.*, blur, contrast and luminance).

C. Contribution and Overview

Our contribution summarizes to this points:

A) We explore feature selection for ego-motion estimation using different CNN architectures. The CNN architectures are used to extract new input features starting from dense optical flow (Figure 2). Three different architectures are proposed: two of them investigate the influence of global and local optical flow fields with respect to the ego-motion estimation (*i.e.*, considering both the full flow image and its different sub-blocks); the last one combine the advantages of the others in a parallel CNN that exploits both global and local information.

B) We show that the presented learned estimators are able to estimate motion outperforming other SotA geometrical and learned methods. In addition the proposed methods are able to use global information to extract camera motion and scale information, while dealing with noise in input.

C) Finally, we show the performances of the presented method in difficult scenarios, using images with very different contrast and blur parameters, to show the robustness of the new features extracted by the CNN.

It is important to note that, in this work, we propose to improve specifically F2F estimation performance. Hence, to provide a precise and in-depth discussion about this fundamental block of any VO approach, we do not consider bundle adjustment or, in general, procedures that refine the motion estimates. However, our approach can be easily embedded into a full key-frame based VO system to reach even better performances.

This work proceeds as follows: Section III describes the foundations of CNNs and the proposed networks architecture; Section IV presents the experimental set-up, the dataset and the performance parameters used; finally, Section V draws the conclusions and the path of future work.

III. Network Structure

In the following, we first describe how we extract dense optical flow information and then we discuss the CNN network architectures used to learn the F2F estimation models.

A. Dense Optical Flow

The input to our network is a *dense* optical flow (OF) extracted with Brox algorithm [17]. The Brox strategy computes the optical flow between two images at time t and $t+1$ using a variational formulation that penalizes the total variation of the flow field by minimizing the following energy function:

$$V(u,v) = V_d + \alpha V_s \tag{1}$$

with

$$\begin{aligned} V_d(u,v) &= \int_\Omega \Phi(|I(\mathbf{x}+\mathbf{w}) - I(\mathbf{x})|^2 \\ &\quad + \gamma|\nabla I(\mathbf{x}+\mathbf{w}) - \nabla I(\mathbf{x})|^2)\mathbf{dx} \\ V_s(u,v) &= \int_\Omega \Phi(|\nabla_3 u|^2 + |\nabla_3 v|^2)\mathbf{dx} \end{aligned}$$

and where $I : \Omega \subset \mathbb{R}^3 \to \mathbb{R}$ denotes a rectangular image sequence, $\mathbf{w} := (u,v,1)^T$, Φ is a concave function (as described by [17]) and α and γ are weighting parameters.

The computed flow field is then quantized in the common RGB encoding, as shown in Figure 2(b), so the input data is a 3 channel, 8-bit depth image.

B. Proposed Network Architecture and Training Procedure

In object recognition and people detection tasks the input images are smaller than the ones typically used in VO. Simply applying one of the already proposed architectures is not straightforward. Down-sampling the image could discard important information for motion estimate. For this reason, we tested three different architectures and compared their performances:

1) **CNN-1b VO**: As a basic exploratory approach we train a deep network on the entire OF image after down-sampling it 8 times with average pooling to reach a dimension of 155×48.

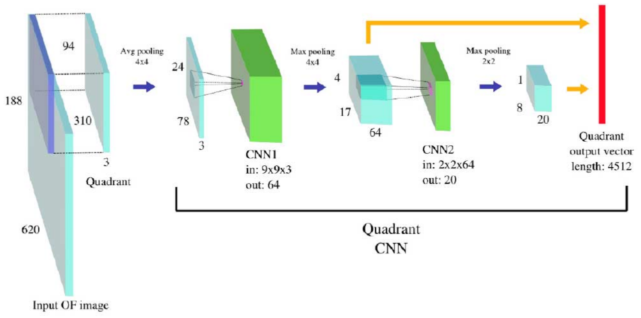

Fig. 3. Quadrant CNN architecture. The image is divided into four quadrants and each one passes through a chain of filter banks (CNN1-pooling, CNN2-pooling). To produce stronger visual features we concatenate the output of CNN1 and CNN2.

2) **CNN-4b VO**: The first alternative configuration tries to exploit local information. We divide the OF image into four sub-images. Each quadrant is down-sampled 4 times and then passed through a series of CNN filters analogous to CNN-1b ones. The final layer is trained to use the output of the four CNN networks to give a global F2F estimate.
3) **P-CNN VO**: The last architecture uses the CNN filters of both CNN-1b and CNN-4b feeding their output to a fully connected network. We do so to explore the performances of a network that merges the global information of CNN-1b with the local information of CNN-4b.

Our hypothesis for P-CNN is that the information of the two other networks is partially different, so can be combined to train a better estimator. In Section IV, we will show that this hypothesis is verified by our experiments. We start the description of these architectures from CNN-4b, since the structure of CNN-1b can be described in terms of the first one, and that P-CNN is the composition of the two.

The architecture of CNN-4b network is shown in Figure 3. The first section of the network is composed of four branches, identical in complexity, but trained separately, that perform the first two convolutional steps (CNN1 and CNN2). Note that each of the four quadrants of the image contains some motion information to compute a motion estimate, with ambiguity between simple turns and forward moving motion. We then link the output of the first CNN-pooling pair with the second one. We do so because exploratory experiments on a down-sampled version of the OF images showed that VO estimators using only CNN1 output, or only the cascade of the two CNNs, were both able to learn good estimators, but the VO estimator learned on the concatenation of the two outputs performed better. This result shows that CNN1 and CNN2 extract different information from the OF images. We presume that CNN1 extracts finer details, while the CNN2 extracts coarser ones, and that this information is not completely overlapping. After this stage the four convoluted features are put back together to form an image that contains the global information and thus is able to solve the motion ambiguities with symmetry information. The last layer computes a fully connected network that uses the information of all four quadrants at both resolutions, as shown in the upper part of Figure 4.

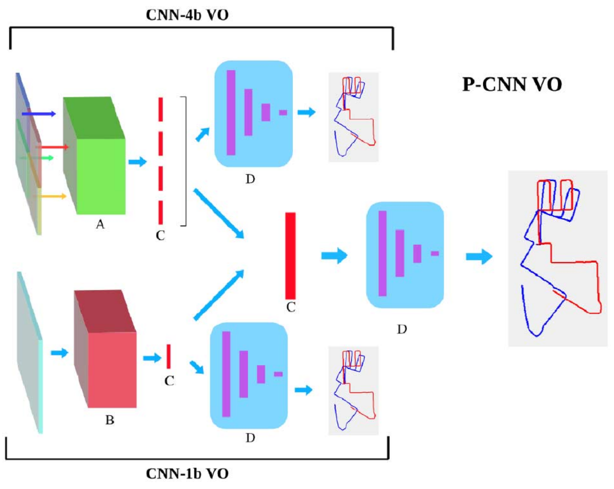

Fig. 4. Global P-CNN network architecture. The capital letters indicate: A) 4xQuadrant CNNs; B) 1-branch CNN; C) Extracted feature vectors of different sizes; D) Fully Connected Networks for estimation. The output of the Convolutional layers of CNN-4b and CNN-1b are concatenated and fed to the FCN of P-CNN.

CNN-1b architecture is similar in shape to a single quadrant of CNN-4b, but with different image dimensions. The initial down-sampling brings the image to a size of 155×48, so the CNN1 and CNN2 layers are wider, but maintaining the number of outputs to 64 and 20 respectively. The max pooling layers are the same. The last architecture, P-CNN, is a composition of the other two networks as shown in Figure 4.

C. Training the Networks

Since it is known that training deep architectures in a global way is an open problem [25], we use the standard *greedy-layerwise* method to find good local minima for the filter coefficients of each layer and then we perform a global training to fine-tune them. More in detail, for each branch we train the CNN1 filter using a fully connected layer next to it to train a first estimator, then we drop the fully connected layers, feed the output of CNN1 to CNN2 and train only this one with a new fully connected estimator. Again we drop the fully connected layers, concatenate the two outputs of the CNNs and train a third estimator. We repeat this procedure for each branch, then we drop the last fully connected layers and concatenate the four quadrant outputs into a last fully connected network that trains the final estimator and fine tunes the CNNs coefficients. The final fully connected layers for the CNN-1b, CNN-4b and P-CNN have two hidden layers with (4500,1000), (600, 2000) and (9000, 3000) nodes, respectively. The whole process requires a few days with a modern GPU or a few hours with a Tesla K40.

D. Estimation Problem Formulation

We want to model a function $\mathbf{f}$ that given the OF of a pair of consecutive $n \times m$ images, is able to output the camera motion that has generated it, filtering out the OF disturbances due to dynamic objects in the scene. The output of the function is the motion vector $\mathbf{y} \in \mathcal{Y} \subset \mathbb{R}^6$ that encodes the displacement

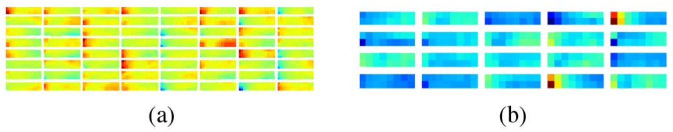

Fig. 5. Application of the convolutional filters to input optical flows. 5(a) and 5(b) show the output of CNN1 and CNN2 when processing the optical flow depicted in Figure 2(b).

of the camera centre and the three euler angles that represent the camera orientation, while the input $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^{n \times m \times 3}$ is the RGB representation of the dense OF $\mathcal{I} \in \mathbb{R}^{n \times m \times 2}$ that is a matrix with OF modulus and phase for each pixel. Thus, our function is defined as $\mathbf{f} : \mathcal{X} \to \mathcal{Y}$.

The first convolutional layer performs K_1 convolutions on each input $\mathbf{x}_i, i = 1 \dots N$ of a motion sequence of length N, producing an output $\mathbf{h}_{i,k} \in \mathcal{Y} \subset \mathbb{R}^{(n-l+1)\times(m-l+1)}$ in this way:

$$\mathbf{h}_{i,k} = \mathbf{W}_k^1 * \mathbf{x}_i + b_k \tag{2}$$

where $\mathbf{W}_k^1 \subset \mathbb{R}^{l \times l \times 3}, k = 1 \dots K_1$ are the filter coefficients, b_k is a bias and $*$ is the convolution operator. The final output of the network is $\mathbf{h}_i \in \mathcal{Y} \subset \mathbb{R}^{(n-l+1)\times(m-l+1)\times K_1}$, that is the composition by the third dimension of every $\mathbf{h}_{i,k}$. This output is like an image slightly smaller in width and height, but with a number of channels equal to the number of convolution filters. After the first CNN1 block we put a max-pooling operation, that is a highly non-linear function that reduces the size of the CNN1 output image by 16 times and selects the maximal response for each non overlapping 4×4 pixel tiles that compose the image. The second convolutional layer performs an analogous operation, but with different filter sizes: $\mathbf{W}_k^2 \subset \mathbb{R}^{q \times q \times 64}, k = 1 \dots K_2$ and 2×2 max-pooling.

The coefficients of $\mathbf{W}_k^1$ and $\mathbf{W}_k^2$ are learned in a supervised greedy layer-wise procedure, as explained in Section III-B, using fully connected NN with rectified-linear activation functions

$$ReLU(\mathbf{x}) = max(0, \mathbf{x}) \tag{3}$$

and using root mean squared loss:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{f}(\mathbf{x}_i) - \mathbf{y}_i\|_2 \tag{4}$$

A sample representation of the effect of the convolutional filters after the training phase is depicted in Figure 5.

IV. Experiments

To evaluate the proposed approach we run experiments on a publicly available dataset. In particular, we compare our ego-motion estimation method based on CNNs (presented in Section III) with different SotA baselines. To further explore the robustness of these architectures, we also perform tests on artificially modified sequences, adding blur and changing contrast and luminance, to simulate adverse recording conditions such as low-light and motion blur.

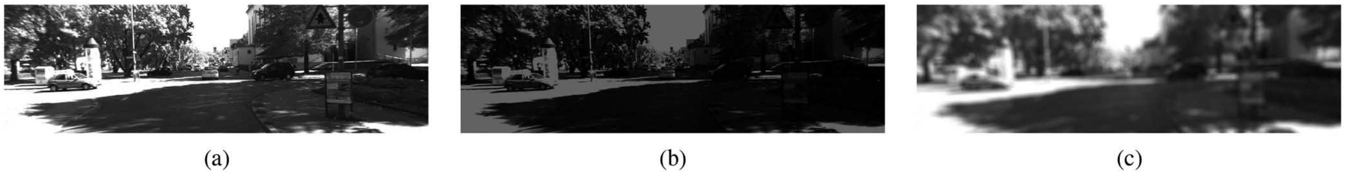

Fig. 6. Example of artificially darkened and blurred images a) standard image b) darkened with max contrast 0.4 and gamma 1.5 c) blurred with Gaussian blur radius of 10 pixels.

In the following, we describe the dataset used to evaluate the performances and the baselines used for comparison. Afterwards, we discuss the results and draw conclusions.

A. Dataset

We test the performances of the CNN-based visual odometry on sequences taken from the KITTI benchmark suite [26], which is a common test bench for many vision algorithms. The sequences are gathered by a car traveling in the streets of the Karlsruhe city equipped with a Pointgrey Flea2 firewire stereo camera. The images are given already undistorted, with a resolution of 1240×386 and a frame rate of 10 Hz (in some sequences the resolution is slightly higher: in these cases we perform a simple crop to uniform all the frames). Each pair of frames is associated with an absolute position with respect to the world reference frame computed with a high precision differential GPS and a Velodyne laser scanner. We used only 11 sequences since the remaining ones are not provided with ground truth. The first 7 are used as training input for the learned methods, described in the following Subsection, and performances of all the estimators are evaluated on the last three sequences 08, 09 and 10. Sequence 08 is filmed in the narrow streets of a peripheral neighbourhood, with some cyclists moving and lots of shadows. Sequence 09 presents a path on a very winding road, with background varying from countryside to suburbs. Sequence 10 presents a paved winding road with high slopes and a number of trucks and vans manoeuvring in front of the camera.

In addition, to test the robustness of the baseline and presented methods, we produced 5 transformed versions of each test sequence. To do so we changed contrast and gamma to simulate different light conditions, and applied Gaussian blur of different radii to simulate defocus or motion blur. We call *darkened 1* the sequence with max contrast 0.4 and gamma 1.5, *darkened 2* the one with max contrast 0.6 and gamma 5.0, *lightened* the sequence with min contrast 0.2, max contrast 0.7 and gamma 0.2, and *blurred s3* and *blurred s10* the sequences with Gaussian blur of radius 3 and 10 pixels respectively. An example of these images is shown in Figure 6.

B. Baselines

We compare the proposed approaches (CNN-1b VO, CNN-4b VO and P-CNN VO) to three different state-of-the-art baselines: a geometric monocular visual odometry, namely VISO2-M, described in [13], a regression model based on Support Vector Machine (SVR VO) as in [27], and a regression model based on standard Neural Network (FCN VO). The

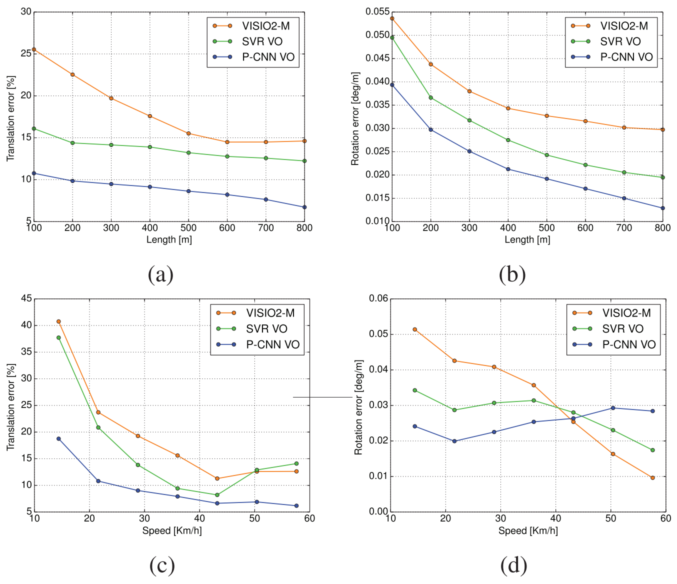

Fig. 7. Average errors across sequence lengths (a and b) and speeds (c and d) on test sequences for baseline and proposed methods.

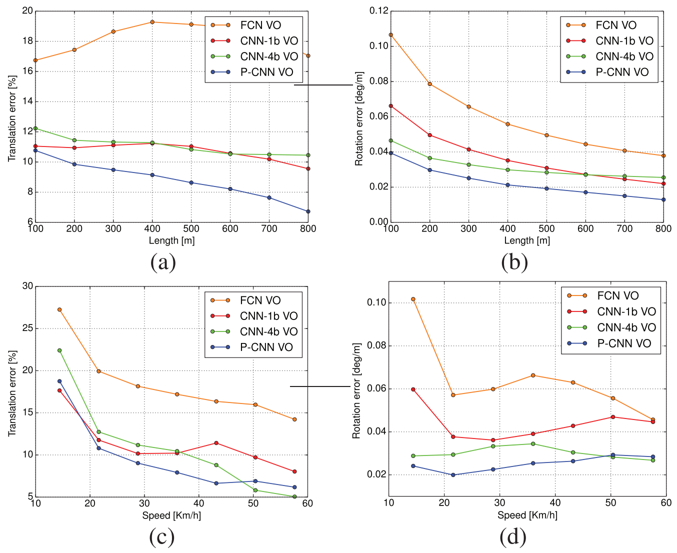

Fig. 8. Average errors across (8(a) and 8(b)) and speeds (8(c) and 8(d)) on test sequences for different network architectures.

implementation of VISO2-M [26] performs frame to frame estimation with some scale recovery using the known distance from the ground plane, but without bundle adjustment and feature tracking on other frames, so it is comparable to our method. The input features used for the training and testing of the SVR

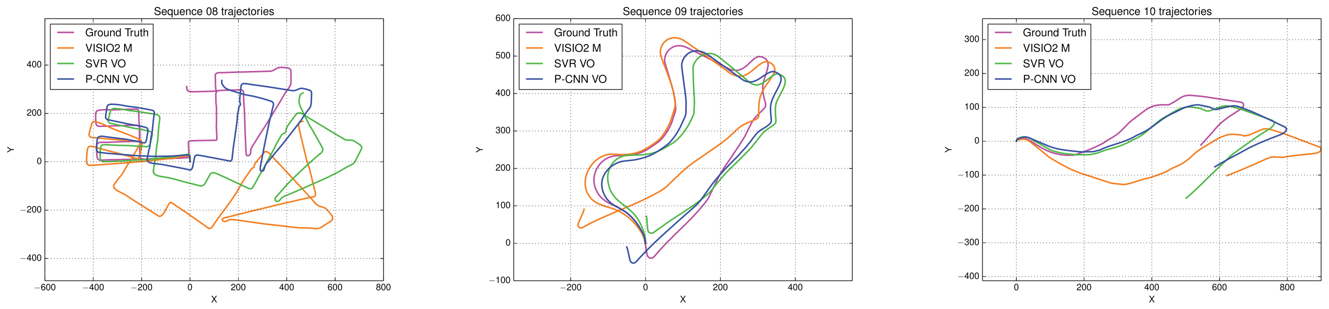

Fig. 9. Comparison of the proposed P-CNN approach with different baseline methods. The P-CNN has better performances almost at every path length and speed, exept for high rotational speeds, where sparse features methods perform better. We explain this fact with the difficulty of dense optical flow to converge to meaningful results when the frame rate (in KITTI is 10Hz) is too low compared to the camera speed.

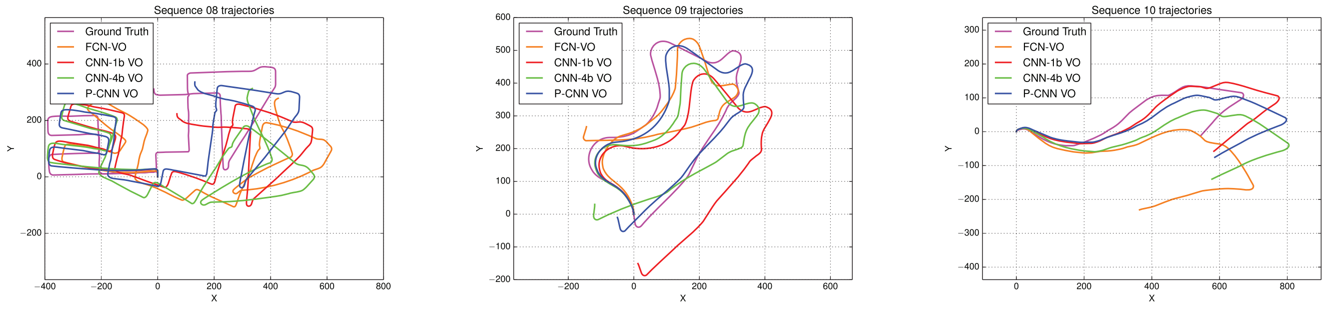

Fig. 10. Trajectories estimated for different network architectures. In this figure, we compare the parallel CNN (P-CNN) with respect to other network configurations.

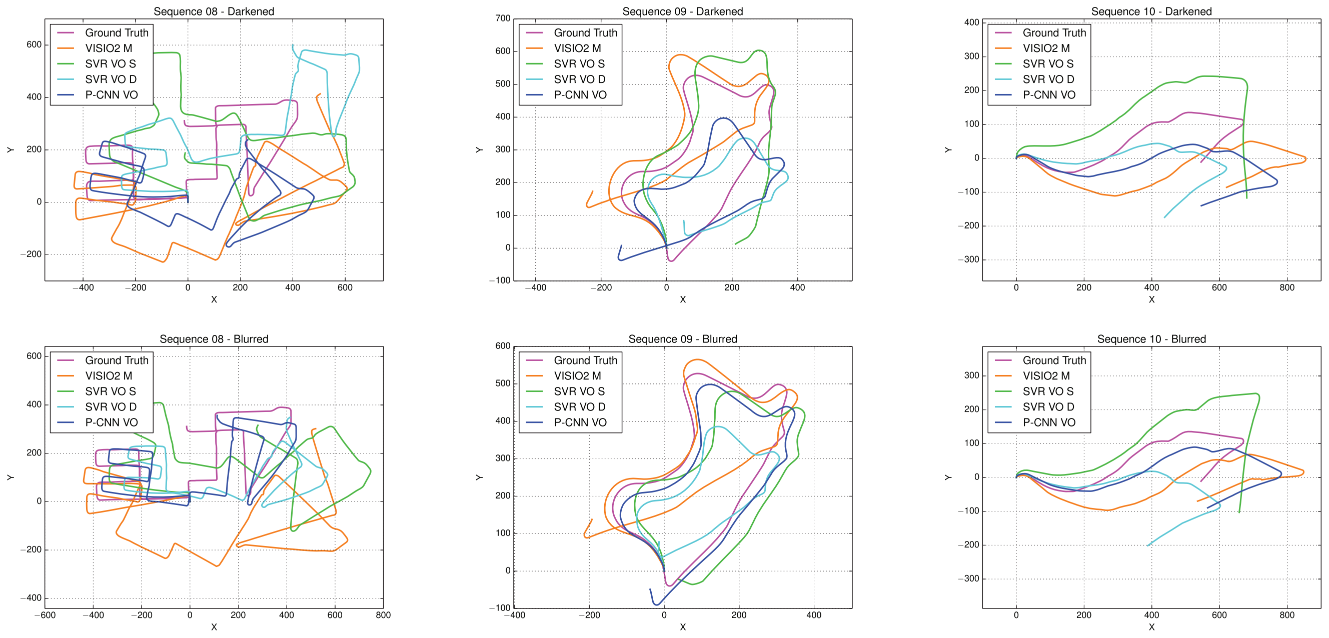

Fig. 11. Trajectories computed with different methods on artificially modified sequences: the first row shows the reconstructed trajectories when images are darkened (0.0-0.4 contrast, 1.5 gamma), while the second row depicts the resulting estimations with respect to blurred sequences (blur radius 3).

VO are computed following the strategy proposed in [27]: optical flows are computed on image pairs and quantized to obtain Histogram of Optical Flow (HOF). The training data, and test code are available on the accompanying web page [28]. The other regression baseline is a Fully Connected Neural Network (FCN VO) that we use to have a direct performance comparison with the CNN+FCN implementations. The training input is the same as for SVR VO. The network is composed by two hidden layers of 1000 nodes and 6 output nodes. SVR VO, FCN VO and the proposed CNN VOs are all trained using the KITTI sequences 00 to 07, and tested on 08 to 10.

The error metric we use to train the models and evaluate the performance is the *Root Mean Square Error* (RMSE) of the difference between predicted and true translations and rotations. We test the estimators using the evaluation code proposed in [26]: for each sequence the performances on different length

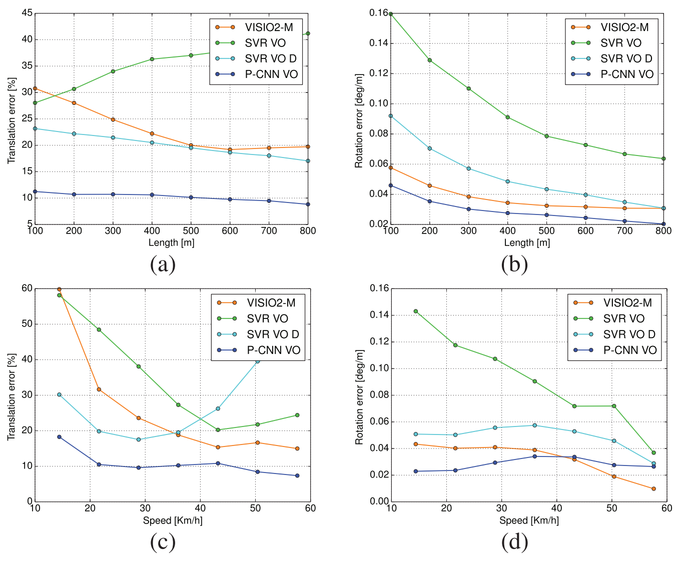

Fig. 12. Average errors across sequence lengths(12(a) and 12(b)) and speeds (12(c) and 12(d)) on artificially darkened sequences for different estimation algorithms (max contrast reduced to 0.4, and gamma increased to 1.5).

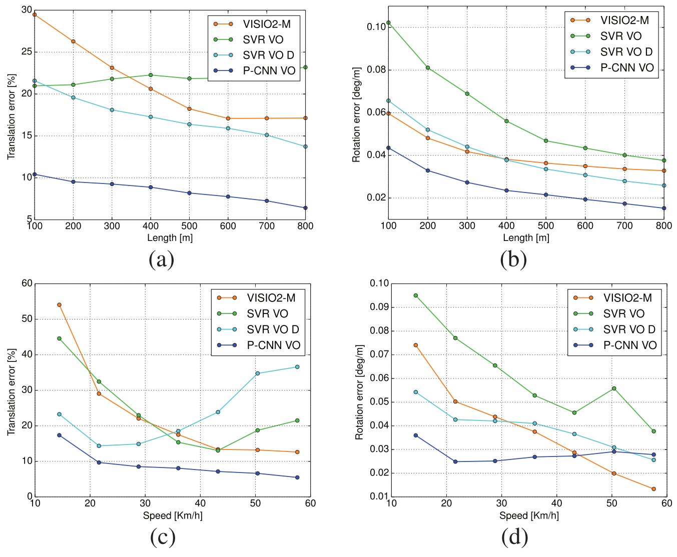

Fig. 13. Average errors across sequence lengths (13(a) and 13(b)) and speeds (13(c) and 13(d)) on artificially blurred sequences for different estimation algorithms (blur radius of 3 pixels). The general behaviour is similar to the one shown in Figure 12.

sub-sequences, ranging from 100 to 800 frames, are computed. Then the average error is computed as the average of the errors divided by the length of the sub-sequence. In addition, the translation and rotation errors corresponding to different speeds are computed. The results of the experiments on the standard sequences are shown in Figures 7 and 8, and the reconstructed trajectories in Figures 9 and 10. Some of the results of the experiments on the darkened and blurred sequences are shown in Figures 11, 12 and 13, while the rest are included in the accompanying web page [28].

C. Discussion

Table I shows the performances of the baseline methods and the proposed P-CNN method on the three unmodified test

TABLE I
COMPARISON OF TRANSLATIONAL AND ROTATIONAL ERRORS BETWEEN THE PROPOSED P-CNN AND THE BASELINES APPROACHES

	VISO2-M		SVR VO		P-CNN VO	
	Trans [%]	Rot [deg/m]	Trans [%]	Rot [deg/m]	Trans [%]	Rot [deg/m]
08	19.39	0.0393	14.44	0.0300	7.60	0.0187
09	9.26	0.0279	8.70	0.0266	6.75	0.0252
10	27.55	0.0409	18.81	0.0265	21.23	0.0405
Avg	18.55	0.0376	13.81	0.0302	8.96	0.0235

TABLE II
COMPARISON OF TRANSLATIONAL AND ROTATIONAL ERRORS BETWEEN DIFFERENT CNN ARCHITECTURES

	CNN1b VO		CNN4b VO		P-CNN VO	
	Trans [%]	Rot [deg/m]	Trans [%]	Rot [deg/m]	Trans [%]	Rot [deg/m]
08	9.42	0.0364	9.91	0.0301	7.60	0.0187
09	11.33	0.0422	8.83	0.0286	6.75	0.0252
10	17.57	0.0319	23.45	0.0467	21.23	0.0405
Avg	10.77	0.0389	11.14	0.0324	8.96	0.0235

sequences. Results show that these sequences have different characteristics, and some are more challenging than others. However, in all the experiments we can observe that the SVR VO and the proposed P-CNN VO outperform VISO2-M, a SotA geometric F2F estimator. We omit the detailed results of the FCN VO because they under-perform all the other methods (Avg. transl. error 18.17%, Avg. rot. error 0.0626 deg/m), but its performances can be observed from Figures 8 and 10.

P-CNN performs better than SVM VO, except in sequence 10. However, the average errors of P-CNN VO are much lower than the other two methods. In Figure 7 the error contributions to the average errors divided per sequence length and speed range are depicted. From this figure, we see that P-CNN has a consistently lower error for each length and speed, except for rotation errors at high speeds, shown in Figure 7(d), where the trend is inverted. We explain this fact observing that rotations at more than 40Km/h are rare in the KITTI dataset, and the learning algorithms have few training examples to learn this behaviour.

In Table II the performances of different network architectures are shown. We compare CNN-1b, CNN-4b and the parallel combination of the two. P-CNN outperforms all the other architectures, except again for sequence 10. The fact that P-CNN has average errors much lower than the two single networks that compose it validates our hypothesis that CNN-1b and CNN-4b extract different information from the images. Thus, their combination in P-CNN is better than the performance of each one. In Figures 9 and 10, the trajectories of each method are shown for qualitative comparison.

Performance on darkened and blurred sequences show that P-CNN performs almost always better than the baseline methods in these scenarios. In the comparison we added an SVR VO using dense optical flow to explore the differences in performances with the features learned by the CNNs. From Figure 12(c) and 13(c) it is interesting to note that the SVR performances with dense optical flow degrades with higher translational speeds, while the P-CNN are always good. This result suggests that the increase in robustness is due to the features learned by the CNN, and not to the dense OF *per se*.

TABLE III
COMPUTATIONAL TIME COMPARISON BETWEEN THE PROPOSED P-CNN ARCHITECTURE AND THE BASELINE METHODS. THE TABLE SHOWS THE TIME REQUIRED TO COMPUTE A SINGLE F2F ESTIMATION, AVERAGED WITH RESPECT TO EACH SEQUENCE

	08 [s/img]		09 [s/img]		10 [s/img]	
	Standard	Blurred s10	Standard	Blurred s10	Standard	Blurred s10
VISIO2-M	0.0634	0.054	0.0682	0.0524	0.0697	0.0157
SVR VO Dense	0.1118	0.1123	0.1142	0.1113	0.113	0.1112
P-CNN VO (CPU)	0.311	0.299	0.305	0.301	0.309	0.307
P-CNN VO (GPU)	**0.051**	**0.049**	**0.041**	**0.045**	**0.052**	**0.0504**

Finally, we compare the computational costs between our P-CNN approach and the baseline methods (*i.e.*, VISIO2 and SVR VO Dense). The performance of VISIO2, SVR VO Dense and P-CNN (CPU) are evaluated using a i7-4720HQ 2.60 GHz processor, while an NVIDIA Tesla K40 GPU is used to run P-CNN (GPU). Table III shows that our approach takes 50 ms on average (30 ms to compute the optical flow and 20 ms to perform the CNN prediction) and, thus, it is well-suited for most real-time applications. As a final remark, VISIO2 runs at 0.0157 seconds per image on the blurred version of sequence 10 because it completely fails to extract features and perform the F2F estimation.

V. CONCLUSION AND FUTURE WORK

In this letter, we explored the architecture and performances of an ego-motion estimation approach based on Convolutional Neural Networks. We showed that this powerful learning paradigm is able to learn both new visual features and a high performing estimation model to achieve robust ego-motion estimation. We studied three different network architectures, comparing them with state-of-the-art ego-motion estimation methods on publicly available sequences. These experiments showed that all these architectures were able to autonomously select a new data representation that allowed the learned estimators to outperform other methods. We also tested the estimators on artificially degraded sequences and showed that the new features learned by the CNNs make the estimation pipeline more robust. We find this behaviour very promising, and future work will explore the performances of the integration of CNNs feature extractors with other direct and semi-direct SotA VO estimators. In addition, our experiments suggest that deep networks are very promising in general for VO estimators learning, and we will conduct further research to experiment with deeper networks and different architectures. In addition, since in this work we focused on F2F estimation, future work will also explore the improvements that can be achieved by integrating strategies such as bundle adjustment, scale estimation, or loop closing.

REFERENCES

[1] H. Strasdat, J. M. M. Montiel, and A. Davison, "Scale drift-aware large scale monocular slam," in *Proc. Robot. Sci. Syst. (RSS)*, Jun. 2010.

[2] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2014, pp. 834–849.

[3] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Jun. 2014, pp. 15–22.

[4] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Nov. 2011, pp. 2320–2327.

[5] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE Robot. Autom. Mag.*, vol. 18, no. 4, pp. 80–92, Dec. 2011.

[6] S. Song, M. Chandraker, and C. Guest, "Parallel, real-time monocular visual odometry," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2013, pp. 4698–4705.

[7] T. A. Ciarfuglia, G. Costante, P. Valigi, and E. Ricci, "A discriminative approach for appearance based loop closing," in *Proc. IEEE Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2012, pp. 3837–3843.

[8] V. Guizilini and F. Ramos, "Semi-parametric learning for visual odometry," *Int. J. Robot. Res.*, vol. 32, no. 5, pp. 526–546, Apr. 2013.

[9] V. Guizilini and F. Ramos, "Semi-parametric models for visual odometry," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2012, pp. 3482–3489.

[10] Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.

[11] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, vol. 1, Jul. 2004, pp. 652–659.

[12] P. F. Alcantarilla, J. Yebes, J. Almazán, and L. M. Bergasa, "On combining visual slam and dense scene flow to increase the robustness of localization and mapping in dynamic environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2012, pp. 1290–1297.

[13] A. Geiger, J. Ziegler, and C. Stiller, "Stereoscan: Dense 3D reconstruction in real-time," in *Proc. IEEE Intell. Veh. Symp. (IV)*, Jun. 2011, pp. 963–968.

[14] A. Davison, I. Reid, N. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1052–1067, Jun. 2007.

[15] E. Eade and T. Drummond, "Scalable monocular slam," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, Jun. 2006, pp. 469–476.

[16] G. Nützi, S. Weiss, D. Scaramuzza, and R. Siegwart, "Fusion of IMU and vision for absolute scale estimation in monocular slam," *J. Intell. Robot. Syst.*, vol. 61, nos. 1–4, pp. 287–299, 2011.

[17] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, May 2004, pp. 25–36.

[18] R. Roberts, H. Nguyen, N. Krishnamurthi, and T. R. Balch, "Memory-based learning for visual odometry," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2008, pp. 47–52.

[19] R. Roberts, C. Potthast, and F. Dellaert, "Learning general optical flow subspaces for egomotion estimation and detection of motion anomalies," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, Jun. 2009, pp. 57–64.

[20] V. Guizilini and F. Ramos, "Visual odometry learning for unmanned aerial vehicles," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2011, pp. 6213–6220.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, Dec. 2012, pp. 1097–1105.

[22] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, Jun. 2014, pp. 580–587.

[23] X. Zeng, W. Ouyang, M. Wang, and X. Wang, "Deep learning of scene-specific classifier for pedestrian detection," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2014, pp. 472–487.

[24] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Sep. 2009, pp. 2146–2153.

[25] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Analysis Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[26] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. Comput. Vis. Pattern Recog. (CVPR)*, Jun. 2012, pp. 3354–3361.

[27] T. A. Ciarfuglia, G. Costante, P. Valigi, and E. Ricci, "Evaluation of non-geometric methods for visual odometry," *Robot. Autonom. Syst.*, vol. 62, no. 12, pp. 1717–1730, 2014.

[28] G. Costante, M. Mancini, P. Valigi, and T. A. Ciarfuglia. (2015). *Extra Marterials for This Work* [Online]. Available: http://www.sira.diei.unipg.it/supplementary/Deep_VO/extra.html.