Learning visual odometry with a convolutional network

Kishore Konda[1], Roland Memisevic[2]
[1]*Goethe University Frankfurt*
[2]*University of Montreal*
konda.kishorereddy@gmail.com, roland.memisevic@gmail.com

Keywords: Visual odometry, Convolutional networks, Motion, Stereo

Abstract: We present an approach to predicting velocity and direction changes from visual information (”visual odometry”) using an end-to-end, deep learning-based architecture. The architecture uses a single type of computational module and learning rule to extract visual motion, depth, and finally odometry information from the raw data. Representations of depth and motion are extracted by detecting synchrony across time and stereo channels using network layers with multiplicative interactions. The extracted representations are turned into information about changes in velocity and direction using a convolutional neural network. Preliminary results show that the architecture is capable of learning the resulting mapping from video to egomotion.

1 INTRODUCTION

Visual odometry is a fascinating area of research with a wide range of applications in robotics, navigation and many other applications. The visual odometry basically involves estimation of egomotion from visual information, such as a sequence of frames from one or more cameras. In the past few years many interesting and efficient approaches like (Badino et al., 2013; Nistér et al., 2004; Chandraker et al., 2013; Kitt et al., 2010) were introduced to deal with the visual odometry problem. Some of the approaches, like (Chandraker et al., 2013), only rely on monocular video, whereas others, like (Badino et al., 2013; Nistér et al., 2004), use stereo information. A common property across most of these works is that they rely on key-point detection and tracking, combined with camera geometry, for estimating visual odometry.

In recent years learning based approaches have shown promising results in many areas of computer vision. Models like convolutional networks were proven to be very effective across a range of vision tasks, like classification and localization (Krizhevsky et al., 2012), depth estimation (Eigen et al., 2014) and many more. Unsupervised feature learning models like (Memisevic, 2011; Konda et al., 2014) demonstrated the ability to learn representations of local transformations from the data via multiplicative interactions. In this work we show how an end-to-end learning based approach based on motion and depth representations can be used to perform visual odometry. We present a preliminary approach for relating learned motion and depth representation to visual odometry information like ’change in direction’ and ’velocity’ using a convolutional network. To our knowledge this work is the first to propose a deep learning based architecture for visual odometry.

Section 2 briefly explains the *synchrony condition* which is the basis for the unsupervised learning model explained in section 3. In the section that follows details on the learning model parameters and convolutional architecture are presented. Finally, Section 5 demonstrates the inference procedure and results from path prediction task.

2 Motion and depth estimation by synchrony detection

A simple approach to learning motion representations by relating frames in a video is presented in (Konda et al., 2014). It shows that detection of a spatial transformation can be viewed as the detection of synchrony between the image sequence and a sequence of features undergoing that transformation called **synchrony detection**. The paper also presents a simple way to detect synchrony by allowing for multiplicative (gating) interactions between filter re-

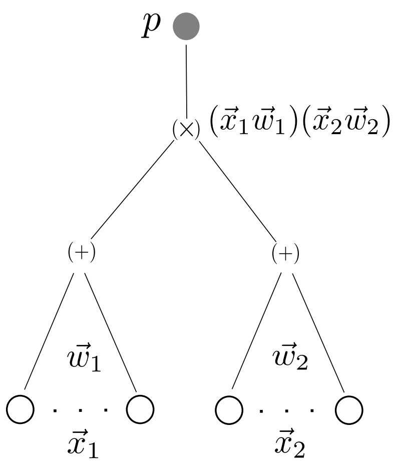


Figure 1: Synchrony detection using multiplicative interaction.

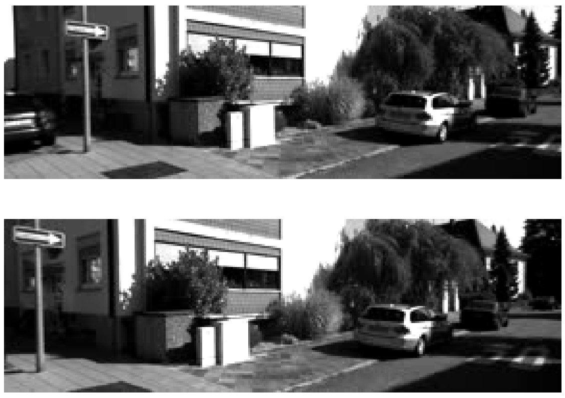


Figure 2: Two subsequent frames from a sequence involving turning action.

sponses as shown in Figure 1. An extension to learning depth by detection of spatial transformation across stereo pairs using synchrony detection was presented in (Konda and Memisevic, 2013). That paper also showed how a combined representation for depth and motion can be learned using a similar architecture. In this work, we show how the joint representation of depth and motion learned using the models presented in (Konda and Memisevic, 2013) can be used to estimate visual odometry using stereo videos sequences as input.

2.1 Why depth?

One can argue that using only local motion information might be sufficient for estimation of visual odometry. In this section we explain how depth information can be used to resolve some of the issues from only using motion information. Figure 2 shows two frames involving change in direction of motion. It shows that the objects that are close the camera undergo a larger shift across the frame as compared to the objects which are far. Motion information alone cannot determine the change in direction since two different motion patterns can represent the same amount of change in direction in the two different cases. In other words, depending on the distance of an object from the camera, the same actual motion can yield a small or a large displacement of the pixels representing that object. Using depth information allows us to disambiguate these cases.

3 Unsupervised learning model

For pre-training a representation of depth and motion we employ the synchrony/depth autoencoder (SAE-D) presented in (Konda and Memisevic, 2013). In contrast to the classic motion energy model, the synchrony-based SAE-D approach is a single-layer module that allows for local, Hebbian-type learning to extract features from video data. Let $\vec{X},\vec{Y} \in \mathbb{R}^N$ be the concatenation of T vectorized frames $\vec{x}_t,\vec{y}_t \in \mathbb{R}^M, t = 1,\ldots,T$, and be defined such that $(\vec{x}_t,\vec{y}_t)$ are stereo image pairs. Let $\mathbf{W}^x,\mathbf{W}^y \in \mathbb{R}^{Q\times N}$ denote matrices containing Q feature vector pairs $\vec{W}^x_q,\vec{W}^y_q \in \mathbb{R}^N$ stacked row-wise.

Encoding: The filter responses are defined as $\vec{F}^X = \mathbf{W}^x\vec{X}$ and $\vec{F}^Y = \mathbf{W}^y\vec{Y}$ corresponding to the sequences $\vec{X}$ and $\vec{Y}$. A simple representation of motion and depth is given by:

$$H_q = \sigma(F^x_q \odot F^y_q) \tag{1}$$

We use the 'truncated Relu' non-linearity from (Memisevic et al., 2014) as σ, which does not require any regularization during training.

Decoding: Reconstructions of the inputs are given by:

$$\hat{X} = (\mathbf{W}^x)^{\mathrm{T}}(\vec{H} \odot \vec{F}^Y) \tag{2}$$

$$\hat{Y} = (\mathbf{W}^y)^{\mathrm{T}}(\vec{H} \odot \vec{F}^X) \tag{3}$$

Learning: Training data for the model is a set of stereo sequence pairs $\vec{X},\vec{Y} \in \mathbb{R}^N$. Just as in a standard autoencoder, the SAE-D model is trained to minimize the reconstruction error given by

$$L((\vec{X},\vec{Y}),(\hat{\vec{X}},\hat{\vec{Y}})) = \|(\vec{X}-\hat{\vec{X}})\|^2 + \|(\vec{Y}-\hat{\vec{Y}})\|^2 \tag{4}$$

The model is trained with stochastic gradient descent.

4 Supervised learning using convolutional neural network

Convolutional Neural Networks (CNNs) have been established as a powerful class of models for a variety of tasks including classification and regression. In this work a CNN is trained to relate local depth and motion representations to local changes in velocity and direction, thereby learning to perform visual odometry. Features learned using the SAE-D model (see previous section) are used to initialize the deep convolutional network. We observed that training the CNN without using features from unsupervised pre-training resulted in noisy filters and overfitting on the training data.

5 Experiments

To evaluate our approach for learning visual odometry we choose the task of path prediction by estimating local velocity and change in direction. As explained in earlier sections this is done by learning to relate local motion and depth representations with the desired information. For this we chose the odometry dataset from the KITTI Vision benchmark (Geiger et al., 2012).

5.1 Dataset

The odometry dataset from the KITTI Vision benchmark (Geiger et al., 2012) consists of stereo sequences collected while driving a vehicle outdoors. A total of 21 sequences are provided out of which 11 sequences are with ground truth trajectories for training and 11 sequences for evaluation without any ground truth. The ground truth information is provided in terms of a 3×4 transformation matrix which projects the i-th coordinate system into 0-th coordinate system. We define the local change in direction as the change in orientation of the vehicle around the axis perpendicular to the direction of motion over a set of subsequent frames. The velocity is defined as the distance traversed by the vehicle in a frame interval. The original resolution of the videos is 1241×376. The videos are down-sampled to 300×100 so as to accommodate the local horizontal shift across the stereo pairs with in a local patch. In our experiments a frame interval or sub-sequence of 5 frames is used to estimate the discretised velocity and change in direction labels. We leave out 3 of the given training sequences $(8, 9$ and $10)$ so as to compare our approach to ground truth.

5.2 Learning

The SAE-D model is trained on local stereo block pairs each of size $16 \times 16 \times 5$ $(space \times space \times time)$ cropped randomly from the training sequences. The

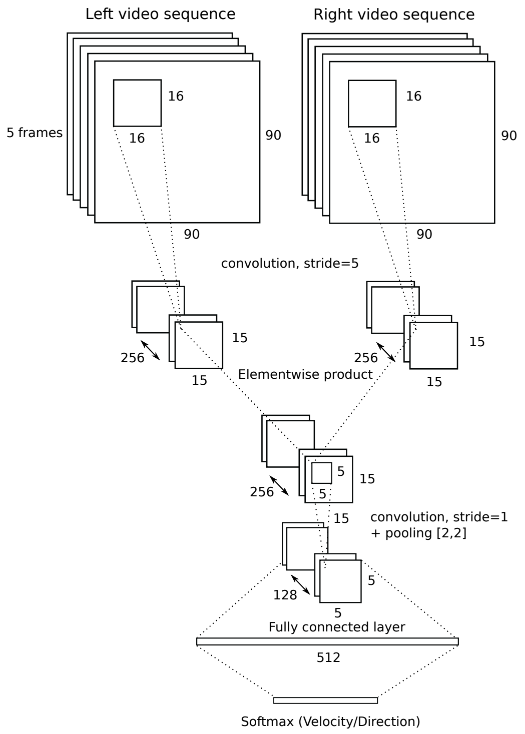


Figure 4: Architecture of the CNN used in this work

total number of training samples is $500,000$ which are dimensionally reduced using PCA. The filters learned by the model are visualized in figure 3. It can be observed that learning results in phase shifting gabor features which are well-suited to representing local translations. In the same figure a small phase shift can be observed in corresponding features both across time (column) and also across stereo pair (row).

The features learned using the SAE-D model are used to initialize the first layer of a CNN with the architecture shown in Figure 4. Before the learned features are used to initialize the CNN they are dewhitened to get back to image space. The output of the first layer of the CNN can be interpreted as a combined representation of local motion and depth. The later part of the CNN relates the local representations to the desired label (change in direction/velocity).

Two different networks with the same architecture as in Figure 4 are trained, one for prediction of velocity and one for prediction of local change in direction. As mentioned above the filters of the first convolutional layer of both networks are initialized with features obtained from unsupervised learning. The input to each network is a 5 frame sub-sequence and the target is a vector representation of discretised velocities and direction changes. We also tried replacing the softmax layer of the CNN with a linear regression layer so as to predict real valued velocities and change

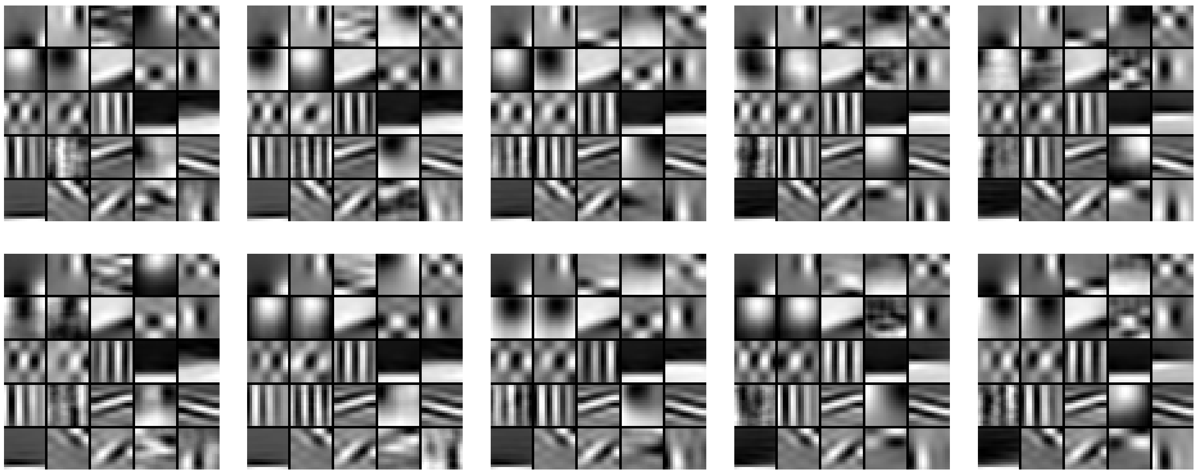


Figure 3: Filters learned from stereo sequences. **Row 1:** Frames 1-5 of the learned filters. **Row 2:** Corresponding stereo pairs of filters in Row 1.

in directions. Due to lack of large amounts of data, especially for the change in directions, linear regression was unable to learn a good prediction model.

6 Path prediction

Using the CNN networks described in the previous section, the velocity and change in direction for each 5-frame sub-sequence of the entire sequence is predicted. A flow diagram of the path prediction process is visualized in figure 5. Using predicted velocity and change in direction information the path of the complete sequence can be recovered. Figure 6 shows two of the predicted paths along with corresponding original path and the path computed from discretised ground truth. From the predicted paths it can be observed that our approach is able to predict most of the change in directions and velocities accurately. As a result of error accumulation one can also observe increasing error in displacement from starting point to the end of a path.

6.1 Computational efficiency

We used a GPU based implementation of both the unsupervised learning model and the convolutional network for better computational efficiency. For the GPU implementations, we used the theano library (Bergstra et al., 2010). We calculated the inference times for the path prediction task by computing total time taken for path prediction of each sequence divided by the total number of frames in that sequence. Average inference times (in seconds/frame) is 0.026. All experiments were performed on a system with a 3.20 GHz CPU, 24 GB RAM and a GTX 680 GPU.

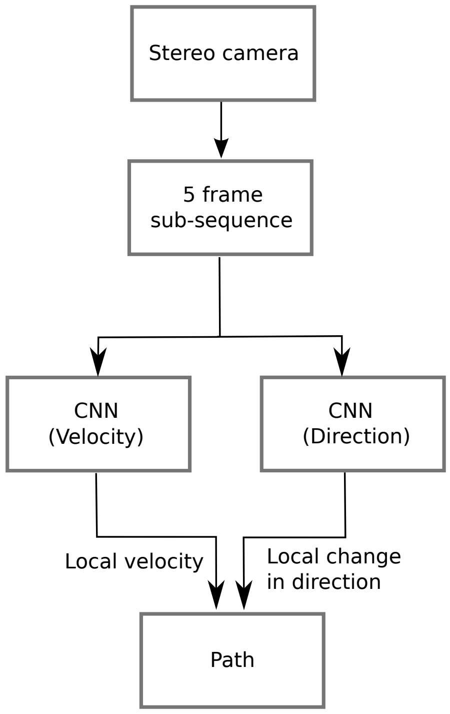
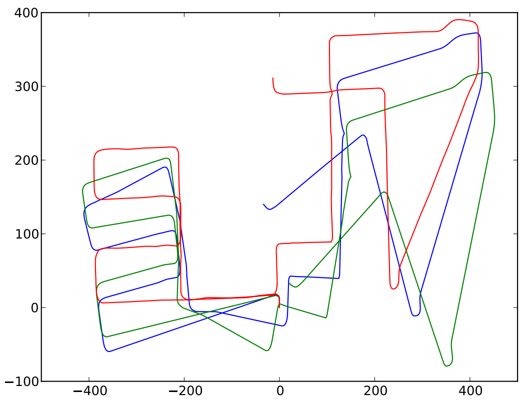


Figure 5: Inference line for path prediction.

7 Conclusion

Based on the preliminary results we can conclude that the architecture is capable of learning the resulting mapping from video to egomotion. In this work we currently do not use any post processing or automatic loop closure techniques for better path predic-


(a) Train sequence (Seq.8)

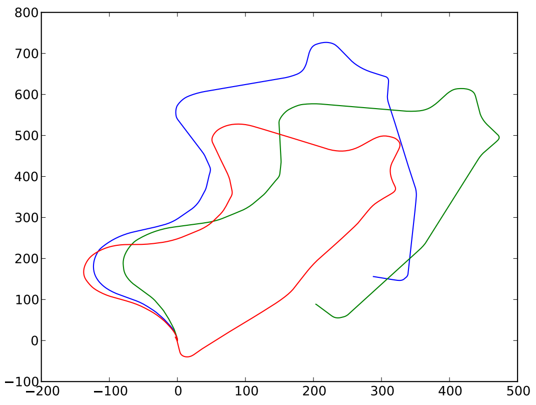

(b) Test sequence (Seq.9)

Figure 6: Original ground truth path is displayed in **Red**, discretised ground truth in **Green** and the estimated one in **Blue**.

tion. In future work we intend to add a convolutional network based landmark detection scheme which will help in automatic loop closure. In the current form the approach cannot be compared to the state-of-the-art approaches for visual odometry in terms of precision. We believe our work is a step towards building a common architecture for many vision tasks like object classification, depth estimation, activity analysis and visual odometry.

Acknowledgements

This work was supported in part by the German Federal Ministry of Education and Research (BMBF) in projects 01GQ0841 (BFNT Frankfurt), by an NSERC Discovery grant and by a Google faculty research award.

REFERENCES

Badino, H., Yamamoto, A., and Kanade, T. (2013). Visual odometry by multi-frame feature integration. In *Computer Vision Workshops (ICCVW), 2013 IEEE International Conference on*, pages 222–229. IEEE.

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. (2010). Theano: a CPU and GPU math expression compiler. In *SciPy*.

Chandraker, M., Reddy, D., Wang, Y., and Ramamoorthi, R. (2013). What object motion reveals about shape with unknown brdf and lighting. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2523–2530. IEEE.

Eigen, D., Puhrsch, C., and Fergus, R. (2014). Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kitt, B., Geiger, A., and Lategahn, H. (2010). Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme. In *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pages 486–492. IEEE.

Konda, K. R. and Memisevic, R. (2013). Unsupervised learning of depth and motion. *CoRR*, abs/1312.3429.

Konda, K. R., Memisevic, R., and Michalski, V. (2014). Learning to encode motion using spatio-temporal synchrony. In *Proceedings of ICLR*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Memisevic, R. (2011). Gradient-based learning of higher-order image features. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1591–1598. IEEE.

Memisevic, R., Konda, K. R., and Krueger, D. (2014). Zero-bias autoencoders and the benefits of co-adapting features. *CoRR*, abs/1402.3337.

Nistér, D., Naroditsky, O., and Bergen, J. (2004). Visual odometry. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–652. IEEE.