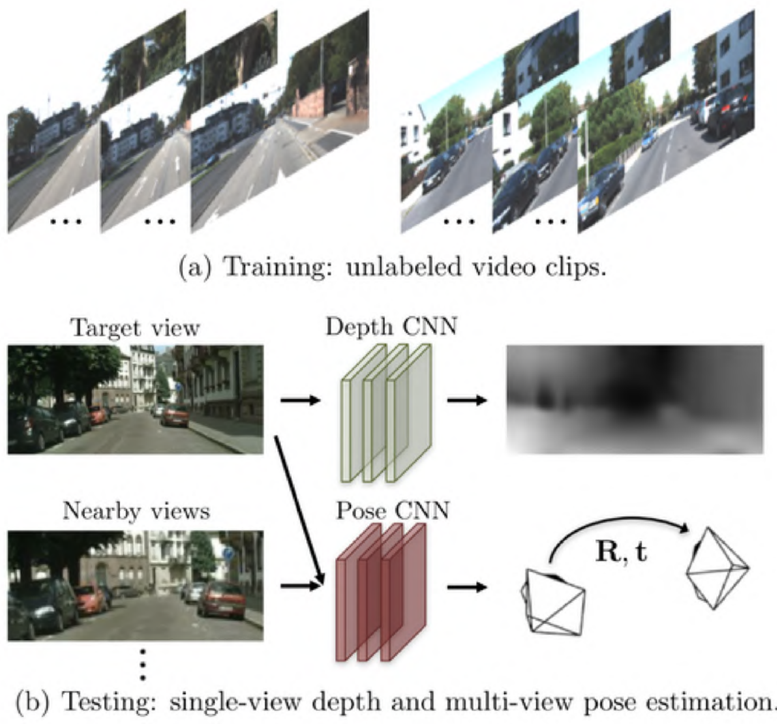
Unsupervised Learning of Vehicle Motion using Image Sequences

AIMS
Autonomous Intelligent Machines & Systems

EPSRC
Pioneering research and skills

Abstract

Recently it has been shown that neural networks are capable of learning monocular depth and ego-motion estimation using unlabelled image sequences. This enables the use of data which is much more accessible for the training of these networks. In this paper we propose an improvement to these networks which considers objects in the scene which move independently of the camera's own motion. This would allow tracking of other objects enabling us learn their motion relative to our own and improve our estimate of our own motion.

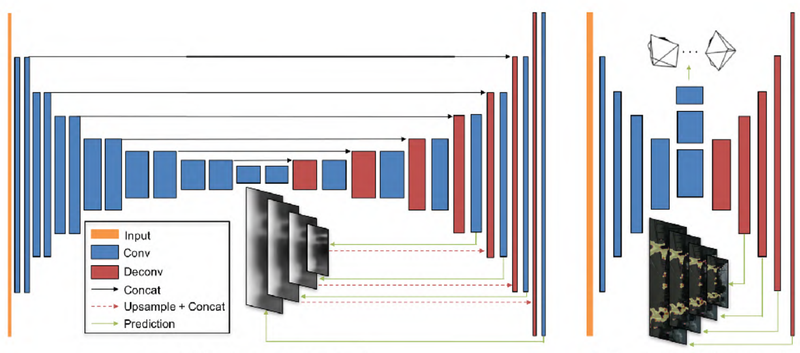


(a) Training: unlabeled video clips.

(b) Testing: single-view depth and multi-view pose estimation.

Over the past few year neural networks have been widely adopted in computer vision for tasks ranging from object classification and semantic segmentation. One of the major issues with these applications is their need for large annotated training sets which can be prohibitively expensive to collect and annotate. On the other hand unsupervised learning can be applied to data without the need to carefully annotate data. Zhou et al. introduced a network that predicts the depth and ego- motion SfMLearner[11]. This network predicted the pose change between three sequential frames and warped the first and last to match the middle, then deriving a loss by comparing the similarity of the warped images to the original central frame. Continuing this work Klodt[6] extended the network to remove the brightness constancy assumption (as pixels on a reflective surface would display differently in sequential images), this is done by generating a photometric uncertainty map which tells us which pixels are unreliable in the warped images. Klodt also used classical SFM algorithms to provide additional supervisory signals to the network.

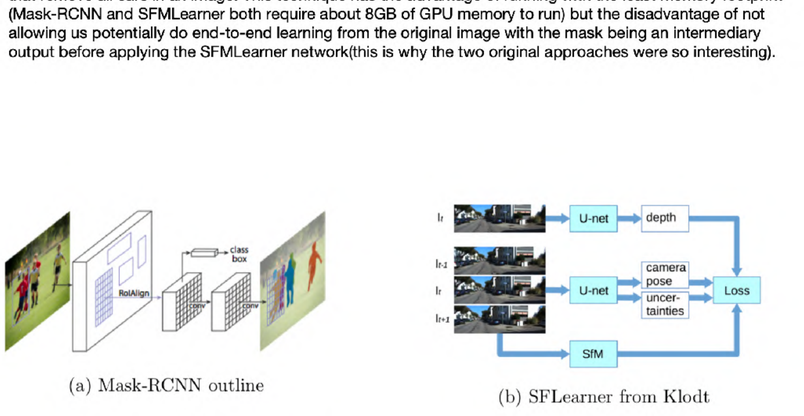
In this paper continuing on from the work of Klodt we try to adapt the network to consider objects in the scene that move independently of the cameras motion. The main dataset used for these networks is Kitti[3] we attempt to segment out cars in images and learn their motion relative to the cameras. This is done using the masks generated by Mask-RCNN using pre-trained weights from imagenet. These masks are fed into the pose network to encourage the network focus on parts of the image moving independently, allowing us to potentially improve camera ego-motion estimation by discarding pixels we believe to be highly likely to move independently.



(a) Single-view depth network

(b) Pose/explainability network

Our initial idea was to run Mask-RCNN[4] on the input images then run the pose network with the generated masks as an additional channel on the input. This caused issues with tensorboard summaries as each input generated multiple output tensors for each output of the network. The next approach to adding Mask-RCNN preprocessing was to change the data loading system to Numpy instead of using Tensorflow input producers. While this worked the images output to Tensorboard had some weird numerical issues resulting in them being largely unrecognisable for some summary values. We then proceeded to store the output masks from Mask-RCNN as images and modified the existing Tensorflow input producer to read each car mask individually resulting in 36037 training examples when we include the masks of individual cars and the background masks that remove all cars in an image. This technique has the advantage of running with the least memory footprint (Mask-RCNN and SFMLearner both require about 8GB of GPU memory to run) but the disadvantage of not allowing us potentially do end-to-end learning from the original image with the mask being an intermediary output before applying the SFMLearner network(this is why the two original approaches were so interesting).

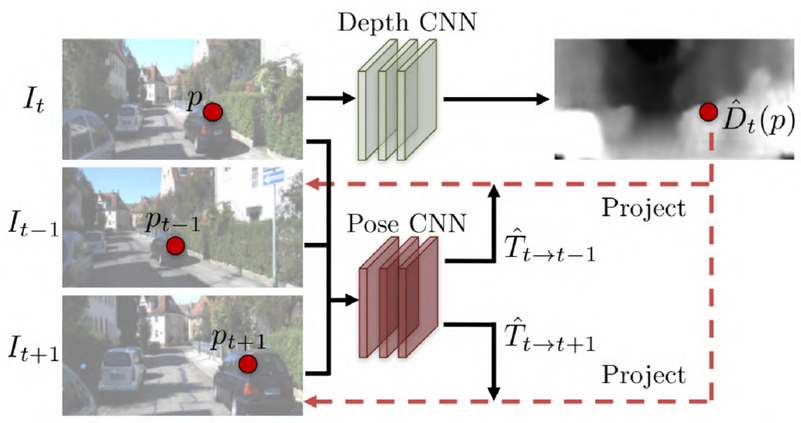


(a) Mask-RCNN outline

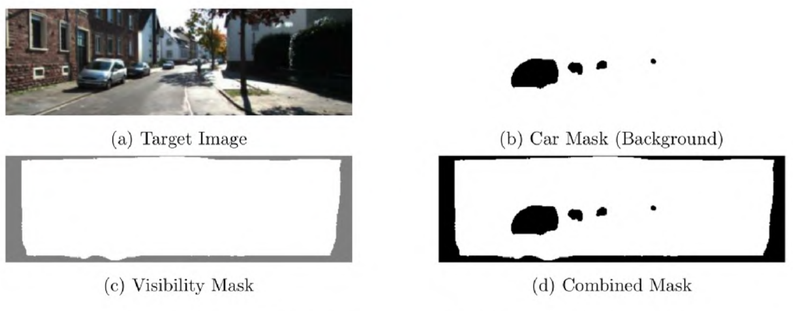
(b) SFLearner from Klodt

Figure 1: Network architectures

Mask-RCNN is broken up into multiple stages: a backbone network that extracts interesting features (typically ResNet50 or ResNet101[5]). To improve the performance of ResNet a Feature Pyramid Network[7] is used which takes high level features and passes them down to lower layers allowing the network extract features at multiple scales. These two parts feed into the Region Proposal Network which scans over the extracted features to find areas that contain objects. The bounding boxes on these objects are then refined and the class objects is determined. These bounding boxes and classes are fed into the mask branch of the network which determines which pixels belong to the object and which are in the background.



SFMLearner works in two stages. First the depth of the target image is calculated using an encoder- decoder structure with skip connections to allow fine-scale details be retained. The Pose network takes the target image and the images before and after (source images) and calculates the change in camera pose between frames. The pose network also outputs pose, photometric and depth uncertainty all of which share an encoder. This gives us an idea of which pixels in the image the network believes can be correctly warped to match the target image(essentially the network attempts to calculate how lambertian the surfaces is). The depth output and pose estimations are then used to warp the source images to match the target image and the error is calculated based off of the closeness of these warped images to the target image. Klodt also utilises SfM supervision where a classical Structure-from-Motion algorithm is used to calculate the pose changes, which then feeds into the loss.



(a) Target Image

(b) Car Mask (Background)

(c) Visibility Mask

(d) Combined Mask

Figure 2: Sample images used during network

Loss Functions

In our network we can evaluate a varying number of source images (images before and after the target/central image that we warp to) and at different scales to improve performance. Before being fed into the network we scale the networks and crop them beck to the original size.

Image Matching

This term evaluates how close the target image and the warped source images are.

$$\sum_{src=1}^{|\text{source images}|}\sum_{sca=1}^{|scales|} w_{abs} \times |\text{projected image}_{src,sca} - \text{target image}_{sca}| + w_{ssim} \times SSIM(\text{target image}_{sca}, \text{projected image}_{src,sca})$$

This combines an L1 loss and SSIM[9] which measures the structural similarity of the target image and the source image which has been warped to the same viewpoint using the predicted pose change from the pose network.

Uncertainty Loss Photo

This loss is applied to the photometric uncertainty loss image that the pose network produces to determine how reliable the warped pixels at a point are (down weighting independently moving and non-lambertian surfaces.

$$\sum_{src=1}^{|\text{source images}|}\sum_{sca=1}^{|scales|} mean(\log(\text{Uncertainty Photo(Masked)})_{src,sca})$$

The uncertainty photo has the image mask of either a car or the background applied before calculation of this loss, allowing us see how well the network believes it can re-project these points.

Pixel Loss

Pixel loss is applied to the current projection error, which depicts how distant the target and warped image are in RGB colour space. This is first masked by the visibility and car/background masks which allows us specify which parts of the image we want our network focus on when calculating pose.

$$\sum_{src=1}^{|\text{source images}|}\sum_{sca=1}^{|scales|} mean(\text{Projection Error(Masked)}_{src,sca})$$

These errors come from the classical SfM supervision, allowing us utilise the predicted values from these techniques in our unsupervised system.

Translation Error:

$$\sum_{src=1}^{|\text{source images}|}\sum_{b=1}^{|batch|} |t_{pred} - t_{actual} \times sfm_{scale}|_{src,b}$$

Rotation Error:

$$\sum_{src=1}^{|\text{source images}|}\sum_{b=1}^{|batch|} \arccos(0.5 \times (\sum_{i=0}^{2} R[i,i]) - 1)_{src,b}$$

SfM Pose Uncertainty Loss:

$$\sum_{src=1}^{|\text{source images}|}\sum_{b=1}^{|batch|} mean(\log(\text{Uncertainty Translation}_{src,b})) + mean(\log(\text{Uncertainty Rotation}_{src,b}))$$

SfM Loss Rotation

$$\sum_{src=1}^{|\text{source images}|}\sum_{b=1}^{|batch|} \frac{\text{Rotation Error}_{src,b}}{\text{Rotation Translation}_{src,b} + \epsilon}$$

SfM Loss Depth

$$\sum_{src=1}^{|\text{source images}|}\sum_{b=1}^{|batch|} mean(|\frac{depth_{pred} - depth_{\text{sfm scaled}}}{depth_{uncertainty}}|(masked))$$

Uncertainty Loss Depth

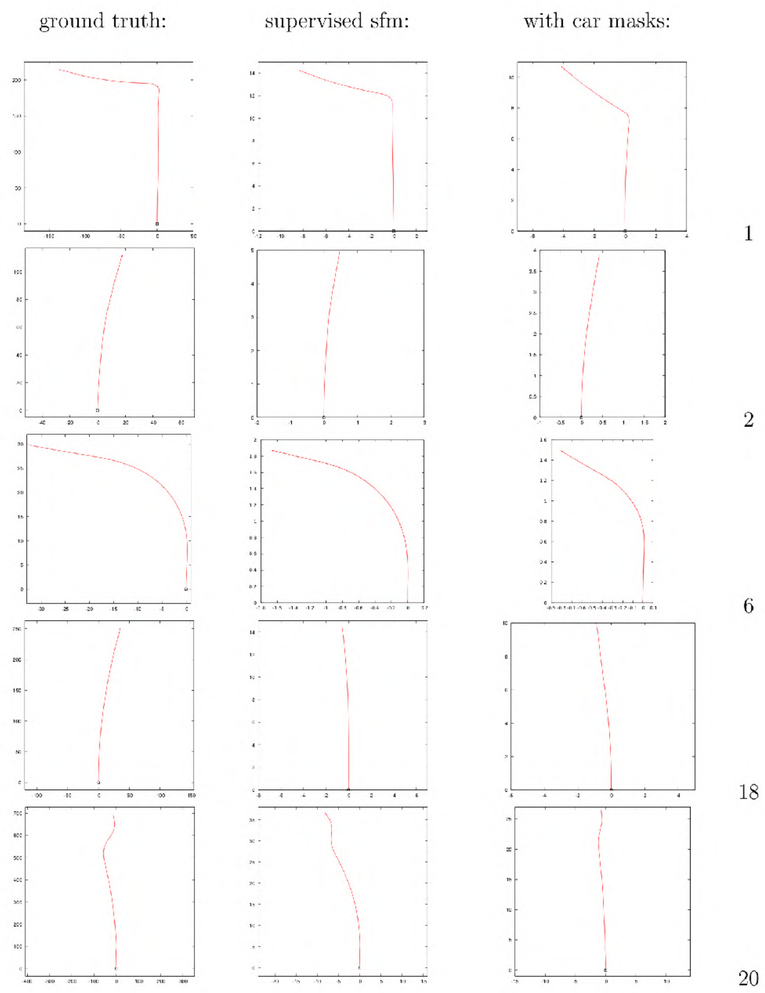
$$\sum_{src=1}^{|\text{source images}|}\sum_{b=1}^{|batch|} mean(\log(\frac{\text{Uncertainty Depth}}{\text{Uncertainty Depth} + \epsilon})(masked))$$

Implementation Details

Reading in of images is handled by Tensorflow[1] queues loading in each image mask separately. This means that the batches we train on don't contain copies of the same input images with a different mask. We use the Adam optimiser with learning rate of 0.0001 and beta of 0.9. Images are evaluated at multiple scales and are scaled and cropped. During training it's possible to increase the number of source images, meaning that pose changes are calculated over more images which is used when comparing to traditional SfM algorithms which require more frames as input. Masks were scaled in value to be in the range [0, 0.05] which is done to match the scaling of the RGB images to the range [-1,1] as not doing this resulted in the masks having no influence in training.

Experiments

We trained on the Kitti raw dataset with pose and depth supervision assisted by ORB-SLAM[8]. First we supply the images with the background mask only, then we train with the background and car masks which allows the parameters be set using inputs similar to the previous works and then fine tuned with the car masks. During the second stage of training the SfM supervision is turned off as this reflects the camera pose changes rather than that of the cars



1

	Error Measures			Accuracy		
	abs. rel.	sq. rel.	RMSE	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
SfMLearner(paper)	0.208	1.768	6.856	0.678	0.885	0.957
SfMLearner(website)	0.183	1.595	6.709	0.734	0.902	0.959
SfMLearner (reproduced)	0.198	2.423	6.950	0.732	0.903	0.957
+ image matching	0.181	2.054	6.771	0.763	0.913	0.963
+ photometric uncertainty	0.18	1.97	6.855	0.765	0.913	0.962
+ pose from SfM	0.17	1.891	6.588	0.776	0.919	0.963
+ pose and depth from SfM	0.166	1.490	5.998	0.778	0.919	0.966
MaskSfMLearner	0.2045	2.3479	7.0043	0.7147	0.8985	0.9584

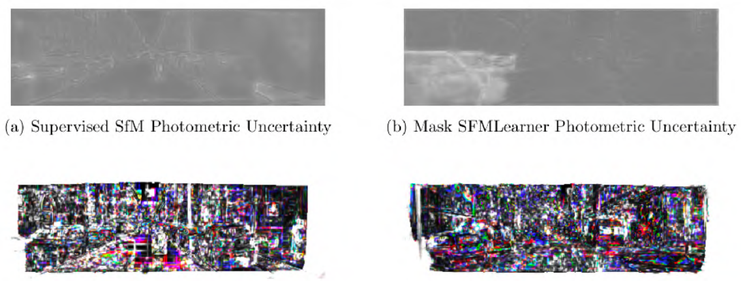
Trajectories

The page above depicts the predicted camera trajectories compared with the ground truth and the supervised SfMLearner over the virtual kitti test sequences. Some sequences show improvement (2 and 20) with others being similar or slightly worse than supervised SfM.

(a) Supervised SfM Depth

(b) Mask SFMLearner Depth

The above images compare the depth images generated by both networks for the source images above. The depth network does not have the car masks as input and is not significantly altered by their addition in the pose network.



(a) Supervised SfM Photometric Uncertainty

(b) Mask SFMLearner Photometric Uncertainty

(a) Supervised SfM Projection Error

(b) Mask SFMLearner Projection Error

References

. [1] Mart´ın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Good- fellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Man´e, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Vi´egas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. URL: https://www.tensorflow.org/.

. [2] Min Bai, Wenjie Luo, Kaustav Kundu, and Raquel Urtasun. Deep semantic matching for optical flow. CoRR, abs/1604.01827, 2016. URL: http://arxiv.org/abs/1604.01827, arXiv: 1604.01827.

. [3] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In Conference on Computer Vision and Pattern Recognition (CVPR), 2012.

. [4] Kaiming He, Georgia Gkioxari, Piotr Doll´ar, and Ross B. Girshick. Mask R-CNN. CoRR, abs/1703.06870, 2017. URL: http://arxiv.org/abs/1703.06870, arXiv:1703.06870.

. [5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. CoRR, abs/1512.03385, 2015. URL: http://arxiv.org/abs/1512.03385, arXiv: 1512.03385.

. [6] M. Klodt and A. Vedaldi. Supervising the new with the old: learning sfm from sfm. 2018(un- published).

. [7] Tsung-Yi Lin, Piotr Doll´ar, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. CoRR, abs/1612.03144, 2016. URL: http://arxiv.org/abs/1612.03144, arXiv:1612.03144.

. [8] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tard´os. ORB-SLAM: a versatile and accurate monocular SLAM system. CoRR, abs/1502.00956, 2015. URL: http://arxiv.org/abs/1502. 00956, arXiv:1502.00956.

. [9] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing, 13(4):600–612, April 2004. doi:10.1109/TIP.2003.819861.

. [10] Zhichao Yin and Jianping Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. CoRR, abs/1803.02276, 2018. URL: http://arxiv.org/abs/1803.02276, arXiv:1803.02276.