Master of Science Thesis in Electrical Engineering
Department of Electrical Engineering, Linköping University, 2020

Monocular depth and motion estimation using neural networks

Erik Örjehag

Master of Science Thesis in Electrical Engineering

Monocular depth and motion estimation using neural networks

Erik Örjehag

LiTH-ISY-EX--YY/NNNN--SE

Supervisor: **Gustav Häger**
ISY, Linköpings universitet

Examiner: **Per-Erik Forssén**
ISY, Linköpings universitet

Division of Computer Vision
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Sammanfattning

Det här som vi har hållit på med är jätteviktigt faktiskt och det vi gjort blev bara sååå bra. Kanske inte helt otippat, men det glass är sååå gott!

Förresten har vi blivit bäst på att skriva rapporter, så nu ska ska vi inte gå in närmare på några detaljer såhär i sammanfattningen.

Abstract

If your thesis is written in English, the primary abstract would go here while the Swedish abstract would be optional.

Acknowledgments

Vi tycker alla har varit så himla goa hela den här långa och tuffa tiden i våra liv.

Linköping, Januari 2020
N N och M M

Contents

Notation

Några mängder

Notation	Betydelse
$\mathbb{N}$	Mängden av naturliga tal
$\mathbb{R}$	Mängden av reella tal
$\mathbb{C}$	Mängden av komplexa tal

Förkortningar

Förkortning	Betydelse
ARMA	Auto-regressive moving average
PID	Proportional, integral, differential (regulator)

1 Introduction

This thesis aims to show how new neural network techniques can be used to predict depth and relative camera motion for an image sequence captured by a monocular RGB camera. Imagine closing one eye and looking out into the world, it is trivial as a grownup human to detect motion and estimate how the head moves in relation to what is seen. Calculating camera movement from an image sequence is a well studied problem and is usually done by finding corresponding features in the images and calculating (using projective geometry) which camera movement can give rise to such correspondences and their relative movement between frames in the sequence.

Recent research has shown that it's possible to predict depth and relative motion from a sequence of images taken with a monocular RGB camera, up to a unknown scale factor. The training data is a sequence of unlabeled images with a small relative motion, for example looking out from the front window of a moving car. Given a target view and a new nearby view it's possible to train depth and pose predicting CNNs jointly using a combined loss function. The depth and pose predictions are used to warp nearby views to the target view and the loss is based on the similarity achieved after warping.

1.1 Contributions

This thesis does an analysis, and a performance comparison of different techniques described in current research papers. Training and testing is done on new datasets.

1.2 Motivation

Localization by only using visual input is highly desirable in robotics applications due to the low hardware cost and power consumption of using cameras compared to, for example, 3D lidars. Obtaining labeled data can be a tedious task, for that reason this thesis will focus on unsupervised learning on unlabeled data.

1.3 Research Questions

1. What ideas from previous work can be combined and what are the performance gains if any?
2. How well do previous methods work on new datasets not tested in the original papers? In what ways do the results break down?
3. What alteration to previous loss functions can be made to improve results?
4. How does the amount of training data affect the results? Can the performance of the methods from previous research be improved simply by training on more data?
5. Can the results from the original papers be replicated in the PyTorch framework?

1.4 Delimitations

The visual localization problem can be solved using, for example, a stereoscopic camera or a time of flight camera. But this thesis will only explore the use of a monoscopic, non depth sensing, RGB camera, because it enables applications where hardware cost is a big factor.

In a full SfM pipeline, both 3D-reconstruction, bundle adjustment and loop-closure detection are usually done as well. This will not be part of this thesis project.

2 Background

2.1 Convolutional neural networks

The central method used in this project is a deep learning algorithm called convolutional neural networks (CNN for short). A CNN architecture can successfully capture the spatial dependencies in an image through convolutional filtering operations with kernels of learnable weights and biases.

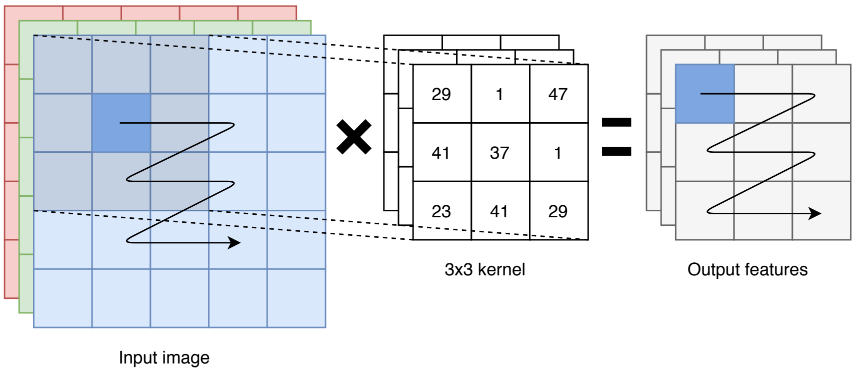


__Figure 2.1:__ Convolutional filtering operation with a 3 channel RGB image and 3x3 kernel

In Figure 2.1 a convolutional filtering operation over an image is illustrated. The matrix kernel is moved in a row by row pattern and is multiplied by a patch of the image to get a value for the output cell.

In order to form a deep neural network multiple filtering operations are chained sequentially with nonlinear activation functions between them. A deep neural network usually consists of many such layers of filtering operations and activation functions. This concept is illustrated in Figure 2.2.

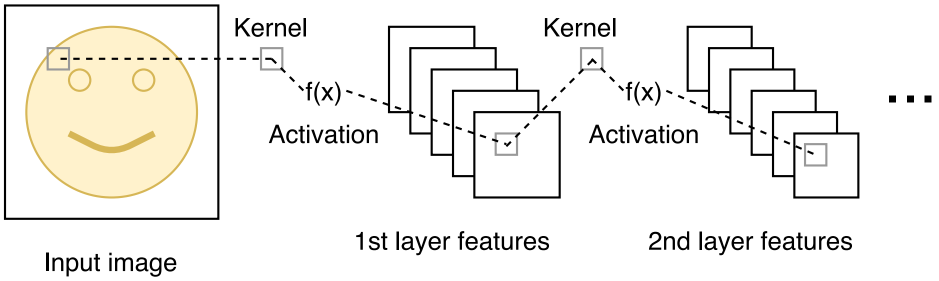


Figure 2.2: *Multiple layers chained with each other to form a deep network*

The activation functions need to be differentiable because the derivatives are used in the learning process. Some common nonlinear activation functions are shown in Figure 2.3.

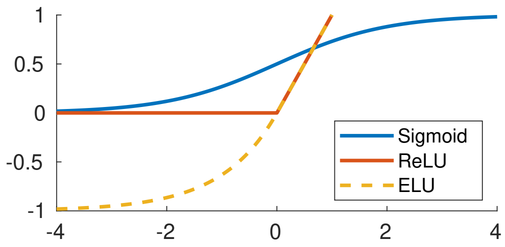


Figure 2.3: *A few common nonlinear activation functions*

The features in the deep neural networks are used to formulate a loss function. The loss defines an objective that we want the network to learn. This means that the process of learning becomes a task of updating the filter kernels in such a way that the loss function is minimized. If the loss function is decreasing during the training process it means that the network is learning. It is crucial that the loss function describes the problem accurately, otherwise the network will not learn the correct behavior. The weights in the network are updated using a method called back propagation. The algorithm computes the gradient of the loss function with respect to the weights in the network for a single input example from the training data using the derivative chain rule, which can be done very efficiently. Updating the weights to minimize the loss function can then be done using gradient decent. As the network is fed with more input examples from the training data, the network slowly learns the correct weights that minimizes the loss function as desired.

3 Related work

In this chapter some of the important contributions of previous research papers are summarized.

3.1 Unsupervised Monocular Depth Estimation with Left-Right Consistency [4]

In a paper from 12 Apr 2017 the authors present MonoDepth, with an implementation available in Tensorflow on github. In this work the depth is predicted using an encoder-decoder type network, but the relative motion between frames is not estimated at all. The KITTI dataset provides stereo image pairs which are used during training, and the relative transformation between the left and right cameras is known. Using only the left image as input to the network both disparity maps for the left and right images are predicted. The two disparity maps are used to project the left image into the view of the right image and vice versa. This can be seen as a precursor to the papers discussed later which uses only a monocular image sequence to train a depth predicting network and pose predicting network jointly.

In the loss function, the left image is reconstructed from the right image and vice versa. This is possible using the disparity maps and known baseline and focal length of the cameras. The L1 norm of the per pixel photometric error as well as SSIM are computed and added to the loss. An additional loss term encourages the left disparity to be equal to the right disparity projected into the left camera viewpoint. Because the photometric error does not work well on low textured regions an edge aware smoothess term is added to propagate the depth values from nearby areas in the disparity map. The method produces metrically accurate results because the baseline and focal length of the cameras are known.

3.2 Unsupervised Learning of Depth and Ego-Motion from Video [13]

In a paper from CVPR 21 July 2017 the authors present SfMLearner with an official implementation in Tensorflow on github. Additional replications in the PyTorch and Chainer frameworks are also available.

Contrary to the paper in section 3.1 only a monocular sequence of images from the KITTI dataset is used during training. In the stereo case the relative pose between the left and right cameras was known, but with this monocular dataset the pose between subsequent frames is unknown. The authors train a pose predicting and depth predicting network simultaneously with a joint loss function to solve this problem.

During training 3 subsequent frames are considered at a time. The frame I_{t-1} and frame I_{t+1} are called the source frames and the frame I_t is called the target frame. The target frame is input to the depth network which estimates a disparity map. The two source frames are fed through a pose estimating network one after each other together with the target frame to find the relative transformations $T_{t\to t-1}$ and $T_{t\to t+1}$.

The authors add an explainability mask to the photometric error term to account for errors in the model. The view synthesis formulation implicitly assumes that the scene does not contain moving objects, that there are no occlusions between the target and source frames, and that the surfaces are Lambertian so that the photo-consistency error of RGB values is meaningful. In order to predict the explainability mask an additional CNN is used. The network has no explicit supervisory signal but is encouraged to be non-zero with an regularization term using a cross-entropy loss with a constant label 1 at each pixel location. This makes the network minimize the view synthesis objective but is allowed some slack due to factors not considered by the model. In later work it was shown that the explainability mask does not help to improve results that much and is often ignored.

To tackle the problem with textured areas and non Lambertian surfaces, a smoothess term is used. An edge aware smoothess term was not used like in previous work[4] but a penalty on the second order gradient of the depth map was used instead. This unfortunately makes edges very fuzzy in the results compared to other methods.

3.3 SuperDepth: Self-Supervised, Super-Resolved Monocular Depth Estimation [11]

In this paper published 3 Oct 2018 the authors train a depth estimating network on only stereo image pairs. After the depth predicting network has been trained they train the pose network from section 3.2 using results from the already trained depth predicting network. This means that the depth and pose networks are not trained with a joint loss function but are instead trained separately.

The main contributions from this paper is the proposal to use supixel-convolution. They additionally use differentiable flig-augmentation to remove edge artifacats on the left and right edges of the depth map seen in previous work using sterio image pairs during training.

To handle occluded pixels between the left and right images a occulsion regularization loss term is added to encourage background depths (low disparaty).

3.4 3D Packing for Self-Supervised Monocular Depth Estimation [6]

The authors present PacknetSfM in their paper on 6 Dec 2019. The main contribution is a new network architecture with packing and unpacking blocks replacing down and up sampling. The new packing blocks uses space to depth transformations and 3d convolutions. The author claims that the new packing and unpacking blocks are better at perserving resolution than standard down and upsampling. As the method is not adopted in later work, it is unclear if the new packing and unpacking approach is actually any good.

The second contribution is a loss on the camera velocity that makes it possible for the monocular depth estimation to be metrically accurate.

Alterations to the loss function not seen in previously discussed papers is a mask on pixels that do not change between frames. They are assumed to have no ego motion. Stationary pixels can occur for example if the car dashboard is visible in frame, or other vehicles are moving at a similar speed nearby. The other contribution is the velocity supervision loss. The velocity of the car is assumed to be known in the training dataset.

3.5 Digging Into Self-Supervised Monocular Depth Estimation [5]

The authors present MonoDepth2 on 17 Aug 2019. They propose mainly two contributions.

Firstly, instead of taking the average of the reprojection errors from all source frames given a target frame they use the minimum. This makes it so that if a feature is occluded in one source image but not in the other the errors will not be averaged together but instead the error from the source frame that is not occluded will be used.

Secondly, instead of calculating the loss for each depth scale in the decoder all the depth maps are upsampled to the original target image size when computing the loss. This way a single pixel in the low resolution layer of the decoder will predict the depth of a patch of pixels in the originally sized input image.

3.6 Self-Supervised 3D Keypoint Learning for Ego-motion Estimation [12]

This paper from 7 Dec 2019 combines the work of the previously discussed papers and also adds keypoint learning from SuperPoint[1] and its successor Unsuperpoint[2] into the pipeline. The researchers train the depth, keypoint and pose estimating networks jointly making them benifit from each other and achieve state of the art results.

4 Method

4.1 Datasets

The neural networks are trained and evaluated on 3 different datasets, KITTI[3], Lyft[9] and Synthia. All datasets are preprocessed to remove frames where the camera is not moving. This is important because if there is no movement between frames then no depth information can be inferred during training when using monoscopic data. In the Kitti dataset the images from the left and right camera are treated as separate image sequences to yield more training data. The images are resized to 128x416 pixels, and the intrinsic camera matrix is updated accordingly. The lidar data from Kitti and Lyft is converted to sparse depth maps and are used during performance evaluation. The dataloader works by loading triplets of adjacent frames in the image sequences, and also the ground truth movement and depth map. An example of this is seen in Figure 4.1.

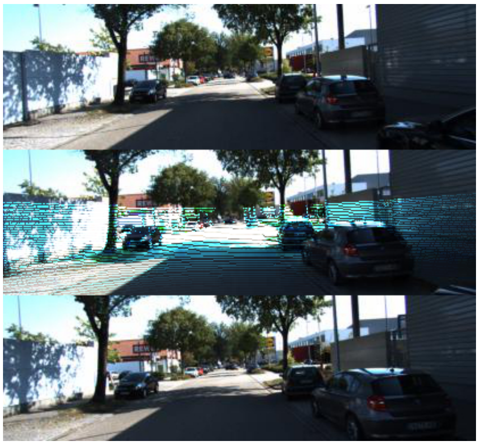


Figure 4.1: *From top to bottom 3 frames from Kitti, I_{t-1}, I_t, I_{t+1} with the sparse depth map overlayed on frame I_t.*

	Sequences / Samples	
	Train	Test
Kitti	110 / 16542	12 / 11349
Lyft	134 / 3759	14 / 1735
Synthia	x / x	x / x

4.2 Architectures

In order to predict depth and motion from monocular images two different CNN architectures are examined.

4.2.1 SfMLearner architecture

This is the architecture from [13]. The authors use a DispNet[10] architecture to predict depth maps at four different scales, and a ResNet18[7] architecture with modified decoder to predict pose updates in an euler angle axis representation.

4.2.2 Monodepth2 architecture

This is the architecture from [5]. The authors use a ResNet18 architecture instead of a DispNet architecture to predict depth estimates. They make this choice because its a smaller and faster architecture. Similarly they use a ResNet18 architecture with modified decoder to predict the pose updates in an euler angle axis representation.

4.3 Differentiable depth image warping

The core component of unsupervised depth learning is the differentiable depth image warp operation in the loss function of the CNN networks. Given the intrinsic camera matrix:

$$K = \begin{pmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{pmatrix}$$

And the predicted depth $D_t(p_t)$ of pixel p_t of the target (current) frame. And the transform $T_{t\to s}$ from the target to source (next/previous) frame:

$$T_{t\to s} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{pmatrix}$$

The position of the target pixel p_t in the source image p_s can be calculated in homogeneous coordinates as:

$$p_s \sim K T_{t\to s} D_t(p_t) K^{-1} p_t$$

The pixel position p_s is however continuous and in order to sample the discrete source image I_s a differentiable bilinear sampling method is used. The method is described in *spatial transformer networks*[8] and works by interpolating the neighbouring 4 pixels values (top-left, bottom-right) by the distance to the the continuous sampling point p_t. This process is illustrated in Figure 4.2.

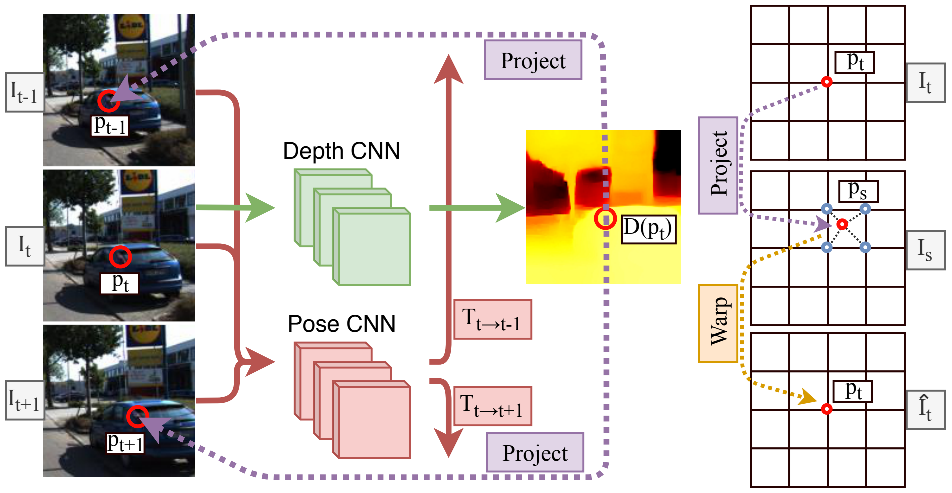


Figure 4.2: The CNN predicts the depth map D of the target image I_t, and also the relative movement, $T_{t\rightarrow t-1}$, $T_{t\rightarrow t+1}$ between the target image and the source images. Each pixel p_t in the target image is projected onto a position in the source images which are sampled using bilinear interpolation. This should recreate the appearance of the target image but with pixels sampled from the source image. An appearance similarity metric between the original target image and the recreated target images can be used as the loss function for the CNN to learn to accurately predict correct depth and movement.

4.4 Loss functions

Photometric loss Is defined as $\mathcal{L}_p(I_t, \hat{I}_s) = |I_t - \hat{I}_s|$.

SSIM loss Is defined as $\mathcal{L}_{ssim}(I_t, \hat{I}_s) = \dfrac{1 - \mathrm{SSIM}(I_t, \hat{I}_s)}{2}$.

Combined loss The photometric and SSIM loss is often combined and balanced using $\mathcal{L}_{ps}(I_t, \hat{I}_s) = \alpha\mathcal{L}_{ssim} + (1-\alpha)\mathcal{L}_p$

Depth smooth loss Is defined as $\mathcal{L}_{smooth}(D_t) = |\delta_x^2 D_t| + |\delta_y^2 D_t|$. Not ideal because it can cause very fussy edges as seen in SfMLearner.

Edge aware depth smooth loss Is defined as $\mathcal{L}_{edge}(D_t) = |\delta_x D_t| e^{-|\delta_x I_t|} + |\delta_y D_t| e^{-|\delta_y I_t|}$. Applied in Monodepth2 giving sharper edges because the smoothness term is weighted to mostly affect areas with small photometric derivitive.

Velocity supervision loss When a velocity measurement exists in the dataset a term to enforce scale accurate estimates can be added like $\mathcal{L}_v = \left| \|\mathbf{t}_{t \to s}\| - |v|\Delta t \right|$, as proposed in packnet[6].

4.4.1 Handling occlusions

Disparity loss To encourage background depths (low disparities) in shadows of the depth map where occlusion has occurred a penalty on the disparity can be added $\mathcal{L}_o = |d_t|$.

Minimum loss across frames In SfMLearner the photometric loss is calculated for the previous and next frames compared to the current in the sequence. The pixel wise average across the frames are then used. This causes problems if a pixel is for example occluded in the previous frame, but visible in the current and next frame. In this situation the average loss will be pretty high even though a correct depth and transformation has been predicted, because of the occluded pixel. Instead Monodepth2 suggests to pick the minimum per pixel error over the frames which creates a more telling loss.

4.5 Handling model limitations

In order to optimize using the photometric reprojecton error as the loss function two assumptions must hold. Firstly the scene must be static, meaning all objects in the scene must be still except the moving camera. Movement by cars and humans in the scene that is not due to the camera movement will cause problems. Secondly there must be photometric consistency between frames for the photometric error to make sense. This means that non lambertian surfaces, change in lighting, and change in exposure between frames will cause problems.

Explainability mask The authors of [13] tackle this problem by having a CNN predict what pixels are valid to use in the photometric loss function. It shares the encoder of the pose predicting network but branches of into a different encoder which estimates a mask of the valid/explainable pixels. The loss function for the mask is the cross entropy loss compared to a mask filled with ones. The photometric loss function is augmented to include the explainability mask removing pixels that cannot be explained by the predicted depth and transformation. This encourages the mask to be filled with ones, but allows some slack due to pixels that can not be explained by the photometric loss.

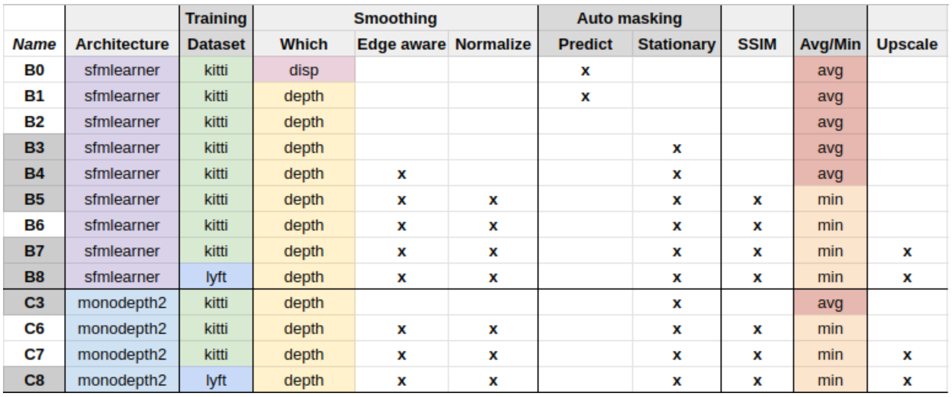
Stationary pixels mask The authors of [5] introduced a mask to remove stationary pixels from the set of previous, current and next frame. This is done by creating a mask where the photometric error is smaller before applying the projection than after. This works because stationary pixels that have not moved in relation to the camera will of course have a small photometric loss without reprojection. This will remove pixels from the car dashboard and also nearby vehicles that are traveling at the same speed.

4.6 Evaluation

TODO: Describe evaluation metrics...

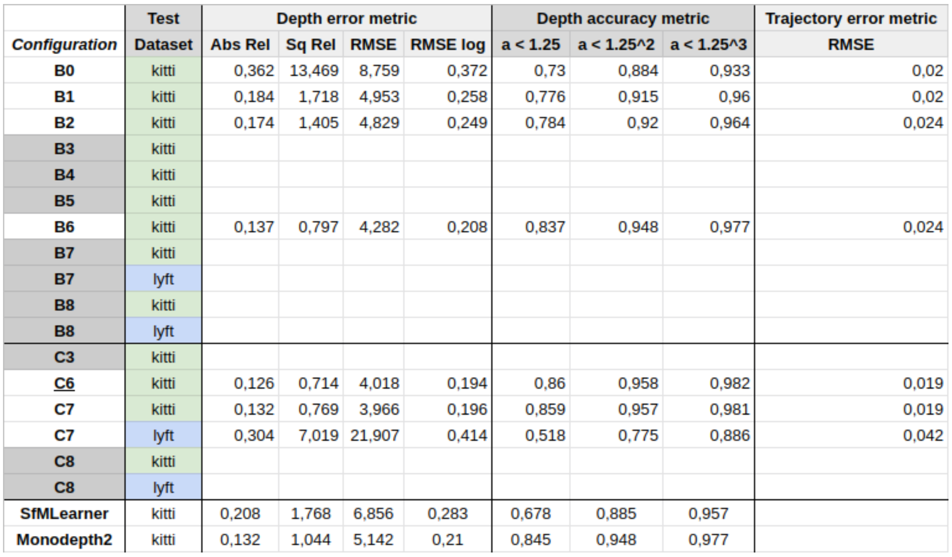
5 Resultat

Figure 5.2 contains some preliminary evaluation results. The gray configurations in Figure 5.1 have not been trained yet. Each configuration takes about 15 hours to train. Still missing possibility to train on Synthia and also random data augmentations, L1 loss on disparity and velocity supervision loss.



		Training	Smoothing			Auto masking				
Name	Architecture	Dataset	Which	Edge aware	Normalize	Predict	Stationary	SSIM	Avg/Min	Upscale
B0	sfmlearner	kitti	disp			x			avg	
B1	sfmlearner	kitti	depth			x			avg	
B2	sfmlearner	kitti	depth						avg	
B3	sfmlearner	kitti	depth				x		avg	
B4	sfmlearner	kitti	depth	x			x		avg	
B5	sfmlearner	kitti	depth	x	x		x	x	min	
B6	sfmlearner	kitti	depth	x	x		x	x	min	
B7	sfmlearner	kitti	depth	x	x		x	x	min	x
B8	sfmlearner	lyft	depth	x	x		x	x	min	x
C3	monodepth2	kitti	depth				x		avg	
C6	monodepth2	kitti	depth	x	x		x	x	min	
C7	monodepth2	kitti	depth	x	x		x	x	min	x
C8	monodepth2	lyft	depth	x	x		x	x	min	x

Figure 5.1: *Different configurations of network architecutes, training datasets and loss terms evaluated to find the best performance*



	Test	Depth error metric				Depth accuracy metric			Trajectory error metric
Configuration	**Dataset**	**Abs Rel**	**Sq Rel**	**RMSE**	**RMSE log**	**a < 1.25**	**a < 1.25^2**	**a < 1.25^3**	**RMSE**
B0	kitti	0,362	13,469	8,759	0,372	0,73	0,884	0,933	0,02
B1	kitti	0,184	1,718	4,953	0,258	0,776	0,915	0,96	0,02
B2	kitti	0,174	1,405	4,829	0,249	0,784	0,92	0,964	0,024
B3	kitti								
B4	kitti								
B5	kitti								
B6	kitti	0,137	0,797	4,282	0,208	0,837	0,948	0,977	0,024
B7	kitti								
B7	lyft								
B8	kitti								
B8	lyft								
C3	kitti								
C6	kitti	0,126	0,714	4,018	0,194	0,86	0,958	0,982	0,019
C7	kitti	0,132	0,769	3,966	0,196	0,859	0,957	0,981	0,019
C7	lyft	0,304	7,019	21,907	0,414	0,518	0,775	0,886	0,042
C8	kitti								
C8	lyft								
SfMLearner	kitti	0,208	1,768	6,856	0,283	0,678	0,885	0,957	
Monodepth2	kitti	0,132	1,044	5,142	0,21	0,845	0,948	0,977	

Figure 5.2: *Evaluation metrics when testing the configurations on the testing split of the datasets*

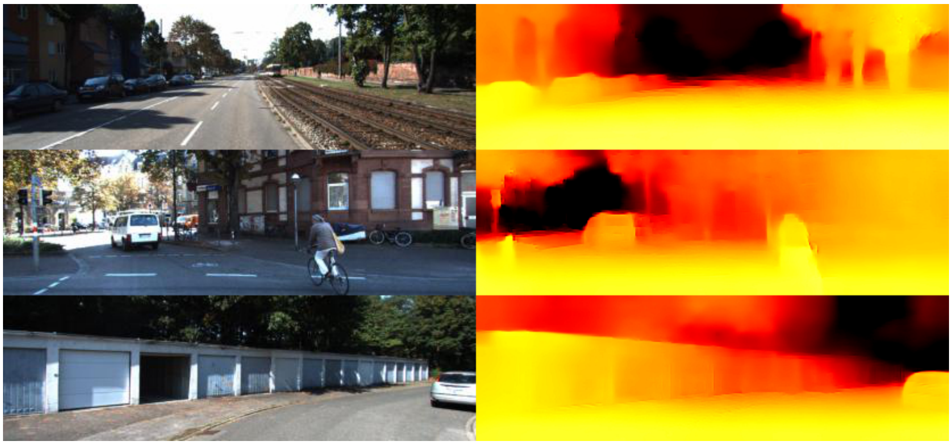


Figure 5.3: *Examples from the Kitti dataset*

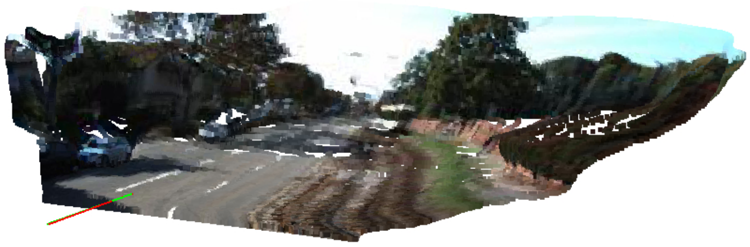


Figure 5.4: *3D render of colorized depth map*

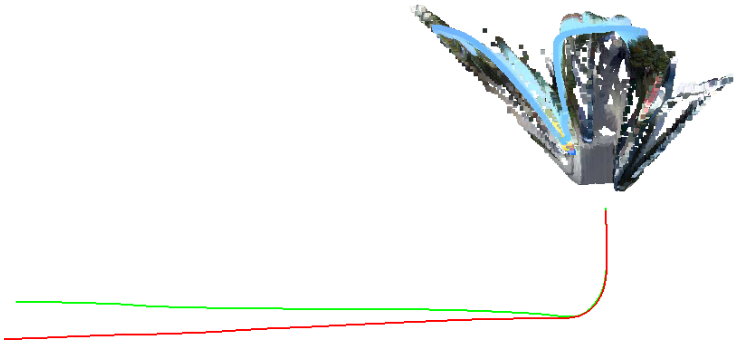


Figure 5.5: *3D visualization of the camera movement in a long image sequence. The green line is ground truth and the red line is the predicted camera trajectory.*

Bibliography

[1] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. 12 2017.

[2] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. pages 337–33712, 06 2018. doi: 10.1109/CVPRW.2018.00060.

[3] Andreas Geiger, P Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: the kitti dataset. *The International Journal of Robotics Research*, 32:1231–1237, 09 2013. doi: 10.1177/0278364913491297.

[4] Clement Godard, Oisin Aodha, and Gabriel Brostow. Unsupervised monocular depth estimation with left-right consistency. 07 2017. doi: 10.1109/CVPR.2017.699.

[5] Clément Godard, Oisin Aodha, and Gabriel Brostow. Digging into self-supervised monocular depth estimation, 06 2018.

[6] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, and Adrien Gaidon. Packnet-sfm: 3d packing for self-supervised monocular depth estimation, 05 2019.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. 7, 12 2015.

[8] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, 06 2015.

[9] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet. Lyft level 5 av dataset 2019. urlhttps://level5.lyft.com/dataset/, 2019.

[10] N.Mayer, E.Ilg, P.Häusser, P.Fischer, D.Cremers, A.Dosovitskiy, and T.Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE International*

Conference on Computer Vision and Pattern Recognition (CVPR), 2016. URL `http://lmb.informatik.uni-freiburg.de/Publications/2016/MIFDB16`. arXiv:1512.02134.

[11] Sudeep Pillai, Rares Ambrus, and Adrien Gaidon. Superdepth: Self-supervised, super-resolved monocular depth estimation. pages 9250–9256, 05 2019. doi: 10.1109/ICRA.2019.8793621.

[12] Jiexiong Tang, Rares Ambrus, Vitor Guizilini, Sudeep Pillai, Hanme Kim, and Adrien Gaidon. Self-supervised 3d keypoint learning for ego-motion estimation, 12 2019.

[13] Tinghui Zhou, Matthew Brown, Noah Snavely, and David Lowe. Unsupervised learning of depth and ego-motion from video. 04 2017.