

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

Unsupervised Learning for Structure from Motion

Erik Örjehag

Master of Science Thesis in Electrical Engineering

Unsupervised Learning for Structure from Motion

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*

Copyright © 2020 Erik Örjehag

Abstract

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

Acknowledgments

I would like to thank my supervisor Gustav Häger and examiner Per-Erik Forssén for their assistance and creative freedom letting me explore the topics that are of high interest to me. I would also like to thank the company Dyno Robotics for lending me the computer resources necessary to perform the experiments in this thesis.

*Linköping, October 2020
Erik Örjehag*

Contents

Notation	ix
1 Introduction	1
1.1 Motivation	2
1.2 Research Questions	2
1.3 Delimitations	3
2 Background	5
2.1 Convolutional neural networks and learning	5
3 Related work	7
3.1 Unsupervised depth and ego motion prediction	7
3.2 Unsupervised keypoint prediction	9
3.3 Unsupervised geometric consensus maximization	10
3.4 Combined depth and feature point detection	10
4 Method	11
4.1 Datasets	11
4.1.1 Sequence datasets	12
4.1.2 Homography dataset	12
4.2 Combining techniques from previous research	13
4.3 Evaluation metrics	13
4.3.1 Depth error and accuracy metrics	13
4.3.2 Camera ego motion error metric	15
4.3.3 Keypoint error and score metrics	15
4.3.4 Consensus maximization performance metrics	15
5 Implementation	17
5.0.1 Architectures for depth and ego motion CNNs	17
5.0.2 Differentiable depth image warping	18
5.0.3 Loss functions	19
5.0.4 Handling occlusions	21
5.0.5 Handling model limitations	22

5.1	Unsupervised keypoint learning	24
5.1.1	Loss function for keypoint learning	26
5.2	Unsupervised consensus maximization	26
5.2.1	Improving training convergence	31
6	Results	33
6.1	Depth and ego motion	34
6.2	Keypoint detection	37
6.3	Consensus maximization	38
7	Discussion	41
7.1	Results	41
7.2	Method	41
7.3	The work in a wider context	41
8	Conclusions	43
8.1	Future work	43
Bibliography		45

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 aim of this thesis to investigate the performance and implementation of some newly published techniques of using unsupervised training of neural networks in the structure from motion pipeline.

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 few 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.

In addition to predicting depth and motion in an image sequence, it is also useful to extract feature points in the image that can be tracked over time in order to build a map of the world. In the process of extracting feature points the system should be able to filter out points that are matched incorrectly with each other, so called outliers.

If depth prediction, camera motion, feature point detection and geometric con-

sensus maximization are implemented in an unsupervised training framework it opens up the possibility to train all subsystems jointly where each part benefits from the rest. Collecting training data would be cheap, since no manual annotation of the data is needed. But this is left as an area of future work and not covered in this thesis.

1.1 Motivation

Here are three main points motivating future research into the use of unsupervised learning methods for visual structure from motion.

- Localization of autonomous vehicles is commonly achieved using lidar sensors due to their high accuracy and robustness, but using cameras instead comes with many other benefits. For example lower hardware cost and power consumption. Cameras are passive sensors which means that they don't interfere with each other. Lidars rely on spinning parts which can break if subjected to shaking or impact. Focusing on monoscopic vision instead of stereo vision again comes down to cost benefits.
- Obtaining labeled data for supervised training can be a tedious task, in that respect unsupervised methods are much more desirable. Labeling data is time consuming, expensive and prone to human errors. Collecting and storing data without labeling it is however comparably easy and inexpensive.
- This thesis focuses on depth, ego motion, keypoint prediction and consensus maximization. The motivation for this choice of unsupervised methods is the idea that in future work it might be possible to combine these methods to jointly learn all the tasks at the same time where each part of the system benefits from the others during training. The hope would be that the performance of the system would increase simply by collecting more unlabeled data.

1.2 Research Questions

1. How well do the unsupervised methods from previous research work on new datasets not tested in the original papers?
2. What are the performance gains of combining different methods from recent research in monocular depth and ego motion prediction?

1.3 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 detection, 3D-reconstruction, bundle adjustment and loop-closure are commonly included. The detection and prediction on input images is usually referred to as the SfM frontend, while the map building is referred to as the backend. This thesis will primarily focus on how unsupervised training can be applied to the frontend, and will not be investigating anything to do with the backend.

This thesis will focus on a few techniques to train a neural network to predict depth and camera motion from a sequence of unlabeled images. Additionally the thesis will evaluate two specific methods to predict feature points and perform geometric consensus maximization on the correspondences using neural networks.

2

Background

2.1 Convolutional neural networks and learning

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.

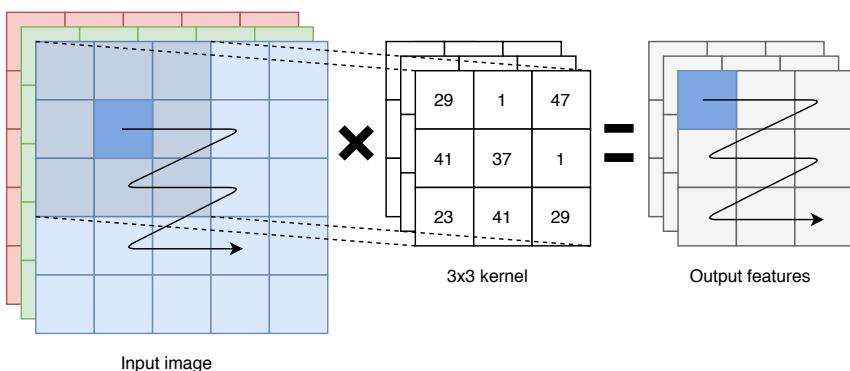


Figure 2.1: Convolutional filtering operation with a 3 channel RGB image and 3x3 kernel.

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.

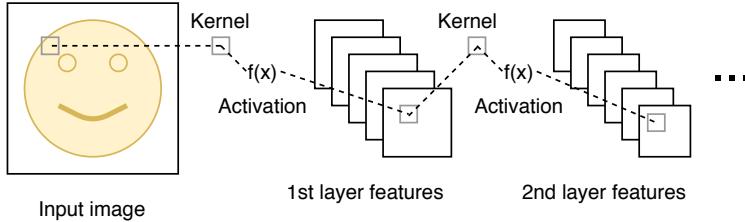


Figure 2.2: Multiple layers chained with each other to form a deep network.

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.

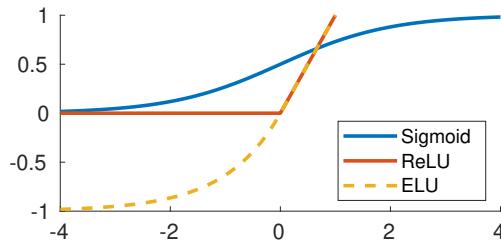


Figure 2.3: A few common nonlinear activation functions.

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 descent. 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. Firstly a list of papers in the field of unsupervised depth and ego motion prediction is presented. Secondly a paper on unsupervised feature point prediction. Thirdly a paper on unsupervised geometric consensus maximization. And finally a paper on combining depth and feature point prediction, albeit using classical consensus maximization that is not learned by a neural network.

3.1 Unsupervised depth and ego motion prediction

All the papers in this section are discussed in the order of publication to show a timeline of progress in the field of depth and ego motion prediction.

In this paper[5] 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. The L1 norm of the per pixel photometric error as well as SSIM [16] 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 smoothness 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.

In this paper[15] the authors present SfMLearner with an official implementation in Tensorflow on github. Contrary to the previous paper[5] 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 $T_{t \rightarrow t-1}$ and $T_{t \rightarrow t+1}$. 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 smoothness term is used. An edge aware smoothness term was not used like in previous work[5] 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 using the edge aware smoothness loss.

In this paper[12] the authors train a depth estimating network on only stereo image pairs. After the depth predicting network has been trained the pose network from [15] is trained, 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 a new subpixel-convolution operation that super-resolve disparities from their low-resolution outputs, thereby replacing the up-sampling layers typically used in the disparity decoder network. The method additionally uses differentiable flip-augmentation to remove edge artifacts on the left and right edges of the depth map seen in previous work using stereo image pairs during training. To handle occluded pixels between the left and right images an occlusion regularization loss term is added to encourage background depths (low disparity).

In this paper[7] the authors present PacknetSfM. The main contribution is a new network architecture with packing and unpacking blocks replacing down and up sampling. The new packing blocks use space to depth transformations and 3d convolutions. The authors claim that the new packing and unpacking blocks are better at preserving 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 effective. The second contribution is a loss on the camera velocity that makes it possible for the monocular depth estimation to be metrically accurate. The velocity of the car is assumed to be known in the training dataset. The third contribution is a mask on pixels that do not change between frames. These pixels are assumed to be part of objects that are stationary with respect to the camera. Such pixels can occur for example if the car dashboard is visible in a frame, or other vehicles are moving at a similar speed nearby.

In this paper[6] the authors present MonoDepth2. 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 up-sampled 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.

3.2 Unsupervised keypoint prediction

Collecting ground truth data to train a neural network for keypoint detection is cumbersome. What constitutes a keypoint is hard to define clearly and consistently for a human annotator.

SuperPoint[2] uses a unsupervised approach to learn prediction of points and their descriptors. The network is first trained on a synthetic dataset of simple geometric shapes. The pseudo ground truth points are defined as the corners and junctions in the synthetic images. This pre-trained network is then trained on "real" images in a siamese network setup using a method they call homographic adaptation. The two siamese siblings are fed images transformed by random but known homographies and the output from the networks are compared in the loss function. The output of SuperPoint is a heatmap where each pixel describes the "point-ness" of that pixel, and a descriptor map. To keep the model fast both the point and descriptor maps are predicted semi-dense grid, one cell for each 8 pixel patch. The maps are up-sampled to the original image size using bicubic interpolation, and the descriptors are normalized.

UnsuperPoint[1] is heavily inspired by SuperPoint but does not need to be pre-trained on a synthetic dataset, instead it is trained in one round of training on

real images. It uses a similar method of using siamese networks and homography adaptation during training, but the output from the network is different. The network uses regression of actual point position coordinates instead of a heat map, incorporates non-maximum suppression, predicts scores for each keypoint and a sparse descriptor map that is sampled and interpolated using the point positions.

3.3 Unsupervised geometric consensus maximization

Consensus maximization is an important strategy in 3D vision problems for robust geometric model estimation from measurements including outliers. The classical method of *Random Sampling and Consensus*, or RANSAC[3] for short, is widely popular with great success. But replicating the same generic behavior using supervised training of neural networks has proven difficult. Unsupervised methods have a huge potential to generalize to any unseen data distribution and are in this context very desirable. This paper[?] introduces just such an unsupervised method of consensus maximization for 3D vision problems. Using the relationship between the set of inliers, and the subspace of polynomials representing the space of target transformations a model fitting cost can be defined without knowing the specific parameters of the geometric transformation. During learning the loss is defined in a way to learn the largest set of inliers with a low model fitting cost. The geometric model parameters can then be extracted from the polynomials after network has learnt to distinguish inliers and outliers.

3.4 Combined depth and feature point detection

In a paper[14] the authors combines the work of the previously discussed papers on depth prediction, but also adds keypoint learning from SuperPoint[2] into the pipeline. The researchers train the depth, keypoint and pose estimating networks jointly, making them benefit from each other and achieve state of the art performance. However, the step that finds possible outlier keypoints, is not differentiable and is not trained jointly with the depth and keypoint networks. The step that calculates the model parameters of the geometric relationship between corresponding keypoints also requires an initial guess that is not differentiable.

4

Method

This chapter describes the methods used to answer the research questions. The first question relates to the datasets used during training and testing of the networks. The second question relates to the performance benefits of applying specific techniques from previous research in unsupervised monocular depth and ego motion prediction.

To begin with the chapter summarizes which datasets were used in this thesis and how the choice relates to the first research question. Following that an explanation of which specific techniques used in monocular depth prediction will be evaluated in the results chapter, and the method used to combine them. Lastly a detailed description of all the error and accuracy metrics that will be presented in the results chapter. A good understanding of what the metrics can tell us is needed to answer both research questions.

4.1 Datasets

The neural networks are trained and evaluated on images from 2 different datasets, KITTI[4] and Lyft[10]. Both 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.

4.1.1 Sequence datasets

TODO....

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

4.1.2 Homography dataset

TODO....

How to generate the random homography used during training is not described in the UnsuperPoint paper, but is an important part of the method. The method used in this thesis generates random homographies from 5 parameters $\alpha_{rotation}$, $\alpha_{translation}$, α_{scale} , α_{sheer} and $\alpha_{perspective}$. The parameters controls the maximum transformation for each aspect of an homography. The final homographgry is constructed from parts as follows.

$$H = H_{affine} H_{sheer} H_{perspective}$$

Assume $u_n \sim U(-1, 1)$ are random uniform variables in the range -1 to 1.

$$H_{affine} = \begin{pmatrix} \cos(r) * s & -\sin(r) & t_x \\ \sin(r) & \cos(r) * s & t_y \\ 0 & 0 & 1 \end{pmatrix}, \text{with } \begin{cases} r = u_1 * \alpha_{rotation} \\ s = u_2 * \alpha_{scale} + 1 \\ t_x = u_3 * \alpha_{translation} \\ t_y = u_4 * \alpha_{translation} \end{cases}$$

$$H_{sheer} = \begin{pmatrix} 1 & s & 0 \\ s & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \text{with } s = u_5 * \alpha_{sheer}$$

$$H_{perspective} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ p & p & 1 \end{pmatrix}, \text{with } p = u_6 * \alpha_{perspective}$$

4.2 Combining techniques from previous research

TODO....

4.3 Evaluation metrics

This section describes the metrics used to evaluate depth, ego motion, feature point and consensus maximization predictions. These specific metrics where chosen because they are also used in the related works in this field, which makes the results comparable to other papers.

4.3.1 Depth error and accuracy metrics

The depth error and accuracy metrics are sparse and only calculated for the pixels of which there exists a ground truth laser measurement in the dataset. The values are averaged over all laser measurements and frames in the test split of the dataset. An error of 0 and an accuracy of 1 is optimal, but can never be achieved in practice.

The depth is predicted relative to an unknown scale, and the ground truth is measured in meters. To alleviate this issue the depth predictions are scaled so that their median is the same as the median of the ground truth for each frame. This does not guarantee that the unit of the predictions is meters, but at least it makes the scales somewhat similar.

The error metrics used in the results chapter are referred to as *Abs Rel*, *Sq Rel*, *RMSE* and *RMSLE*.

The *Abs Rel* error is based on *MAE* which is the mean of the absolute errors which means that it has the same unit as the errors, and is conceptually quite easy to interpret.

$$\text{MAE} = \frac{\sum_{n=1}^N |y_n - \hat{y}_n|}{N}$$

Because the depth from the neural network is without unit and only predicted relative to an unknown scale, a variation on the *MSE* metric called *Abs Rel* is used.

$$\text{Abs Rel} = \frac{\sum_{n=1}^N \frac{|y_n - \hat{y}_n|}{y_n}}{N}$$

The *MSE* metric is the mean of squared errors, for an unbiased estimator it represents the variance of the errors.

$$\text{MSE} = \frac{\sum_{n=1}^N (y_n - \hat{y}_n)^2}{N}$$

Once again, because the depth is predicted relative to an unknown scale a variation on *MSE* called *Sq Rel* is used instead.

$$\text{Sq Rel} = \frac{\sum_{n=1}^N \frac{(y_n - \hat{y}_n)^2}{y_n}}{N}$$

The *RMSE* metric is the square root of the mean of squared errors. For an unbiased estimator it represents the standard deviation of the errors. Because the errors are squared in the *RMSE* metric it is more sensitive to outliers than for example *MSE*.

$$\text{RMSE} = \sqrt{\frac{\sum_{n=1}^N (y_n - \hat{y}_n)^2}{N}}$$

The *RMSLE* metric is similar to *RMSE* but is useful because it penalizes large errors less when both the actual and predicted values are large.

$$\text{RMSLE} = \sqrt{\frac{\sum_{n=1}^N (\log y_n - \log \hat{y}_n)^2}{N}}$$

To measure the depth accuracy, in the range from 0 to 1, the following metric is used.

$$a_\gamma = \frac{\sum_{n=1}^N (\max(\frac{v_n}{\hat{y}_n}, \frac{\hat{y}_n}{v_n}) < \gamma)}{N}, \text{ for } \gamma \in \{1.25, 1.25^2, 1.25^3\}$$

This should be interpreted as the ratio of predictions that is within the ratio of γ relative to the ground truth.

4.3.2 Camera ego motion error metric

The error of the camera motion predictions are measured using *RMSE*. But instead of taking the mean over all poses in a sequence, the alignment error is calculated part wise over a track length of only 5 poses. The final error is presented as the mean *RMSE* of all parts. Each track part is transformed to have its first pose coincide with the world origo. This makes it so that the first pose in the track part of both the prediction and ground truth is the identity matrix $I \in \mathbb{R}^{4 \times 4}$. Because the ego motion is predicted relative to an unknown scale and the ground truth is in meters, we scale each predicted track part to have similar scale to the ground truth track part.

4.3.3 Keypoint error and score metrics

TODO....

The unsupervised keypoint network was compared with the ORB feature detector from OpenCV. To measure the performance differences 3 different metrics where used.

Repeatability score (RS) is a measure on how good a method is at finding the same keypoints in two different images. Given two images of the same scene but from different angles, it is necessary that the method finds keypoints in the images that represents the same physical feature in the scene. Otherwise the set of matching keypoints is going to be very small when the points are paired based on their descriptors.

Localization error (LE) is a measure of the average distance error between corresponding points.

Matching score (MS) is a measure on how many keypoints are matched correctly based on their descriptors.

4.3.4 Consensus maximization performance metrics

TODO....

The unsupervised consensus maximization network was compared with the *findHomography()* and *findFundamentalMat()* from OpenCV.

Homography error (HE) is calculated as the average distance between the corners of the image transformed by the estimated homography and the ground truth homography.

RE and SED for fundamental matrix...

5

Implementation

This section describes the parts that will be implemented in PyTorch for this thesis.

5.0.1 Architectures for depth and ego motion CNNs

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

The first architecture is referred to as SfMlearner[15]. The authors use a DispNet[11] architecture to predict depth maps at four different scales, and a ResNet18[8] architecture with modified decoder to predict pose updates in an euler angle axis representation.

The second architecture is referred to as Monodepth2[6]. 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.

TODO: Insert diagram of general encoder-decoder type network used for depth and ego motion prediction....

5.0.2 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 $T_{t \rightarrow s}$ from the target to source (next/previous) frame:

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

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

$$p_s \sim (K \quad \mathbf{0}) T_{t \rightarrow s} D_t(p_t) K^{-1} p_t$$

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*[9] 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 5.1.

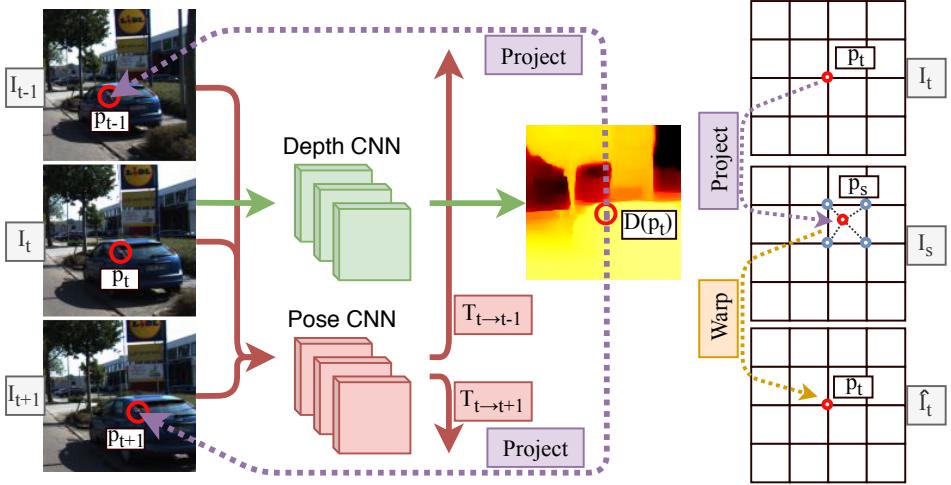


Figure 5.1: The CNN predicts the depth map D of the target image I_t , and also the relative movement, $T_{t \rightarrow t-1}$, $T_{t \rightarrow t+1}$ between the target image and the source images. Each pixel p_t in the target image is projected onto a position in the source images which are sampled 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.

5.0.3 Loss functions

In order to learn the objective of depth and ego motion prediction, different combinations of the following loss terms were evaluated.

To measure the similarity of the target image I_t and the reconstructed image \hat{I}_t a photometric loss term defined as the mean of the absolute value of the difference of pixel intensities of the two images was used.

$$\mathcal{L}_p(I_t, \hat{I}_t) = \text{mean}(|I_t - \hat{I}_t|)$$

Another photometric loss term evaluated in this thesis is based on structured similarity, referred to as SSIM[16]. It was originally developed to measure the quality of digital television, comparing a compressed digital image to the original distortion-free image.

$$\mathcal{L}_{ssim}(I_t, \hat{I}_t) = \text{clamp}(0, 1, \frac{1 - \text{SSIM}(I_t, \hat{I}_t)}{2})$$

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

with $C_1 = 1e-4$ and $C_2 = 9e-4$. To compute the per-patch mean μ_x and standard deviation σ_x I first used a 1 pixel reflection padding on the edges of the input images and then a 3×3 average pool filter with stride 1 to get the mean. Then $\sigma_x = \mu_{x^2} - \mu_x^2$.

The two above mentioned photometric loss terms can also be combined and balanced using $\mathcal{L}_{ps}(I_t, \hat{I}_s) = \alpha \mathcal{L}_{ssim} + (1 - \alpha) \mathcal{L}_p$. In my experiments I used $\alpha = 0.85$.

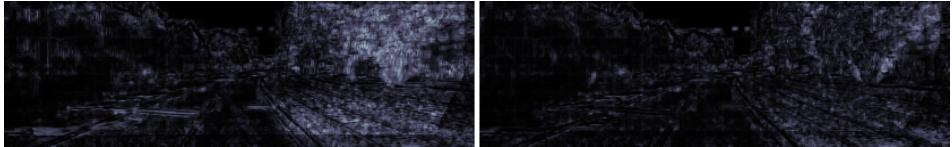


Figure 5.2: The images illustrates the photometric reconstruction loss \mathcal{L}_{ps} with $\alpha = 0.85$ for each pixel in the reconstructed image. The left image shows the loss after 1 epoch of training and the right image shows the loss after 30 epochs. The reconstruction loss should decrease during training as the network learns to predict better depth maps, which is what we see.

To propagate the depth from textured regions to regions of uniform color a depth smoothness loss term is used. The first alternative is a loss on the second derivative of the depth intensities.

$$\mathcal{L}_{smooth}(D_t) = \text{mean}(|\delta_x^2 D_t| + |\delta_y^2 D_t|)$$

I also ran experiments applying the smoothness loss on the disparity map instead of the depth map, but with worse performance. The second alternative investigated was an edge aware smoothness loss that weights the first order derivative of the depth map with the exponential of the first derivative of the pixel intensities. This method allows large changes in depth near sharp features in the image but penalizes changes in depth in smooth regions. The method showed great promise in the Monodepth2 paper, resulting in sharper edges because the smoothness term is weighted to mostly affect areas with small photometric derivative.

$$\mathcal{L}_{edge}(D_t) = \text{mean}(|\delta_x D_t| e^{-|\delta_x I_t|} + |\delta_y D_t| e^{-|\delta_y I_t|})$$

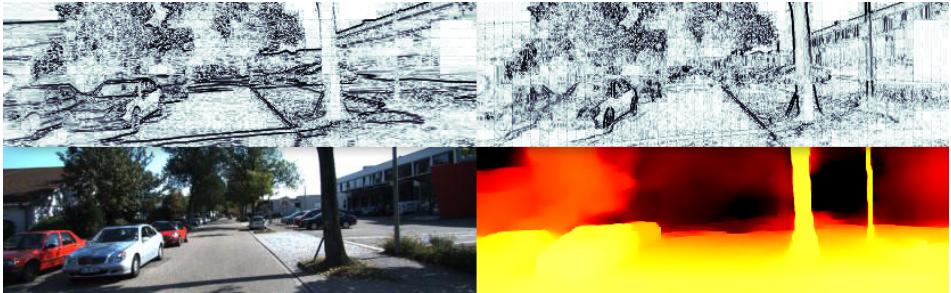


Figure 5.3: The top row of images illustrates the edge weighting terms $e^{-|\delta_y I_t|}$ and $e^{-|\delta_x I_t|}$ respectively. The weight is near 0 at the edges of tree trunks and near 1 on the pavement. This will preserve the details in the depth map around trees but keep the pavement smooth.

5.0.4 Handling occlusions

In the SfMLearner paper 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 alleviates this issue.



Figure 5.4: By picking the per pixel minimum reprojection error the issue created by occluded pixels can be alleviated. The top image is the previous frame, the middle image is the current target frame and the bottom image is the next frame in the sequence. If the minimum reprojection error can be found in the previous frame then it is colored blue, if its from the next frame it is green. Because the door to the right is occluded by the orange truck in the previous frame, the reprojection loss from the next frame is used instead where the door is visible. The wall to the left is outside the boundaries of the next image, so the reprojection error from the previous frame is used instead.

5.0.5 Handling model limitations

In order to optimize using the photometric reprojection 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 [15] 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 off 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.

TODO: Debug why this is not working as well as expected and then insert image showing the explainability mask....

Stationary pixels mask The authors of [6] 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.



Figure 5.5: The model assumes a stationary world without moving objects in it. By excluding stationary pixels from the loss issues arising from moving objects can be alleviated. The black pixels in the image are the ones removed because their photometric error is smaller before warping the image using the depth map compared to after warping. The red horizontal lines on the van and bicyclist illustrates that they do not move with respect to the camera, and the green slanted line illustrates that the back bike tire is moving with respect to the camera. The mask successfully removes pixels on the van and bicyclist, but also removes some pixels on the pavement that should not be removed.

5.1 Unsupervised keypoint learning

In order to build a map it is useful to extract features, or keypoints, from images. Every keypoints should have a unique descriptor which can be used to identify it in the map which makes it possible to expand the map and also localize the camera. How to build a map and localize in it is out of scope for this thesis. This section will describe an unsupervised learning method to extract keypoints from images that would be usable in a full SFML pipeline.

The method evaluated in this thesis is based on the UnsuperPoint paper [1] from 2019. The network architecture is illustrated in Figure 5.6. The input image is fed into a shared backbone. The features from the backbone are then split into three different encoders that estimate the score, position and a descriptor for each keypoint. The network will estimate a keypoint in every 8×8 patch of the image, but only the top N keypoints sorted by score are used in the evaluation.

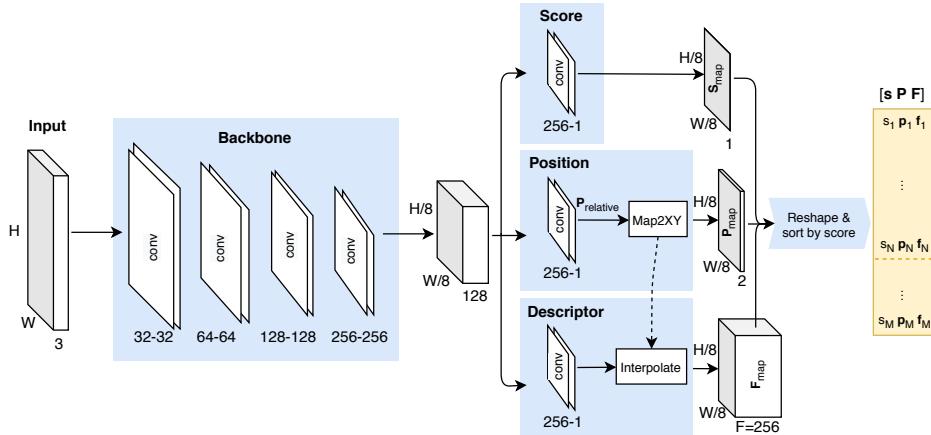


Figure 5.6: The network has a common backbone and then splits into separate score, position and descriptor encoders. The output is reshaped and sorted by descending score.

The network is trained in a siamese twin setup, where two duplicate networks that share weights are fed different inputs and the loss is formulated by comparing the output scores, positions and descriptors.

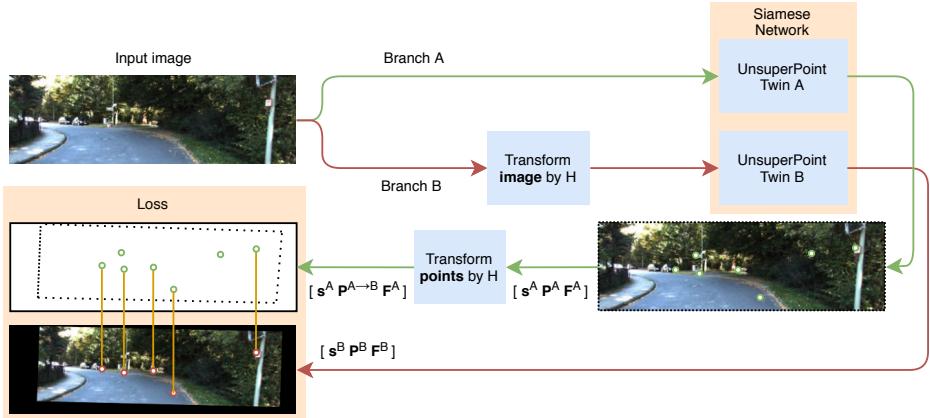


Figure 5.7: This figure illustrates the flow from input image to loss function. The image from branch A is fed directly into the UnsuperPoint network, while in branch B the image is first transformed by a random homography H . The keypoint positions from twin A are transformed by the same homography H and the output from the two branches are compared in order to formulate the loss function.

The score encoder is terminated by a sigmoid function and therefore predicts scores $s_n \in \mathbb{R}$ between 0 and 1 for each keypoint. The purpose of the scores are to rank the quality of the keypoints in all 8×8 px patches of the image and only pick the best ones. Typically patches in the sky and other non-textured areas will have keypoints with low scores.

The position encoder is also terminated by a sigmoid function and predicts the relative position of the keypoint in each patch. In the Map2XY block in Figure 5.6 the relative positions are converted to absolute pixel positions.

$$P_{map,x}(r, c) = 8 * (c + P_{relative,x}(r, c))$$

$$P_{map,y}(r, c) = 8 * (r + P_{relative,y}(r, c))$$

The descriptor encoder predicts a descriptor vector of length $F = 256$ for each keypoint. The purpose of the descriptor is to find corresponding keypoints in different images. The constitutional layers produce 1 descriptor vector for each 8×8 patch. This vector can be used directly as the keypoint descriptor and it works pretty well. Even better results can be achieved using the absolute keypoint positions to sample the values in the descriptor map with the same interpolation method used to do the warping in Figure 5.1.

5.1.1 Loss function for keypoint learning

TODO: Describe briefly the loss terms used....

Corresp G

pairs hat

dk

lposition

lsim

lscore

lunixy

ldesc

ldecorr

5.2 Unsupervised consensus maximization

Consensus maximization in a set of features related by a geometric constraint that also contains outliers is an important aspect of an SFM system. The problem consists of finding the largest set of inlier features in a set that is consistent with a geometric constraint. In a 3D vision setting the geometric constraint could be for example a rigid transformation $[R \ t]$, a fundamental matrix F or a homography H that relates pairs of corresponding 3D or 2D points.

A popular algorithm to solve this problem is RANSAC[3], but is however not differentiable and therefore not learnable by a neural network. In this paper[?] from 2019 the authors present a differential method to learn consensus maximization in an unsupervised setting. The explanation of the method in the paper is however very theoretical and non-accessible to master level students. In this section I will try to explain the method with emphasis on ease of understanding and with the mathematical toolbox of an engineering student. I will focus on the method used to extract the parameters of the constraint, as it's not explained by the paper in detail and took a lot of figuring out on my part, especially how to extract the homography.

Consider one of the simplest geometric constraint on a set of corresponding 2D points in subsequent frames of the KITTI dataset, the fundamental matrix F . Given a point u in frame 1, and point v in frame 2 they are related in the following way.

$$u^T F v = 0$$

To be more explicit the relation can be expressed as follows.

$$\begin{pmatrix} u_x & u_y & 1 \end{pmatrix} \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix} = 0$$

Multiplication of the two rightmost matrices gives:

$$\begin{pmatrix} u_x & u_y & 1 \end{pmatrix} \begin{pmatrix} f_{11}v_x + f_{12}v_y + f_{13} \\ f_{21}v_x + f_{22}v_y + f_{23} \\ f_{31}v_x + f_{32}v_y + f_{33} \end{pmatrix} = 0$$

Multiplication of the remaining two matrices and grouping the monomials in parentheses for readability gives the following linear equation.

$$f_{11}(u_x v_x) + f_{12}(u_x v_y) + f_{13}(u_x) + f_{21}(u_y v_x) + f_{22}(u_y v_y) + f_{23}(u_y) + f_{31}(v_x) + f_{32}(v_y) + f_{33}(1) = 0$$

Now consider a set of m corresponding points that are related by the same fundamental matrix F . Place the 9 monomials in the parentheses for all m points in a matrix $M \in \mathbb{R}^{m \times 9}$. A matrix of monomials, such as this one, is called a Vandermonde matrix.

$$M = \begin{pmatrix} u_{x,1}v_{x,1} & u_{x,1}v_{y,1} & u_{x,1} & u_{y,1}v_{x,1} & u_{y,1}v_{y,1} & u_{y,1} & v_{x,1} & v_{y,1} & 1 \\ & & & & \vdots & & & & \\ u_{x,m}v_{x,m} & u_{x,m}v_{y,m} & u_{x,m} & u_{y,m}v_{x,m} & u_{y,m}v_{y,m} & u_{y,m} & v_{x,m} & v_{y,m} & 1 \end{pmatrix}$$

The constraint is satisfied by all m points if

$$M \begin{pmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}^{mx1}$$

Now the solution \mathbf{f} should look very familiar. The column vector \mathbf{f} is in fact the very definition of a nullspace of M . This is the key insight in this method of unsupervised consensus maximization. Finding the nullspace of M is done

by performing a singular value decomposition, which is a differentiable operation.

$$U\Sigma V^T = M$$

The nullspace will be the last column vector of $V \in \mathbb{R}^{9 \times 9}$ and the last singular value σ_9 in the diagonal of Σ will be zero. This is correct only if the set of corresponding points contains only inliers. If the set contains outliers that can not be related by the geometric constraint then the last singular value σ_9 will be greater than zero. If the inliers and outliers were known they could be represented by a vector $w \in [0, 1]^m$ where 1 means inlier and 0 means outlier. Now M can be weighted by w which would cancel all rows in M which comes from an outlier pair of points.

$$\text{diag}(w)Mf = 0$$

The problem of finding the largest set of inliers has now been transformed into finding the weights w such that the last singular value σ_9 of $\text{diag}(w)M$ is near zero, and the sum of the weights should also be large to have as many inliers as possible.

To learn the weight $w \in w$ for each pair of points, the PointNet[13] architecture was used. The pair of points are concatenated into rows of a $X \in \mathbb{R}^{mxn}$ matrix. For 2D points $n = 2 + 2 = 4$ and for 3D points $n = 3 + 3 = 6$. The input X is fed through the PointNet segmentation network with a single output class, signifying if the point pair is an inlier or not. The key innovation of the PointNet architecture is that it can directly consume point cloud data, as opposed to first transforming the input to a 3D voxel grid or an image. The network is also invariant to the ordering of the points in the input list. Figure 5.8 illustrates the process of feeding the input through pointnet to get the weights, and then applying the weights to the Vandermonde matrix of which we want to minimize the last singular value.

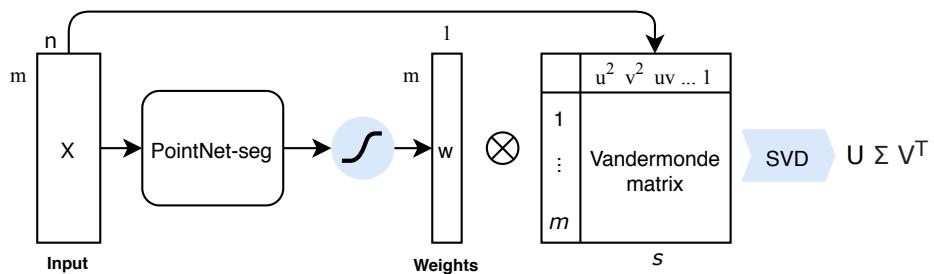


Figure 5.8

Because the fundamental matrix is a constraint expressed by 1 linear equation a basis of only 1 nullspace vector is needed from the singular value decomposition to construct it. But the method can also be used to learn constraints such as a homography or rigid 3D transformation that are constrained by 3 linear equations and therefore require a basis of 3 nullspace vectors. In that case the last 3 singular values from the SVD should be minimized in the loss function.

In the following sections the method used to extract the homography from the basis vectors will be explained. Two points that are related by a homography is expressed as follows. The monomials are grouped by parentheses.

$$u \times Hv = 0 \rightarrow$$

$$[u]_{\times} Hv = 0 \rightarrow$$

$$\begin{pmatrix} 0 & -1 & u_y \\ 1 & 0 & -u_x \\ -u_y & u_x & 0 \end{pmatrix} \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \rightarrow$$

$$\begin{cases} -h_{21}(v_x) - h_{22}(v_y) - h_{23}(1) + h_{31}(v_x u_y) + h_{32}(v_y u_y) + h_{33}(u_y) = 0 \\ h_{11}(v_x) + h_{12}(v_y) + h_{13}(1) - h_{31}(v_x u_x) - h_{32}(v_y u_x) - h_{33}(u_x) = 0 \\ -h_{11}(v_x u_y) - h_{12}(v_y u_y) - h_{13}(u_y) + h_{21}(v_x u_x) + h_{22}(v_y u_x) + h_{23}(u_x) = 0 \end{cases}$$

The monomials are the same as for the fundamental constraint, which means that the same Vandermonde matrix M can be used. But because an homography is constrained by $r = 3$ linear equations the last 3 singular values of the SVD should be minimized during training.

Extract 3 basis vectors from the nullspace which will be the 3 rightmost columns of V in the SVD of $\text{diag}(\mathbf{w})M$.

$$B = (\mathbf{v}_7 \quad \mathbf{v}_8 \quad \mathbf{v}_9) \in \mathbb{R}^{9 \times 3}$$

The 3 linear equations of the homography are tangled which means that the elements of B can not be directly mapped onto the elements in H . Notice that the first equation of H does not contain u_y and the second equation of H does not contain u_x . We can exploit this fact to perform a change of basis from $B \in \mathbb{R}^{9 \times 3}$ to $B' \in \mathbb{R}^{9 \times 2}$ that should have the following structure.

Monomial	Structure of B'	Corresponding element in H
$u_x v_x$	$\begin{pmatrix} 0 & \cdot \\ 0 & \cdot \\ 0 & \cdot \end{pmatrix}$	$\begin{pmatrix} h_{31} \\ h_{32} \\ h_{33} \end{pmatrix}$
$u_x v_y$		
u_x		
$u_y v_x$	$\begin{pmatrix} \cdot & 0 \\ \cdot & 0 \\ \cdot & 0 \end{pmatrix}$	$\begin{pmatrix} h_{31} \\ h_{32} \\ h_{33} \end{pmatrix}$
$u_y v_y$		
u_y		
v_x	$\begin{pmatrix} \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \end{pmatrix}$	$\begin{pmatrix} h_{21} & h_{11} \\ h_{22} & h_{12} \\ h_{23} & h_{13} \end{pmatrix}$
v_y		
1		

To perform the change of basis to get the structure of B' we use the nullspace $\mathbf{n}_1 \in \mathbb{R}^{3 \times 1}$ of rows 1, 2 and 3 from B , and the nullspace $\mathbf{n}_2 \in \mathbb{R}^{3 \times 1}$ of row 4, 5, 6 from B .

$$A_1 = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix}, A_2 = \begin{pmatrix} b_{41} & b_{42} & b_{43} \\ b_{51} & b_{52} & b_{53} \\ b_{61} & b_{62} & b_{63} \end{pmatrix}$$

\mathbf{n}_1 = "rightmost column of right-singular vectors of A_1 "

\mathbf{n}_2 = "rightmost column of right-singular vectors of A_2 "

$$\mathbf{b}^{n_1} = B\mathbf{n}_1$$

$$\mathbf{b}^{n_2} = B\mathbf{n}_2$$

Now the \mathbf{b}^{n_1} and \mathbf{b}^{n_2} basis vectors will have the following structure as desired.

$$\mathbf{b}^{n_1} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{pmatrix}, \mathbf{b}^{n_2} = \begin{pmatrix} \cdot \\ \cdot \\ 0 \\ 0 \\ 0 \\ \cdot \\ \cdot \end{pmatrix},$$

The basis vectors have zeros at the correct place. The next step is to adjust the scale so that row 4, 5 and 6 of \mathbf{b}^{n_1} and row 1, 2 and 3 of \mathbf{b}^{n_2} have the same norm because they both represent the same elements h_{31} , h_{32} and h_{33} . We also make sure they have the same sign.

$$s = \text{sign}(b_1^{n_2} + b_2^{n_2} + b_3^{n_2}) \text{sign}(b_4^{n_1} + b_5^{n_1} + b_6^{n_1})$$

s will be -1 if they have different signs, or 1 if they are the same sign.

$$B' = \left(\mathbf{b}^{n_1} / \| (b_4^{n_1} \ b_5^{n_1} \ b_6^{n_1}) \| \quad \mathbf{b}^{n_2} / \| (b_1^{n_2} \ b_2^{n_2} \ b_3^{n_2}) \| \ s \right) \in \mathbb{R}^{9 \times 2}$$

It is now possible to assign the elements of H by pattern matching.

$$H = \begin{pmatrix} b'_{72} & b'_{82} & b'_{92} \\ b'_{71} & b'_{81} & b'_{91} \\ -b'_{41} & -b'_{51} & -b'_{61} \end{pmatrix}$$

5.2.1 Improving training convergence

TODO: Describe changes I had to make compared to the original paper in order to achieve convergence during training....

6

Results

This chapter presents the results from the experiments conducted to evaluate the performance of the unsupervised learning methods investigated in this thesis.

6.1 Depth and ego motion

C	Net	DS	Edge	Norm	Expl	Stat	SSIM	Comb	US
1	SL	K						avg	
2	SL	K			x			avg	
3	SL	K				x		avg	
4	SL	K	x			x		avg	
5	SL	K	x			x	x	min	
6	SL	K	x	x		x	x	min	
7	SL	K	x	x		x	x	min	x
8	SL	L	x	x		x	x	min	x
9	M2	K				x		avg	
10	M2	K	x	x		x	x	min	
11	M2	K	x	x		x	x	min	x
12	M2	L	x	x		x	x	min	x

Table 6.1: All configurations of the depth and ego motion network that were evaluated. The C column identifies the specific configuration. The Net column shows which architecture was used, either SL for SfmLearner or M2 for Monodepth2. The DS column shows which dataset was used during training, either K for Kitti or L for Lyft. The Edge column indicates if the edge aware smoothing loss term \mathcal{L}_{edge} was used. The Norm column indicates if depth map normalization was used. The Expl column indicates if an explainability mask was used. The Stat column indicates if a mask to remove stationary pixels from the loss was used. The SSIM column indicates if \mathcal{L}_{ps} was used, otherwise just \mathcal{L}_p . The Comb column shows which methods was used to combine the loss terms from the two source images, either the average or the minimum loss across frames. The US column indicates that up scaling of the depth maps in the pyramid was used, otherwise the target frame was down scaled to match the size of the smaller depth maps in the pyramid.

E	C	DS	AbsRel	SqRel	RMSE	RMSLE	1.25	1.25 ²	1.25 ³	Ego
1	1	K	0.174	1.405	4.829	0.249	0.784	0.920	0.964	0.024
2	2	K	0.184	1.718	4.953	0.258	0.776	0.915	0.960	0.020
3	3	K	0.140	0.793	4.549	0.217	0.818	0.939	0.976	0.020
4	4	K	0.143	0.819	4.708	0.222	0.810	0.937	0.975	0.022
5	5	K	0.133	0.727	4.305	0.204	0.843	0.950	0.979	0.023
6	6	K	0.137	0.797	4.282	0.208	0.837	0.948	0.977	0.024
7	7	K	0.135	0.778	4.248	0.208	0.841	0.948	0.997	0.020
8	7	L	0.340	7.811	23.071	0.447	0.457	0.734	0.868	0.043
9	8	K	0.512	5.185	10.757	0.611	0.305	0.549	0.732	0.495
10	8	L	0.739	20.236	31.859	0.773	0.240	0.434	0.587	1.324
11	9	K	0.125	0.697	4.298	0.203	0.845	0.948	0.979	0.021
12	10	K	0.126	0.714	4.018	0.194	0.860	0.958	0.982	0.019
13	11	K	0.132	0.769	3.966	0.196	0.859	0.957	0.981	0.019
14	11	L	0.304	7.019	21.907	0.414	0.518	0.775	0.886	0.042
15	12	K	0.322	3.177	7.179	0.378	0.549	0.797	0.906	0.036
16	12	L	0.303	8.051	19.312	0.385	0.637	0.827	0.906	0.059

Table 6.2: All the experiments measuring the performance of the different configurations in Table 6.1. The E column identifies a specific experiment. The C column shows which configuration was used. The DS column shows which Dataset was used during testing, K for Kitti and L for Lyft. The dataset used during testing differs from the one used during training in some experiments. The AbsRel, SqRel, RMSE and RMSLE columns are the depth error metrics described in section 4.3.1, smaller is better. The 1.25, 1.25² and 1.25³ columns are the depth accuracy metrics described in section 4.3.1, larger is better. The Ego column is the camera ego motion error metric described in section 4.3.2.

TODO: Describe the improvements (if any) seen by applying each additional method for depth prediction and explain why this could be....

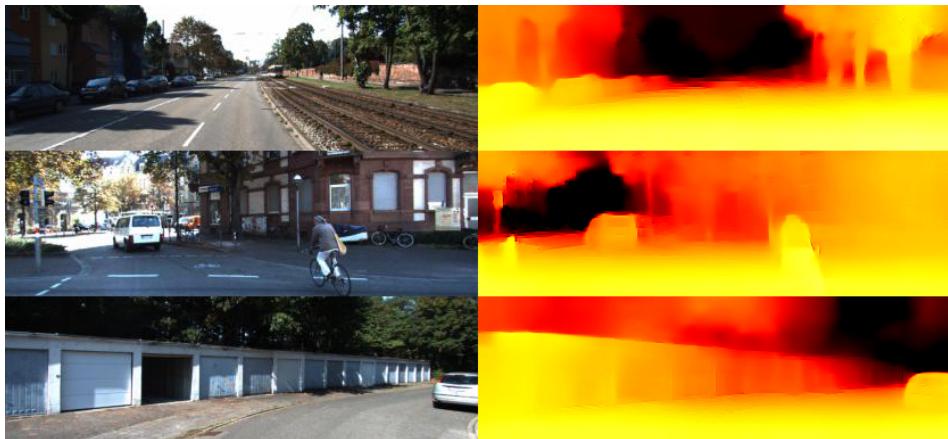


Figure 6.1: Examples from the Kitti dataset



Figure 6.2: 3D render of colorized depth map

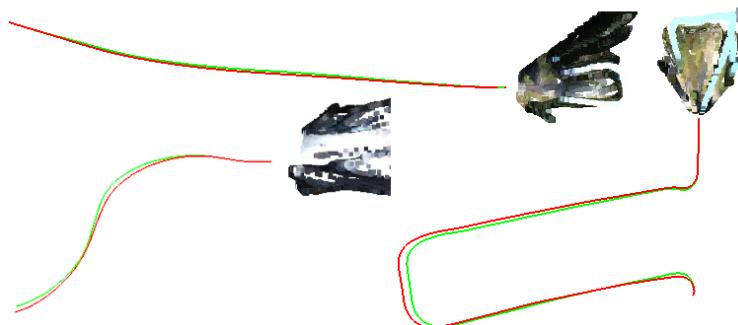


Figure 6.3: 3D visualization of the camera movement in three different image sequences. The green lines are the ground truth and the red lines are the predicted camera trajectories.



Figure 6.4: 3D visualization from 3 different angles of the same frame in the kitti dataset.

6.2 Keypoint detection



Figure 6.5: Results from the keypoint prediction network. The top row are the original input images fed to branch A, and below are the transformed images fed to branch B. Circles that are the same color have matching descriptors.

Method	RS \uparrow	LE \downarrow	MS \uparrow	CMR \uparrow	Nt	Nm
UnsuperPoint	0.796	0.666	0.488	0.834	338	203
ORB	0.841	0.764	0.302	0.564	310	171

Table 6.3: Experiments measuring the performance of keypoint detection methods.

TODO: Describe what the results are showing....

6.3 Consensus maximization

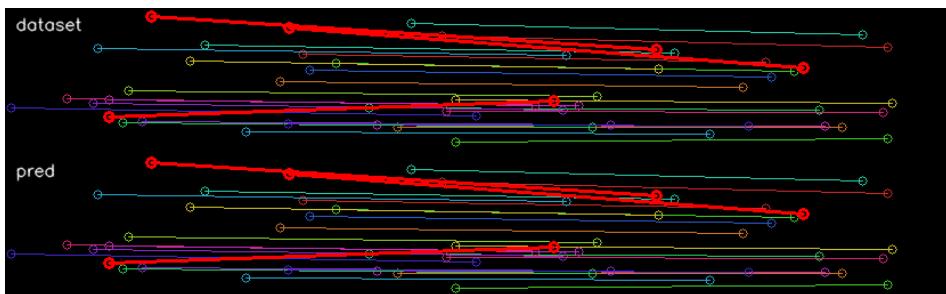


Figure 6.6: Outlier detection in homography estimation in synthetic dataset. Top row is ground truth and bottom row is the prediction.

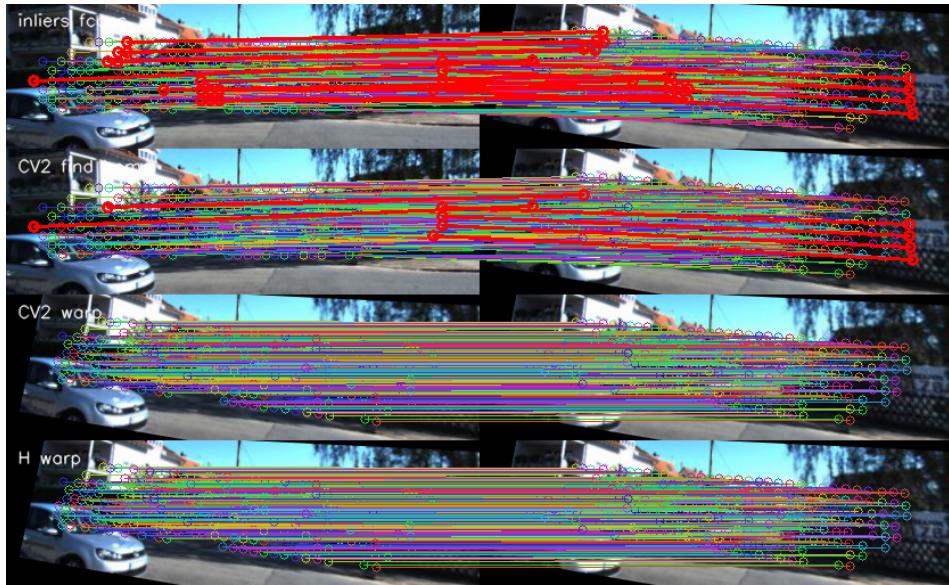


Figure 6.7: Outlier detection and homography estimation on images from kitti dataset. First row is from OpenCV `findHomography()`, second row is from the network, third row is the image from branch A transformed by the homography found by OpenCV, fourth row is the image from branch A transformed by the homography found by the network.

Method	$HE \downarrow$
ConsensusNet	???
RANSAC	???

Table 6.4: Metrics for homography estimation.

		Predicted		Total
		Inlier	Outlier	
Actual	Inlier	TP	FN	$TP + FN$
	Outlier	FP	TN	$FP + TN$
		Total	$TP + FP$	$FN + TN$
				N

Table 6.5: Confusion matrix for inlier/outlier prediction using Consensus-Net.

		Predicted		Total
		Inlier	Outlier	
Actual	Inlier	TP	FN	$TP + FN$
	Outlier	FP	TN	$FP + TN$
Total		$TP + FP$	$FN + TN$	N

Table 6.6: Confusion matrix for inlier/outlier prediction using RANSAC.

TODO: Insert actual numbers from the testing code and describe what the results are showing....

7

Discussion

7.1 Results

The best combination for depth and ego motion predictions.

7.2 Method

Small disparity, large depth, for cars moving with the same relative speed, no ego motion.

TODO: Image of holes from small disparity

Over-fitting while training homography.

TODO: Plot over fitting. Image + Loss + Histogram

Alterations needed to make consensus converge.

7.3 The work in a wider context

TODO: I dont know what to write here....

8

Conclusions

Återkoppling till syfte och frågeställningar. Konsekvenser för målgrupp.

How well do the unsupervised methods from previous research work on new datasets not tested in the original papers? Hur har jag svarat på denna fråga....
Hur har jag svarat på denna fråga.... Hur har jag svarat på denna fråga....

What are the performance gains of combining different methods from recent research in monocular depth and ego motion prediction? Hur har jag svarat på denna fråga.... Hur har jag svarat på denna fråga.... Hur har jag svarat på denna fråga....

8.1 Future work

TODO: Describe future work to couple the training of depth, keypoints and consensus more tightly where all subsystems could benefit from each other....

Bibliography

- [1] Peter Hviid Christiansen, Mikkel Fly Kragh, Yury Brodskiy, and Henrik Karstoft. Unsuperpoint: End-to-end unsupervised interest point detector and descriptor. *CoRR*, abs/1907.04011, 2019. URL <http://arxiv.org/abs/1907.04011>.
- [2] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. 12 2017.
- [3] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. ISSN 0001-0782. doi: 10.1145/358669.358692.
- [4] 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.
- [5] Clement Godard, Oisin Aodha, and Gabriel Brostow. Unsupervised monocular depth estimation with left-right consistency. 07 2017. doi: 10.1109/CVPR.2017.699.
- [6] Clément Godard, Oisin Aodha, and Gabriel Brostow. Digging into self-supervised monocular depth estimation, 06 2018.
- [7] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, and Adrien Gaidon. Packnet-sfm: 3d packing for self-supervised monocular depth estimation, 05 2019.
- [8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. 7, 12 2015.
- [9] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, 06 2015.

- [10] 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. url`https://level5.lyft.com/dataset/`, 2019.
- [11] 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.
- [12] 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.
- [13] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016.
- [14] Jie Xiong Tang, Rares Ambrus, Vitor Guizilini, Sudeep Pillai, Hanme Kim, and Adrien Gaidon. Self-supervised 3d keypoint learning for ego-motion estimation, 12 2019.
- [15] Tinghui Zhou, Matthew Brown, Noah Snavely, and David Lowe. Unsupervised learning of depth and ego-motion from video. 04 2017.
- [16] 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, 2004.