ALMA MATER STUDIORUM - UNIVERSITA DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA

Dipartimento di Informatica - Scienza e Ingegneria - DISI
Corso di Laurea Magistrale in Ingegneria Informatica

TESI DI LAUREA

in

EMBEDDED SYSTEMS

Monocular Depth Estimation enhancement by depth from SLAM Keypoints

CANDIDATO
Lorenzo Andraghetti

RELATORE
Prof. Stefano Mattoccia

CORRELATORI
PhD. Matteo Poggi
PhD. Alessandro Pieropan

Anno Accademico 2017/18
Sessione II

ALMA MATER STUDIORUM - UNIVERSITA DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA

Dipartimento di Informatica - Scienza e Ingegneria - DISI
Corso di Laurea Magistrale in Ingegneria Informatica

TESI DI LAUREA

in

EMBEDDED SYSTEMS

Monocular Depth Estimation enhancement by depth from SLAM Keypoints

CANDIDATO
Lorenzo Andraghetti

RELATORE
Prof. Stefano Mattoccia

CORRELATORI
PhD. Matteo Poggi
PhD. Alessandro Pieropan

Anno Accademico 2017/18
Sessione II

Ai miei nonni e ai miei genitori.
I miei pilastri.

Contents

1 Introduction **13**
1.1 What is a depth map? . . . 13
1.1.1 How to get depth . . . 13
1.2 The unsupervised choice and SLAM exploitation . . . 15
1.3 Application of depth sensing technology . . . 16
1.4 Thesis Structure . . . 16

2 Related Work **17**

3 Monodepth **21**
3.1 Approach . . . 21
3.2 Network Structure . . . 23
3.3 Losses . . . 24
3.4 Implementation Details . . . 25
3.5 Post-Processing . . . 26

4 Experimentation **27**
4.1 Employing ground truth . . . 27
4.2 RGBD and RGBDM models . . . 29
4.3 Sparse convolutions models . . . 31
4.3.1 Sparse Convolutions . . . 31
4.3.2 Double-Encoder model . . . 33
4.3.3 Autoencoder models . . . 33
4.3.4 Results . . . 35
4.3.5 Comparison with Monodepth . . . 36

5 ORB-SLAM **39**
5.1 SLAM . . . 39
5.2 ORB-SLAM . . . 39
5.3 Obtain sparse disparity maps from ORB-SLAM . . . 40

6 Evaluation and Results **43**
6.1 Evaluation Metrics . . . 43
6.2 Cityscapes . . . 45
6.3 KITTI . . . 46
6.3.1 KITTI 2015 . . . 46
6.3.2 Eigen Split . . . 48
6.3.3 Odometry Sequence 03 . . . 49
6.4 Evaluation on Sparse Input . . . 50

7 Conclusions and further developments **55**

References **57**

A Evaluation Tools **61**
A.1 Sparse Test . . . 61
A.2 Show Disparities . . . 63

B More qualitative results **65**

C UVR ZED dataset **69**
C.1 Creating the Dataset . . . 69
C.2 Fine-tuning . . . 71

D Demo with Pangolin Visualizer **75**

Abstract

Training a neural network in a supervised way is extremely challenging since ground truth is expensive, time consuming and limited. Therefore the best choice is to do it unsupervisedly, exploiting easier-to-obtain binocular stereo images and epipolar geometry constraints. Sometimes however, this is not enough to predict fairly correct depth maps because of ambiguity of colour images, due for instance to shadows, reflective surfaces and so on. A Simultaneous Location and Mapping (SLAM) algorithm keeps track of hundreds of 3D landmarks in each frame of a sequence. Therefore, given the base assumption that it has the right scale, it can help the depth prediction providing a value for each of those 3D points. This work proposes a novel approach to enhance the depth prediction exploiting the potential of the SLAM depth points to their limits.

Sommario

Allenare una rete neurale in modo supervisionato è estremamente impegnativo poiché la ground truth è costosa in termini di spesa e di tempo ed è spesso limitata. Si preferisce quindi allenare in modo non-supervisionato sfruttando immagini stereo, più facili da ottenere, impiegando vincoli geometrici. Talvolta però, questo non è sufficiente per predizioni corrette poiché le immagini spesso contengono ambiguità dovute ad ombre, superfici riflettenti e così via. Un algoritmo di Simultaneous Location and Mapping (SLAM) tiene traccia di centinaia di marcatori 3D in ogni immagine di una sequenza video. Data l'assunzione base che abbia la scala di profondità corretta, questo algoritmo può aiutare nella predizione delle mappe di profondità fornendo un valore per ognuno di questi punti 3D. Questo studio propone un nuovo metodo per migliorare la predizione delle mappe di profondità sfruttando al massimo il potenziale dei punti provenienti da SLAM.

Chapter 1

Introduction

1.1 What is a depth map?

A depth map is an image same height and width as a photograph, but in every pixel it's known how many meters away an object is. Depth is crucial when it comes to understanding the world around us, such as navigation and mapping. That's very helpful for sensing tasks such as composing things like in augmented reality or robots that need to figure out how to navigate through a scene or how to grasp an object. When the robot walks around an object, it starts to accumulate a whole 3D shape of it, not just the depth from one side, but actually the volume of the whole object which it is filming.

1.1.1 How to get depth

To capture depth using video cameras, normally two of them would be needed, displaced horizontally from one another and pointing at the same direction, like our eyes. That displacement is called baseline.

Let's say for instance that we have a stereo camera pointing at the tree in Figure 1.1. Left and right images will be different, showing the tree and the person with some displacement. If it's possibile to solve for correspondence between these two images, then we essentially get disparity which is inversely

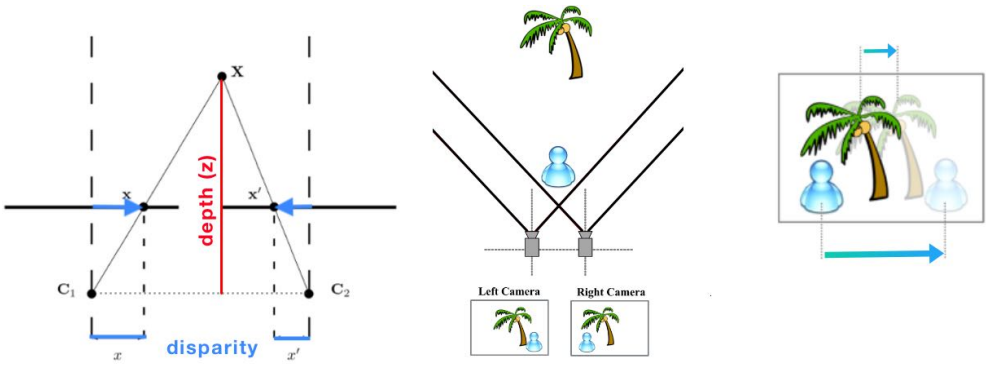

Figure 1.1: Disparity is simple to experience, closing one eye and then rapidly close it while opening the other. Objects that are close will appear to jump a significant distance while objects further away will move very little. That motion is **disparity**.

proportional to depth:

$$disparity \propto \frac{1}{depth} \tag{1.1}$$

Another option would be to have a laser scanner (e.g. Lidar), which basically use the time of travel to estimate a depth value for every point where the laser hits. This method is however very expensive and time consuming and it would be moreover impossibile to have one of those systems in every car. Since it's dependent on laser beams, the result would be a sparse depth map, which is not enough for navigation and mapping, especially since it has unreliable data for moving and reflective objects. Similarly, structured light sensor such as Microsoft Kinect have limited range and do not work well outside.

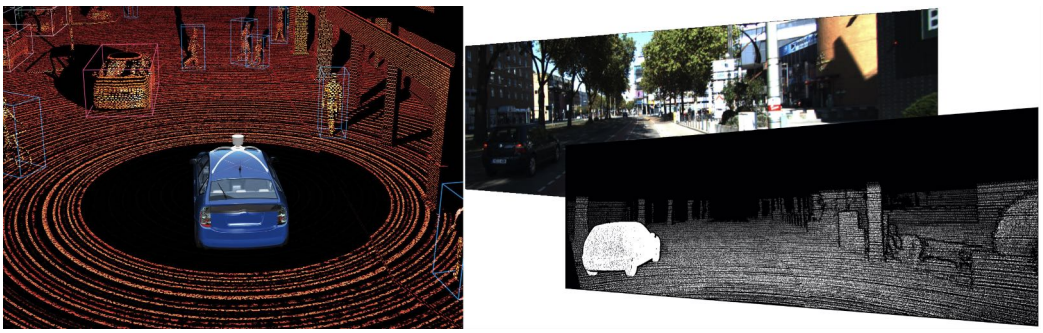

Figure 1.2: On the left, the result of a lidar 360 depth sensing. On the right, colour image and sparse disparity map from Velodyne Lidar data in KITTI 2015 test set.

The technology that we are going to present would just use a regular colour camera. The images that could be captured from that can be turned into depth images, which tell how far away every object on the street is. The core of this technology is a neural network that can predict the depth from just one image, avoiding a lot of efforts on depth sensing in various situations.

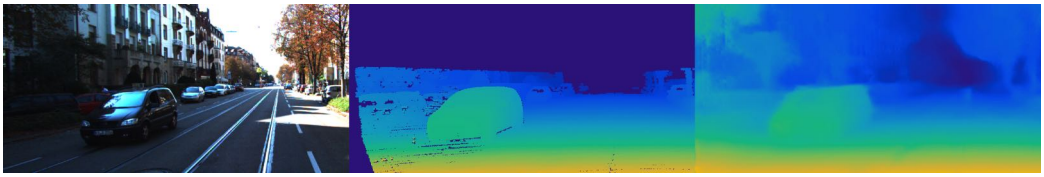

Figure 1.3: Starting from the left: RGB image, Lidar and CNN predicted disparity map. The Lidar map is sparse and lacks the upper part on the image. The CNN disparity map is dense but less precise than the Lidar one.

1.2 The unsupervised choice and SLAM exploitation

Most of the old approaches to depth prediction are supervised regression problems and therefore they need almost perfect ground truth depth data during training. This task is extremely challenging and expensive, thus the best choice is to train a neural network in a unsupervised way, exploiting easier-to-obtain binocular stereo images and epipolar geometry constraints. However, sometimes this is not enough to predict fairly correct depth maps because of ambiguity of colour images due for example to shadows, reflective surfaces and so on. A Simultaneous Location and Mapping (SLAM) algorithm keeps track of hundreds of 3D landmarks[1] in each frame of a sequence, therefore if we keep the base assumption that it has the right scale, it can help the depth prediction providing a value for each of those 3D points. This work proposes a novel way to enhance and exploit the potential of the SLAM depth points to their limits.

[1]Landmarks are features which can easily be re-observed and distinguished from the environment.

1.3 Application of depth sensing technology

Self-driving cars are obviously a good application of this technology, but also augmented reality and even the medical field. Endoscopy for instance has a setup that is physically restrictive and there's no room to put a stereo camera or laser scanner at the end of the endoscope. Specifically for the self-driving scenario, a good application of this technology might be where the cameras are on each edge of the car and every one of them could be a depth sensor. This system can estimate the size of an object and how far away it is. Then the car might be able to react to that and anticipate the motion of other vehicles, know when to merge, when to veer or to navigate some other direction.

1.4 Thesis Structure

This work will be presented starting from a list of all the related work in Chapter 2, followed by a full explanation of Monodepth model by Godard et al. [7] in Chapter 3. In Chapter 4 all the approaches to exploit sparse depth input will be shown along with the results. In Chapter 5 will be explained how ORB-SLAM algorithm works and how it has been possibile to get depth from it. In Chapter 6 there will be a complete report about everything that concerns evaluation and the results on different datasets. Chapter 7 will present conclusions and further developments.

Appendix A shows all the tools created for evaluation.
Appendix B shows the more qualitative results on datasets.
Appendix C explains why and how the UVR ZED dataset was created.
Appendix D shows how the demo with Pangolin visualizer has been created.

Chapter 2

Related Work

There's a substantial amount of papers aiming to estimate depth from a single image. This means that the neural network should need only one image to predict the depth. Although, it needs more informations to learn inferring depth from images at training time.

The first approaches were supervised and they needed indeed ground truth data to enforce the network to infer depth. Saxena et al. [20] proposed a method to estimate the absolute scales of different image patches and inferred the depth image using a Markov Random Field model. This approach focuses on local context, thus it lacks global context required to generate realistic outputs and it needs moreover to hand-tune some parameters to do so. Liu et al. [14] trained a CNN to learn how to handle those values. Ladický et al. [12] obtained better results incorporating semantic segmentation into their model. Eigen et al. employed two deep network stacks: one that makes a global prediction based on the entire image, and another that iteratively refines this prediction locally. Unlike before, they do not rely on hand crafted features but instead learn directly from the raw pixel values.

More recent works focus on unsupervised depth estimation with limited resource requirement, especially because ground truth depth is hard to retrieve. So the research has so far focused on training only with stereo images. Xie et al. [24] trained their Deep3D model using stereo images to do image interpolation. Their approach is trained end-to-end directly on stereo pairs extracted from 3D movies and consist in a image reconstruction of the right

view from the left one. Garg et al. [5] adopted the same image reconstruction approach and generate an inverse warp of the target image using the predicted depth and known inter-view displacement, to reconstruct the source image. However these two methods make strong approximations which lead to poor reconstruction quality. Kuznietsov et al. [11] kept the image reconstruction method using warping to reproduce the right image and enforcing the depth map to be consistent to the ground truth depth with a direct image alignment loss.

Godard et al. [7] designed a model that learns to predict depth from both images of a stereo pair by processing reference image only, enabling a left-right consistency check to improve the quality of the reconstructed image. The key of their method is that they can simultaneously infer both disparities (left-to-right and right-to-left), using only the left input image and obtain depths by enforcing them to be consistent with each other. The predictions of Godard et al. appear more detailed than Kuznietsov's which seem to be smoother. Luo et al. [15] outperformed all the other methods by showing that the monocular depth estimation problem can be reformulated as two sub-problems: a view synthesis procedure, followed by stereo matching imposing geometrical constraints explicitly during inference. However, this method is way to complex and power greedy and it turned out to unfit our purposes.

The aim of this thesis is to enhance the disparity prediction by giving sparse depth output of a SLAM system as input to the already well-performing system of Godard et al. Therefore, by following the example of Ma et al. [16] whose model learns directly from the RGB-D raw data using a single deep regression network, it's possibile to get more precise depth maps. They employ however a supervised method, which is not the purpose of this thesis.

Traditional convolutional networks perform poorly when applied to sparse data even when the location of missing data is provided to the network, for example using a mask. Exploiting sparse convolutions [23], which explicitly considers the location of missing data during the convolution operation, it was possibile to greatly increase the precision of the depth maps.

The best choice would be to use a Monocular SLAM algorithm, but since

there are no open source systems with fairly correct depth scale, only a stereo SLAM system can be used. Therefore, depth points have been obtained from ORB-SLAM2 algorithm by Raúl Mur-Artal and Juan D. Tardós [18] employing stereo mode.

Besides using the disparity domain instead of the depth one, this work exploits the sparse disparity in input as a basis for prediction, helping the network with a glimpse of the scene structure.

Chapter 3

Monodepth

Godard et al. [7] designed a model that learns to predict depth from a single image. Their goal is to learn a function f that can predict the depth d for every pixel in the image, $d = f(I)$. Most of the existing learning based old approaches treat this as supervised learning problem, where the input is a color image and the output is compared to a target depth (Figure 3.1). This forces the network to predict globally coherent predictions. However, capturing reliable depth data is both expensive and time consuming. For example, the ground truth depth maps from lidar sensors provided in KITTI have unreliable data for moving and transparent objects. Therefore, unsupervised learning becomes an essential solution to infer feasible depth maps.

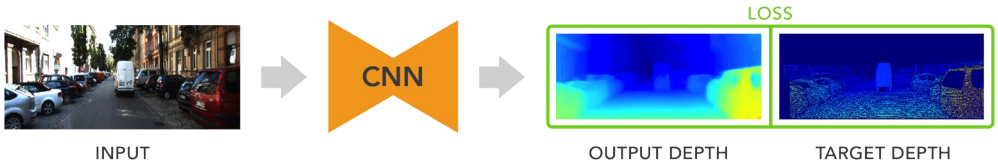

Figure 3.1: Supervised approach to depth prediction task.

3.1 Approach

The key intuition is to pose depth estimation as an image reconstruction problem at training time. Given a calibrated stereo camera, once we learn a function that is able to reconstruct the left image from the right or viceversa,

then we have learned a glimpse of the 3D shape of the scene the camera is pointing to. During training, the network needs I_l and I_r, corresponding to the left and right color images from a calibrated stereo pair, captured at the same moment in time and with a known baseline. The network will predict a left-to-right disparity map that, exploiting bilinear sampling from the left image, can be used to artificially reconstruct a right image $\tilde{I}_r$. Disparity is a scalar value per pixel that, given the baseline distance b between cameras and the focal length f, allows to recover the depth D from it: $Depth = \frac{baseline \times focal\ length}{disparity}$.

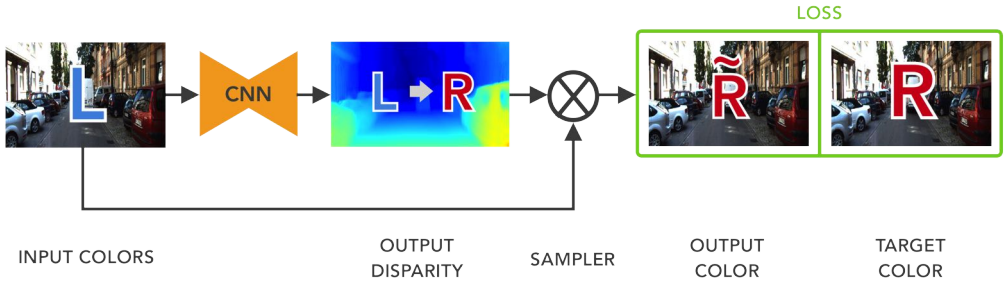

Figure 3.2: Naive sampling where the disparity map produced is aligned with the target.

As illustrated in Fig. 3.2, naively learning to generate the right image by sampling from the left one will produce disparities aligned with the right image (target). However, we want the output disparity map to align with the input left image, meaning the network has to sample from the right image. We could instead train the network to generate the left view by sampling from the right image, thus creating a left view aligned disparity map (No LR in 3.3).

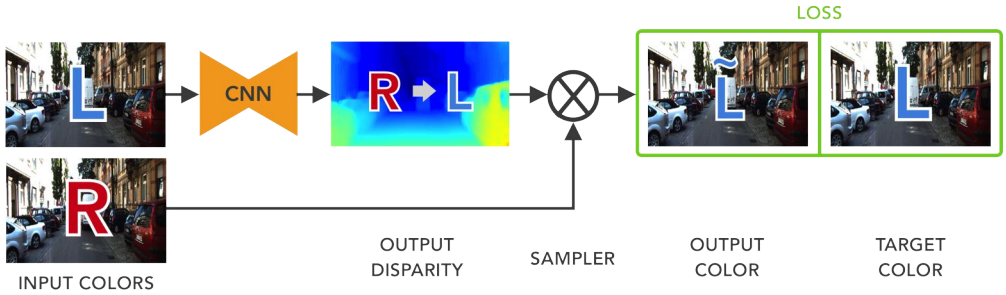

Figure 3.3: The no-LR model which corrects the alignment between input and target, but suffers from artifacts.

This approach works fairly well, but yields to 'texture-copy' artifacts and errors at depth discontinuities. Therefore Godard at al. solved this by training the network to infer the disparity maps for both views like in figure 3.4.

3.2 Network Structure

The network requires only the left image as input and exploits the right one for the losses during training. The novel approach consists in enforcing consistency between both disparity maps, leading to more accurate results.

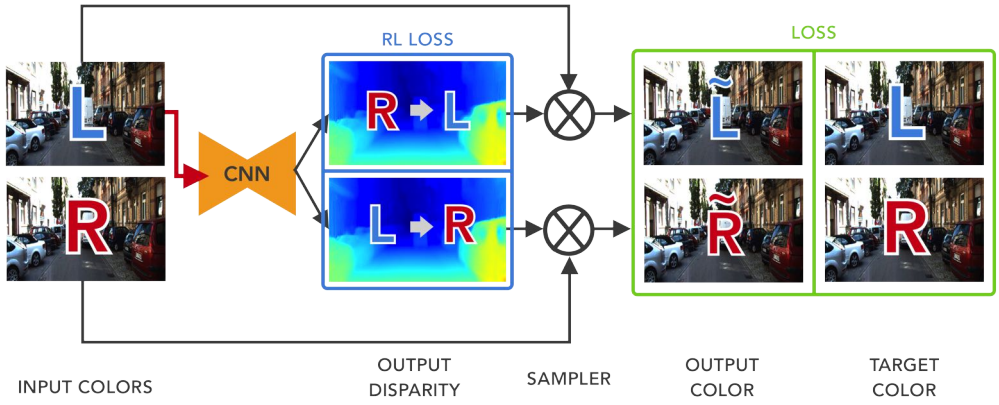

Figure 3.4: The complete approach which uses the left image to produce disparities for both images, improving quality and enforcing consistency.

They have used VggNet [21] and a variant of ResNet50 [9] for training, thus a convolutional neural network with encoder and decoder. The latter uses skip connections from the former's activations block in order to resolve higher resolution details. The network outputs two disparity maps (left-to-right, right-to-left as in figure 3.4) at four different scales as shown in figure 3.5.

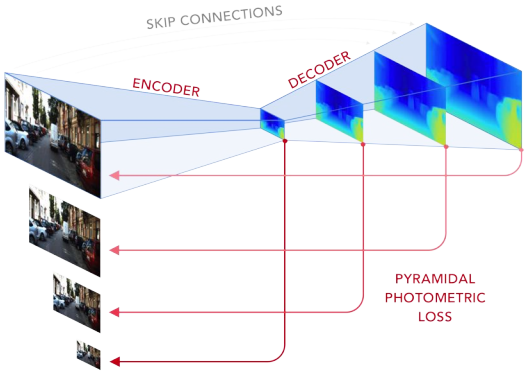

Figure 3.5: Example of the four different scale predictions. Each disparity map is employed in the photometric loss.

Each scale is used by the photometric loss, which is composed by L1 loss and SSIM loss between the RGB image and the disparity.

3.3 Losses

Monodepth network is forced to infer well defined disparity maps according to three main losses. Each scale s has a loss C_s, thus the total loss is the sum $C = \sum_{s=1}^{4} C_s$ where C_S is defined as:

$$C_s = \alpha_{ap}(C_{ap}^L + C_{ap}^R) + \alpha_{ds}(C_{ds}^L + C_{ds}^R) + \alpha_{lr}(C_{lr}^L + C_{lr}^R) \tag{3.1}$$

Each of the three terms is composed by the left and the right component, but only the left image is fed to the CNN.

Appearance Matching Loss

This loss enforces the reconstructed image to appear similar to the corresponding training input. It's a sort of photometric loss composed by a combination of L1 and single scale SSIM which compares the input image I_{ij}^L and its reconstruction $\widetilde{I}_{ij}^L$ obtained by bilinear sampling.

$$C_{ap}^L = \frac{1}{N}\sum_{ij} \alpha \frac{1 - SSIM(I_{ij}^L, \widetilde{I}_{ij}^L)}{2} + (1-\alpha)||I_{ij}^L - \widetilde{I}_{ij}^L|| \tag{3.2}$$

where N is the number of pixels and the SSIM is simplified by exploiting a 3x3 filter instead of a Gaussian and $\alpha = 0.85$.

Disparity Smoothness Loss

This loss enforces smooth disparities exploiting an L1 penalty on the disparity gradients ∂d. To preserve edges on disparity maps, they weight this cost with an edge aware term using the image gradients ∂I.

$$C_{ds}^L = \frac{1}{N}\sum_{ij} |\partial_x d_{ij}^L| e^{-||\partial_x I_{ij}^L||} + \partial_y d_{ij}^L| e^{-||\partial_y I_{ij}^L||} \tag{3.3}$$

Left-Right Consistency Loss

This loss enforces the left and the right disparities to be consistent. They force consistency by exploiting an L1 penalty between the left-to-right disparity map and the reconstructed one which comes from sampling the right-to-left one the same way they do with left and right images:

$$C_{lr}^{L} = \frac{1}{N} \sum_{ij} |d_{ij}^{L} - d_{ij+d_{ij}^{L}}^{R}| \tag{3.4}$$

3.4 Implementation Details

Godard's implementation has been modified to suit the thesis purposes. Although the model is basically the same, training and test times are different since this study has been conducted with a Tesla K80 GPU instead of Titan X GPU. As detailed below, this times with a VggNet are much slower on a dataset of 30K images by 50 epochs with 256×512 maps.

	training time	FPS
Titan X	25h	28
Tesla K80	46h	8

In this study, all the monodepth settings have been kept untouched:

- 50 epochs
- 8 as batch size
- Exponential Linear Units [2]
- Adam Optimizer [10] with $\beta_1 = 0.9, \beta_2 = 0.999$ and $\epsilon = 10^{-8}$
- learning rate of $\lambda = 10^{-4}$ for the first 30 epochs, then it's been halved every 10 epochs.
- Data augmentation: horizontal flip of the input and color augmentation, both with with a 50% chance; random gamma, brightness and color shifts.

- Loss weights: $\alpha_{ap} = 1$, $\alpha_{lr} = 1$ and $\alpha_{ap} = 0.1/r$ where r is the downscaling factor of the corresponding layer with respect to the resolution of the input image that is passed into the network.

Godard et al. constrain the output of the network to be between 0 and $d_{max} = 0.3$. A study was conducted to find the best temperature parameter, but this value is been kept untouched for fairness on evaluation. Read Chapter 6 for more detailed description on this study.

3.5 Post-Processing

To obtain nice looking outputs, Godard et al. designed an algorithm to post-process the maps which employs the two disparity maps that the network outputs and a per-pixel weight map w^L for d^L as:

$$w^L(i,j) = \begin{cases} 1 & if j \leq 0.1 \\ 0.5 & if j > 0.2 \\ 5 * (0.2 - i) + 0.5 & else \end{cases} \tag{3.5}$$

where i, j are normalized pixel coordinates, and the weight map w'^L for d''^L is obtained by horizontally flipping w'^L. They calculate the final disparity as:

$$d = d^L w^L + d''^L w'^L \tag{3.6}$$

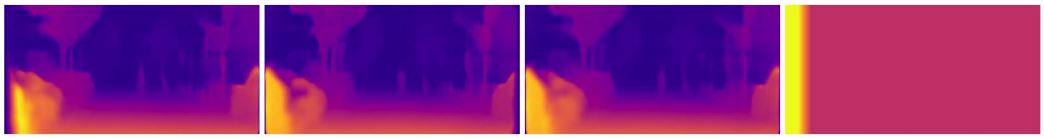

Figure 3.6: From monodepth [7]. Example of a post-processed disparity map. From left to right: The disparities d^L, d''^L, d and the weight map w'^L.

Chapter 4

Experimentation

The main idea of this work is to enhance Godard's model with data from SLAM algorithm. A way to do it is to get the depth from the keypoints and convert them into disparity to match the output of the network. This chapter presents all the experimentations on sparse inputs and it shows that forcing the network to use those is not easy neither banal. From now on, we will refer to monodepth model as *RGB model* which employs only RGB images to infer disparity maps like in Figure 4.1

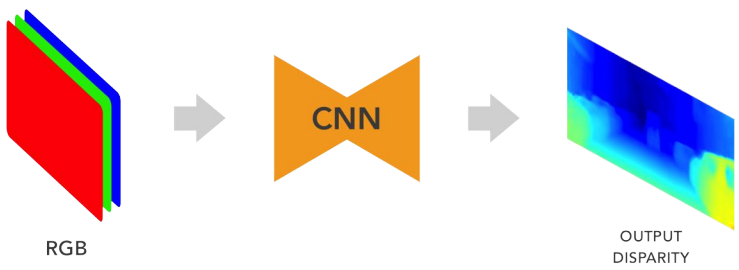

Figure 4.1: The RGB model, original monodepth model in which only RGB images are fed to the network at test time.

4.1 Employing ground truth

The first idea to start experimenting with sparse input was to employ ground truth depth from Lidar. KITTI [6] dataset provides either Lidar values as

BIN files and as PNG images. The former contains all the 360 degrees Lidar measurements, while the latter is a projection of Lidar values onto the focal plane. It's basically a sparse map in which each value is a depth measurement from Lidar like they come from the camera.

Since a SLAM algorithm can get hundreds of 3D landmarks on features, 200 points are sampled from the ground truth depth maps employing the Harris Corner Detector [8]. This algorithm detects corners on RGB images in a similar fashion as SLAM does when it searches for features. The sampling pipeline is shown in Figure 4.2 and it starts from RGB and ground truth depth.

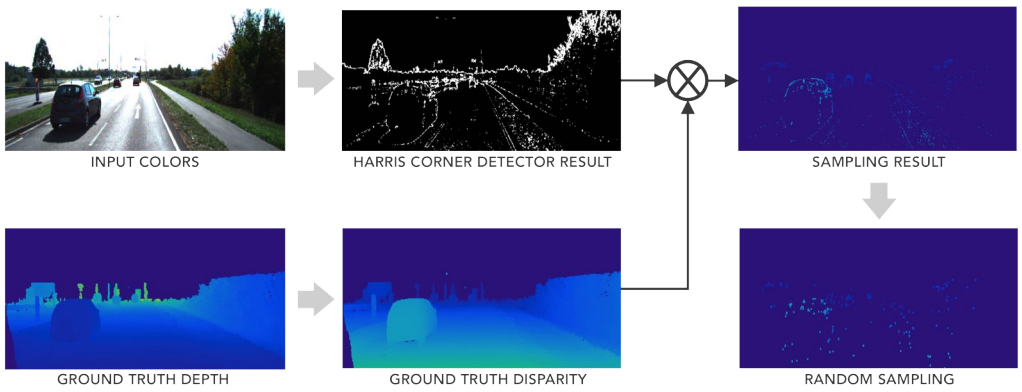

Figure 4.2: Sampling pipeline based on Harris Corner Detector.

Harris Corner Detector yields to a binary map with corners, which intersected with ground truth converted in disparity, leads to a sparser ground truth disparity map with values only on corners. Subsequently, 200 points are sampled in a random manner from that map, yielding to a potentially good input for the network. Using ground truth for this purpose is like finding an *upper bound* of the results which this model could ever reach with actual points from SLAM.

Evaluating novel models

To get an idea of how good a new model is, we need to define a metric of evaluation. Let's define the metric a_i as the percentage of predicted pixels

where the absolute relative error is within a threshold. Specifically:

$$a_k = \frac{\mathbf{card}(\{Gt_i : \max\{\frac{Pr_i}{Gt_i}, \frac{Gt_i}{Pr_i}\} < 1.25^k\})}{\mathbf{card}(\{Gt_i\})} \tag{4.1}$$

where Gt_i and Pr_i are respectively the i-th correspondent ground truth and the prediction points, and **card** is the cardinality of a set. Since it's a percentage of good pixels, a higher a_k indicates better prediction. Considering this metric, the models below were evaluated and then the last one has been selected for all the other evaluation.

4.2 RGBD and RGBDM models

The key intuition is to concatenate the sparse disparity map from SLAM to the RGB images as a fourth channel, yielding to a RGBD model like in Figure 4.3.

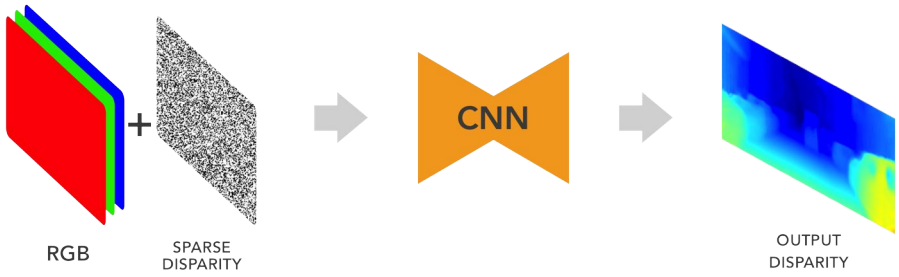

Figure 4.3: The RGBD model which employs the sparse disparity map as fourth channel.

The only way to enforce the network to mesh this values with the output, is to add a loss between this input and the output. This map however, needs to be divided by the image width to be comparable to the output, otherwise we force a wrong value into the output yielding to bad results. Inspired by [16] we use an outer L1 loss between the input sparse disparity map I_{sd} and the output dense disparity map O_{dd} only in those pixels where I_{sd} is greater then zero, thus where there's an actual value:

$$C_{L1} = \frac{1}{N} \sum_{I_{sd}(i,j)>0} |I_{sd}(i,j) - O_{dd}(i,j)| \tag{4.2}$$

where N is the number of the actual values (e.g. 200).

This loss needed to be weighted by a α_{L1} term to see actual results, so 3.1 becomes:

$$C_s = \alpha_{ap}(C_{ap}^L + C_{ap}^R) + \alpha_{ds}(C_{ds}^L + C_{ds}^R) + \alpha_{lr}(C_{lr}^L + C_{lr}^R) + \alpha_{L1}(C_{L1}^L) \quad (4.3)$$

After some tests, the best weight was empirically found at 30. Employing a higher weight, the maps would show some artifacts on sparse points location.

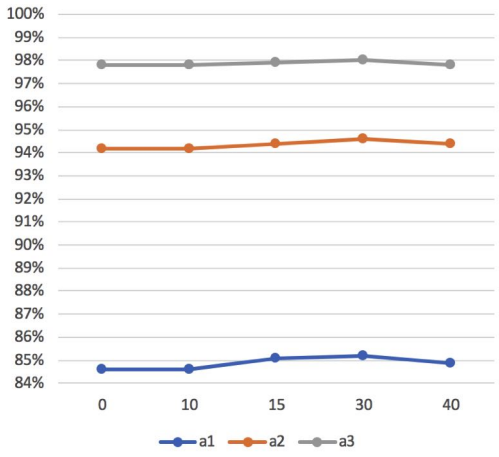

Figure 4.4: Plot of thresholded asbolute error by loss weights.

Model	α_{L1}	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
RGB	0	0.846	0.942	0.978
RGBD	10	0.846	0.942	0.978
RGBD	15	0.851	0.944	0.979
RGBD	30	**0.852**	**0.946**	**0.980**
RGBD	40	0.849	0.944	0.978

Figure 4.5: RGB and RGBD models trained with VGG on KITTI. RGBD had 200 sparse disparity points in input. We can see that the RGBD model performs slightly better than RGB one when the loss weight is around 30, although forcing a higher weight, the results begin to drop.

It is, however, mandatory to use a lower weight, since ground truth points are employed: forcing a higher loss with SLAM points could lead instead to "bumps" on sparse points location. We want the points to enhance the global prediction preserving a good looking map, thus we set the weight loss at 10.

Unfortunately this approach led to very little changes in the results. Therefore we came up with the RGBDM model, like in Figure 4.6, to help the network somehow to use the sparse points in a better way.

This approach was even worse than the original RGB model, either with and without loss; therefore it has been discarded right away in favour of a novel model that employs sparse convolutions.

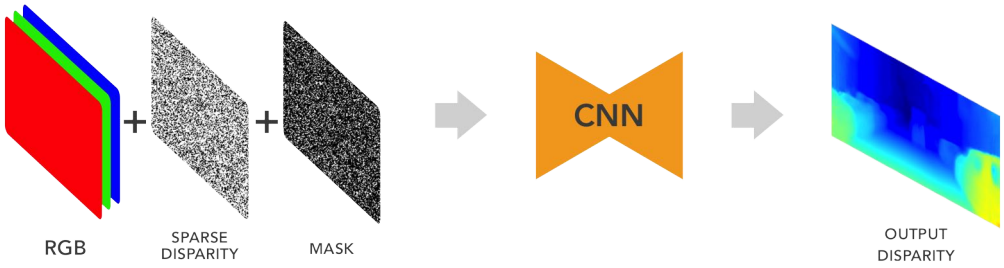

Figure 4.6: The RGBDM model which employs the sparse disparity map and a binary mask corresponding to the sparse points.

4.3 Sparse convolutions models

Regular convolution perform poorly when applied to sparse data. Sparse convolution [23] solve this problem, explicitly considering the location of missing data during the convolution operation. This kind of convolution generalizes well to novel datasets and is invariant to the level of sparsity in the data. Therefore, it has been tested in different models.

4.3.1 Sparse Convolutions

The output of a standard convolutional layer in a CNN is computed via:

$$f_{u,v}(x) = \sum_{i,j=-k}^{k} x_{u+i,v+j} w_{i,j} + b \tag{4.4}$$

where $2k+1$ is the kernel size, w are the weights and b the biases.

Sparse convolution layers instead consider sparsity by evaluating only observed pixels and normalizing the output according to a binary mask o that denotes corresponding binary variables which indicate if an input is observed ($o_{u,v} = 1$) or not ($o_{u,v} = 0$).

Thus the 4.5 becomes:

$$f_{u,v}(x, o) = \frac{\sum_{i,j=-k}^{k} o_{u+i,v+j} x_{u+i,v+j} w_{i,j}}{\sum_{i,j=-k}^{k} o_{u+i,v+j} + \epsilon} + b \tag{4.5}$$

A small ϵ is added to avoid division by zero where there's no valid input.

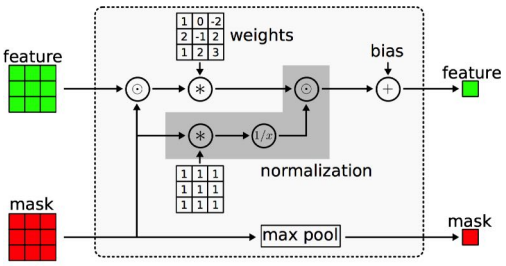

Figure 4.7: From [23]. Sparse convolution operation with binary masks. $\odot$ denotes elementwise multiplication, $*$ convolution, $1/x$ inversion and $max\ pool$ the max pooling operation. The input feature can be single-channel or multi-channel.

The layer works weighting the elements of the convolution kernel according to the validity of the input pixels. That validity is then carried to subsequent layers of the network keeping track of the visibility state and make it available to those. Subsequent observation masks $f^o_{u,v}(o)$ are defined via max pooling operation:

$$f^o_{u,v}(o) = \max_{i,j=-k,\dots,k} o_{u+i,v+j} \tag{4.6}$$

which evaluates to 1 if at least one variable is visible to the filter and 0 otherwise.

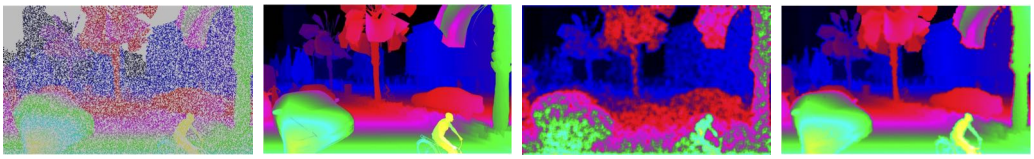

Figure 4.8: From [23]. Example of sparse convolution on Lidar input.

In Figure 4.8 we can see the comparison between regular convolution and sparse convolution. The first image from the left is the sparse input (visually enhanced). The second image is the ground truth. The third and the fourth are respectively the result of regular convolutions and the result of sparse convolutions. We can see that the latter perform much better and retain the shape of objects.

4.3.2 Double-Encoder model

The first attempt to implement sparse convolutions was based on creating a second encoder only for the sparse disparity map. Thus, the network became a double encoder system like in Figure 4.9.

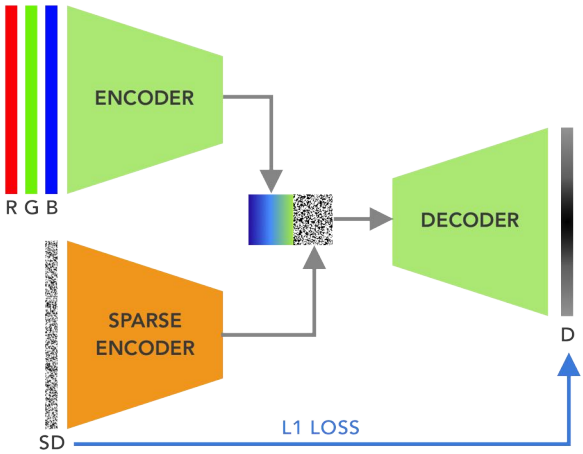

Figure 4.9: The double encoder model. The RGB encoder is a VggNet or ResNet one, while the SD encoder is the same, but with sparse convolutions instead of regular ones.

This model is composed by two encoders: RGB and SD. The RGB one performs regular convolutions on input image and the SD one performs sparse convolutions on the sparse disparity map. The two results are concatenated and fed to a single decoder that produce good dense disparity maps. This model works, but doesn't seem to substantially increase the performance and it's much slower to train than the RGB model.

4.3.3 Autoencoder models

To reach even better results, two autoencoder models have been created employing the structure presented in [23] and shown in Figure 4.10.

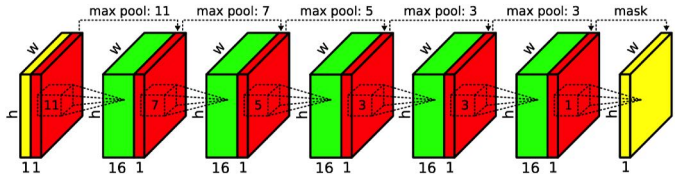

Figure 4.10: From [23].The autoencoder structure whose input is a sparse disparity map (yellow) and a binary observation mask (red).

Figure 4.11: A detail of denser disparity output from the autoencoder.

After several sparse convolution layers with decreasing kernel size (11×11 to 3×3), the result is a denser disparity map (Figure 4.11). This increment of size is due to sparse convolutions kernels. The output of the autoencoder is concatenated to the RGB image and fed to a RGBD model (Figure 4.3). This novel model is shown in Figure 4.12.

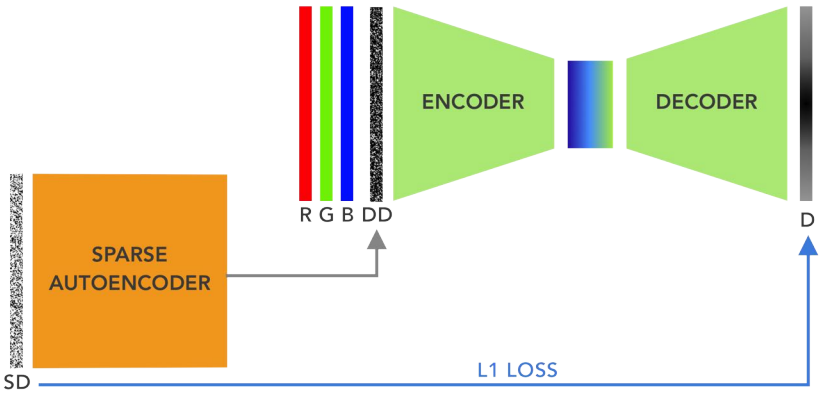

Figure 4.12: The autoencoder model with L1 loss between input and output.

This model led to a substantial improvement in metrics and outputs, but a loss has been added to force the autoencoder to preserve the input on denser disparity maps. This addition increased the percentage of good values on lower threshold in the absolute relative error, thus it was chosen as the best model so far that works with ground truth points. The model is shown in Figure 4.13.

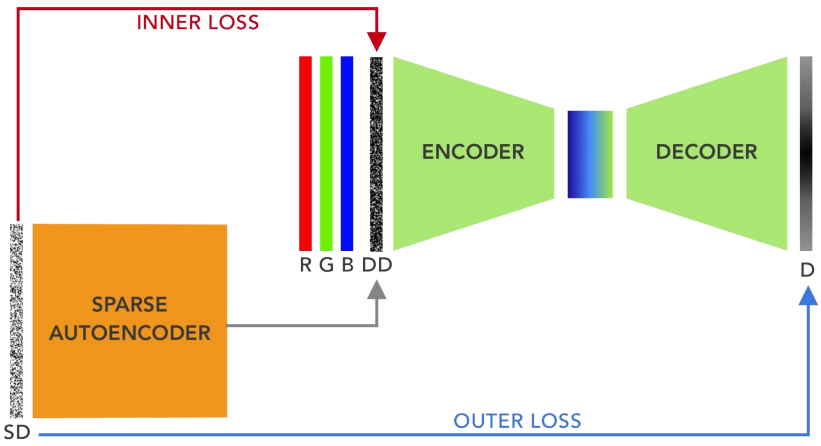

Figure 4.13: The autoencoder model with inner and outer loss.

The final loss has now two new terms C^L_{inner} and C^L_{outer}, thus 3.1 becomes:

$$C_s = \alpha_{ap}(C^L_{ap} + C^R_{ap}) + \alpha_{ds}(C^L_{ds} + C^R_{ds}) + \alpha_{lr}(C^L_{lr} + C^R_{lr}) + \alpha_{L1}(C^L_{inner} + C^L_{outer}) \tag{4.7}$$

4.3.4 Results

In Figure 4.14 we can see some results with RGB, RGBD, RGBDM and the double-loss autoencoder model, setting the same loss weight at 10 for all the models and with 200, 2000 and 20000 points in input.

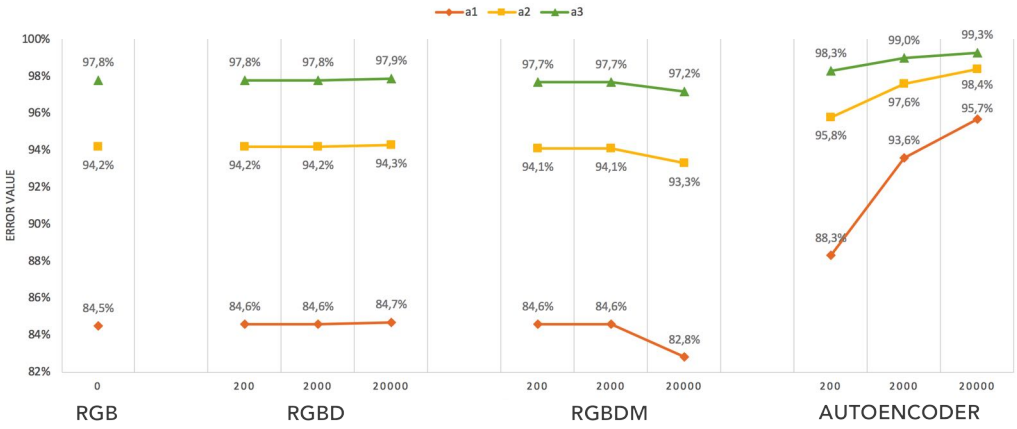

Figure 4.14: Absolute relative error with three different thresholds for RGB, RGBD, RGBDM and double-loss autoencoder models.

It shows that the RGBD model has no improvement, the RGBDM has ever worse results on 20000 points, while the autoencoder model has very high improvements proportional to the number of points in input.

In Table 4.1 we can see that the best model is the double-loss autoencoder with loss weight at 20. We however kept experimenting with the loss weight at 10, since forcing the output to be identical to the input will work only with ground truth. The best model to work with inaccurate input points from SLAM, will be the model with loss weight at 1 because it will infer smooth maps, while the one with loss weight at 10 will have better metrics, but slightly less smoothness in maps.

Model	α_{L1}	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Double-Encoder	10	0.844	0.942	0.978
Single-Loss Autoencoder	10	0.846	0.943	0.979
Double-Loss Autoencoder	0	0.856	0.947	0.980
Double-Loss Autoencoder	1	0.872	0.953	0.982
Double-Loss Autoencoder	10	0.888	0.959	0.984
Double-Loss Autoencoder	20	**0.891**	**0.961**	**0.985**

Table 4.1: Absolute relative error with threshold on double-encoder, single-loss autoencoder and double-loss autoencoder with different loss weights, all with VGG.

4.3.5 Comparison with Monodepth

In Figure 4.15 and 4.16 we show some qualitative results of the autoencoder model, compared to monodepth (RGB model).

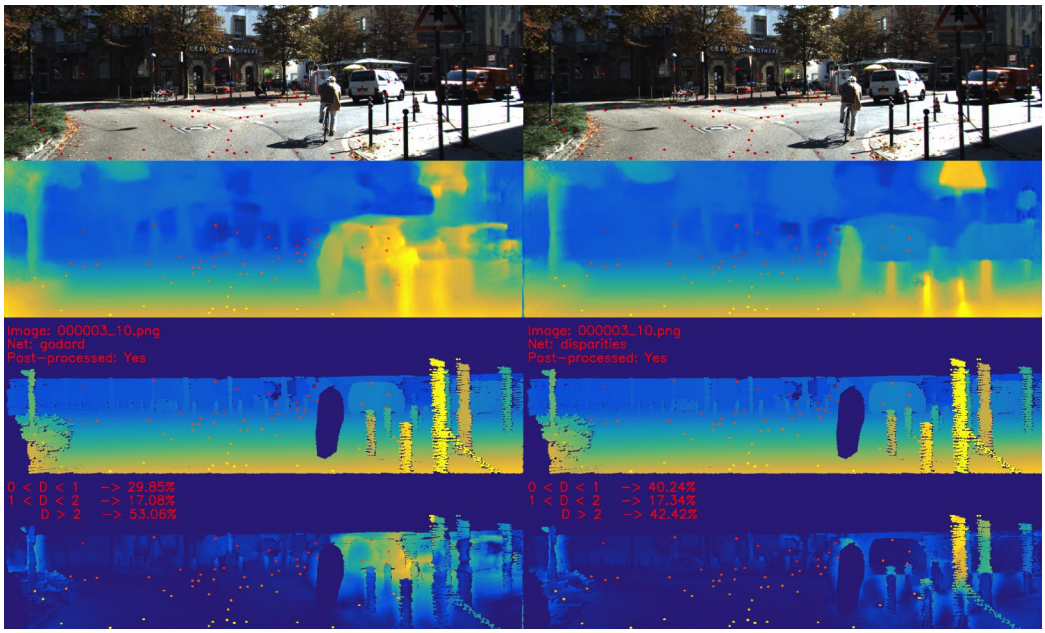

Figure 4.15: Comparison between RGB and Autoencoder model on the same scene taken form KITTI2015 test set.

The first row of each of the two examples shows the RGB image, while the second row shows monodepth prediction on the left and autoencoder prediction on the right. The third row shows the ground truth disparity. The fourth row shows the L1 difference between the related prediction and the

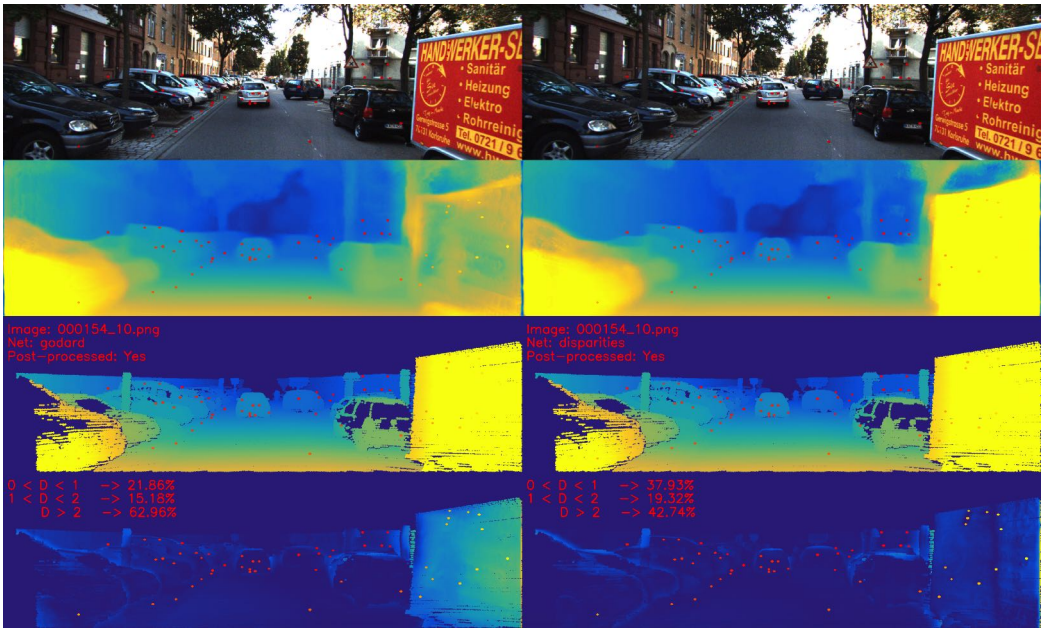

Figure 4.16: Another example of better prediction on a scene from the same test set.

ground truth. You can see that the two vans are almost perfectly detected compared to monodepth.

You can find some information about the tool used for evaluation in Appendix A at page 61 and more results on Appendix B at page 65.

Chapter 5

ORB-SLAM

5.1 SLAM

The term SLAM is an acronym for Simultaneous Localization And Mapping. It was originally developed by Hugh Durrant-Whyte and John J. Leonard [13] and then improved and enhanced by a lot of researchers. This algorithm allows to build a map of an unknown environment and localize the sensor (e.g. a camera) in the map. Cameras are cheap sensors, so Visual SLAM is the most researched topic nowadays.

Figure 5.1: From [17]: Camera frames (blue), Current Camera position (green), slam Keypoints (black, red) and Current Local Keypoints (red).

5.2 ORB-SLAM

Developed by Raul Mur-Artal, J. M. M. Montiel and Juan D. Tardós [17][18], it has been the best open source state-of-the-art SLAM algorithm for monocular, stereo and RGB-D cameras. The system works in real-time on standard CPUs in various environments. It uses ORB [19], which are oriented multi-scale corners with a 256 bits descriptor associated. They are extremely fast to compute and match and they also have good invariance to viewpoint.

This system consists in three main threads that run in parallel: tracking, local mapping and loop closing. The tracking is in charge of localizing

the camera with every frame, inserts new Keyframes and extracts features (MapPoints). The local mapping processes new keyframes and performs local Bundle Adjustment [22] to achieve optimal reconstruction in the surroundings of the camera pose. Finally, the loop closing searches for loops with every new keyframe and tries to achieve global consistency.

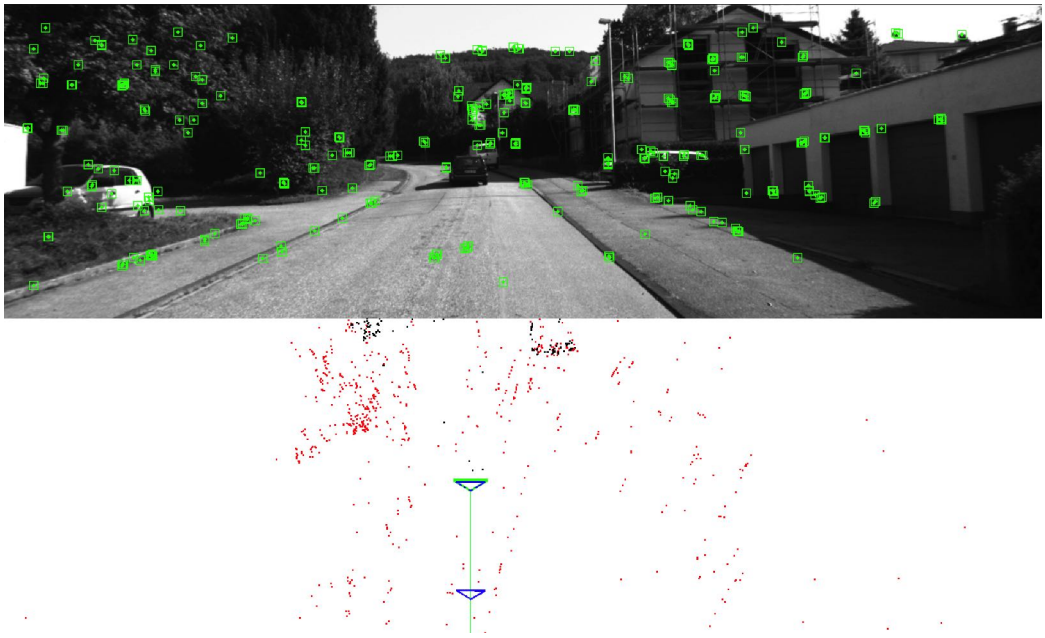

Figure 5.2: ORB-SLAM2 on Stereo Mode at Odometry Sequence 03 by KITTI. KeyFrames (blue), Current Camera (green), MapPoints (black, red) and Current Local MapPoints (red).

5.3 Obtain sparse disparity maps from ORB-SLAM

ORB-SLAM2 code is available online[1] and it has been used during this study to get the depth values from Keypoints. They refer to Keypoints as MapPoints p_i, which stores the 3D position $[X_i, Y_i, Z_i]^T$ in the world coordinate system and the position on camera plane $[u_i, v_i]^T$, along with some other information like viewing direction, ORB descriptor and so on. Each Keyframe

[1] https://github.com/raulmur/ORB_SLAM2

K_i stores the camera pose T_i, the camera intrinsics and all the ORB features extracted.

Therefore it was possible to extract all the depth values z_i from the position of the point on camera coordinate system $\widetilde{p_i} = [x_i, y_i, z_i]$, computing a reprojection world-to-camera of every MapPoint p_i employing the camera pose T_i:

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = T_i \times \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix} \tag{5.1}$$

Finally, the i-th point used to compose the sparse disparity map in input is:

$$SD_i = \begin{bmatrix} u_i \\ v_i \\ z_i \end{bmatrix} \tag{5.2}$$

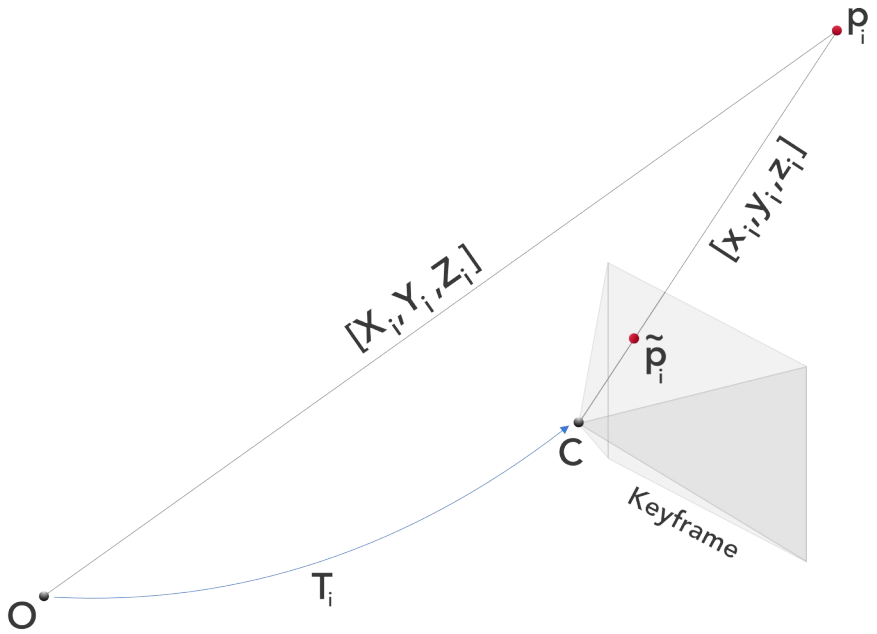

Figure 5.3: O is the center of the world coordinate system, T_i is the world-to-camera transformation and C is the center of the camera coordinate system. p_i is the the point in world coordinates, which are $[X_i, Y_i, Z_i]^T$ and $\widetilde{p_i}$ is the point at camera coordinates, which are $[x_i, y_i, z_i]^T$.

Chapter 6

Evaluation and Results

Evaluation, as well as training, has been done on two different datasets: KITTI [6] and Cityscapes [3]. The monodepth code[1] has been changed to allow training and testing on any dataset as long as a proper filenames file is provided. Evaluation is done comparing the predicted disparity map and the ground truth provided by the datasets.

6.1 Evaluation Metrics

Evaluation is based on 8 metrics for test sets with ground truth depth maps and one more metric is employed for the ones with ground truth disparity maps. Every metric is computed on post-processed disparity maps resized to the ground truth shape. Given Gt_i and Pr_i^R as the ground truth and resized-prediction correspondent points in the depth domain, $\widetilde{Gt}_i$ and $\widetilde{Pr}_i^R$ in the disparity one and N as the set of non-zero points in ground truth:

Absolute Error

$$\frac{1}{card(N)} \sum_{Gt_i \in N} |Gt_i - Pr_i^R| \tag{6.1}$$

[1] https://github.com/mrharicot/monodepth

Absolute Relative Error

$$\frac{1}{card(N)} \sum_{Gt_i \in N} \frac{|Gt_i - Pr_i^R|}{Gt_i} \tag{6.2}$$

Squared Relative Error

$$\frac{1}{card(N)} \sum_{Gt_i \in N} \frac{||Gt_i - Pr_i^R||^2}{Gt_i} \tag{6.3}$$

Root Mean Squared Error (linear)

$$\sqrt{\frac{1}{card(N)} \sum_{Gt_i \in N} ||Gt_i - Pr_i^R||^2} \tag{6.4}$$

Root Mean Squared Error (logarithmic)

$$\sqrt{\frac{1}{card(N)} \sum_{Gt_i \in N} ||\log Gt_i - \log Pr_i^R||^2} \tag{6.5}$$

D1-all

This metric comes from [6] and is computed only on ground truth disparity maps. It is the percentage of outliers averaged over all ground truth pixels.

$$100 \times \frac{\mathbf{card}(\widetilde{N})}{\mathbf{card}(N)} \tag{6.6}$$

where N is the set of non-zero ground truth points $\widetilde{Gt}_i > 0$ and $\widetilde{N}$ is a subset of N which corresponds to the following logical expression:

$$\sum_{\widetilde{Gt}_i \in N} |\widetilde{Gt}_i - \widetilde{Pr}_i^R| \geq 3 \cap \sum_{\widetilde{Gt}_i \in N} \frac{|\widetilde{Gt}_i - \widetilde{Pr}_i^R|}{\widetilde{Gt}_i} \geq 0.05 \tag{6.7}$$

which is the set of good pixels (i.e. $error < 3px$ and $error < 5\%$)

Threshold

$$\frac{\mathbf{card}(\{Gt_i : \max\{\frac{Pr_i^R}{Gt_i}, \frac{Gt_i}{Pr_i^R}\} < 1.25^k\})}{\mathbf{card}(\{Gt_i\})} \tag{6.8}$$

where k is the level of the threshold. We used k = 1,2,3.

6.2 Cityscapes

Cityscapes[2] [3] is a large-scale dataset that contains a diverse set of stereo video sequences recorded in street scenes from 50 different german cities. We used 22972 images to train on. This dataset brings higher resolution, image quality, and variety compared to KITTI, while having a similar setting. This allows the CNN to infer good disparity on vertical objects. Like [7], we cropped the input images to only keep the top 80% of the image, removing the very reflective car hoods from the input. Although they provide disparity maps for each image, those are not a good ground truth compared to the Lidar one, so they don't provide a reliable evaluation.

Figure 6.1: Cityscapes' german cities.

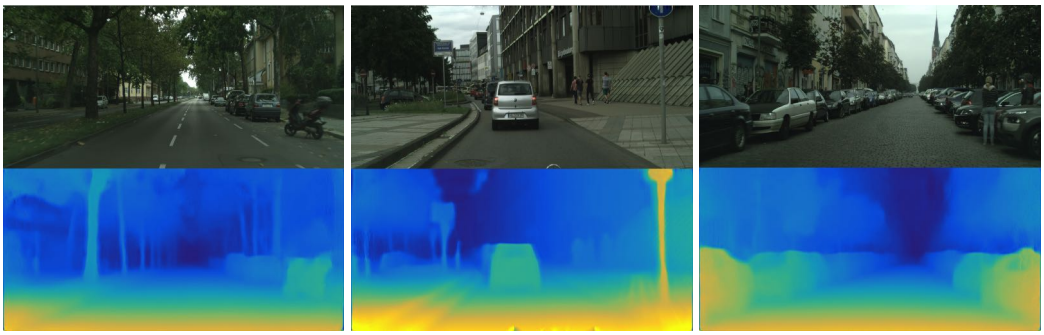

Figure 6.2: Three examples of disparity prediction on Cityscapes dataset.

		LOWER IS BETTER						HIGHER IS BETTER		
CITYSCAPES		Abs	Abs Rel	Sq Rel	RMSE	RMSE log	*D1-all*	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
VGG	Godard	10.6573	0.6760	9.1872	13.877	0.529	94.003	**0.065**	0.339	0.872
	Ours	**10.4691**	**0.6655**	**8.9863**	**13.706**	**0.523**	**93.961**	0.063	**0.366**	**0.887**
RES	Godard	10.7521	0.6853	9.4891	14.004	0.534	94.425	0.055	0.332	0.873
	Ours	**9.0226**	**0.5702**	**7.2518**	**12.205**	**0.472**	**90.739**	**0.140**	**0.546**	**0.904**

Table 6.1: Results on KITTI2015 test set with models trained on Cityscapes. This results are obviously bad, since the models are used to see images with a narrower baseline than KITTI's (0.22m rather than 0.54m), thus the scale is wrong. This table shows however that our models perform better than Godard's.

[2] https://www.cityscapes-dataset.com

6.3 KITTI

Figure 6.3: KITTI station wagon used for capturing datasets.

The KITTI[3] dataset [6] provides a reliable benchmark for autonomous driving task, such as depth prediction, stereo matching, optical flow, visual odometry and object detection. They equipped a standard station-wagon with two grayscale and two color cameras with a baseline of 0.54 meters. Accurated ground truth data is provided by a velodyne laser scanner and a GPS localization system with integrated inertial measurement unit and RTK corrections. The datasets are captured by driving in mid-sized cities, rural areas and highways. In its raw form, the dataset contains 42,382 rectified stereo pairs from 61 scenes, with a typical image being 1242×375 pixels in size. We evaluated models trained on KITTI and models pre-trained on Cityscapes and then fine-tuned on KITTI. All evaluations are done on post-processed disparity maps.

6.3.1 KITTI 2015

The KITTI 2015 split test is composed by 200 high quality disparity images provided as part of the official training set. This test set is specific for depth prediction and is much better than the reprojected velodyne laser depth values because of some post-processing to make them reliable. They also have CAD models inserted in place of moving cars. This means better depth values on car positions, but they result in ambiguous disparity values on transparent surfaces such as car windows, and issues at object boundaries where the CAD models do not perfectly align with the images. This test set is evaluated on all the metrics explained above, including the *D1-all*. It is important to note that measuring the error in depth space while the ground truth is given in disparities leads to precision issues. In particular, the non-thresholded measures can be sensitive to the large errors in depth caused by

[3] http://www.cvlibs.net/datasets/kitti/

prediction errors at small disparity values.

KITTI		LOWER IS BETTER						HIGHER IS BETTER		
		Abs	Abs Rel	Sq Rel	RMSE	RMSE log	*D1-all*	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
VGG	Godard	2.5852	0.1175	1.2385	5.828	0.208	30.478	0.845	0.942	0.978
	Ours	**2.0941**	**0.0912**	**0.9734**	**5.204**	**0.177**	**19.121**	**0.890**	**0.959**	**0.984**
RES	Godard	2.3980	0.1069	1.0794	5.537	0.194	26.626	0.861	0.951	0.981
	Ours	**1.8962**	**0.0806**	**0.8211**	**4.848**	**0.161**	**15.836**	**0.903**	**0.967**	**0.987**

Table 6.2: Evaluation on KITTI 2015 split with models trained on KITTI.

CITYSCAPES + KITTI		LOWER IS BETTER						HIGHER IS BETTER		
		Abs	Abs Rel	Sq Rel	RMSE	RMSE log	*D1-all*	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
VGG	Godard	2.1878	0.0995	0.9527	5.119	0.179	24.531	0.876	**0.960**	**0.986**
	Ours	**2.1570**	**0.0975**	**0.8915**	**5.094**	**0.177**	**24.469**	**0.879**	**0.960**	**0.986**
RES	Godard	2.1750	0.0982	0.9080	5.074	0.179	24.153	0.874	0.958	0.984
	Ours	**1.8355**	**0.0798**	**0.7862**	**4.689**	**0.156**	**15.794**	**0.910**	**0.970**	**0.988**

Table 6.3: Evaluation on KITTI 2015 split with models pre-trained on Cityscapes and fine-tuned on KITTI.

In Table 6.2 and Table 6.3 we can see evaluation on KITTI 2015 split on post-processed disparity maps. On ResNet the results differ from Godard's original paper [7], probably because of some changes on Tensorflow versions. Pre-training on the Cityscapes dataset improves the results over using KITTI alone. In Figure 6.4 we can see some qualitative results on KITTI 2015 split.

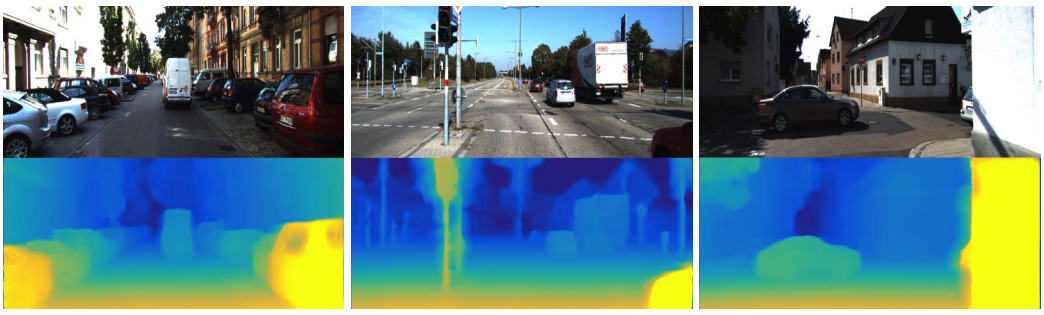

Figure 6.4: Three examples of disparity prediction on KITTI 2015 split test.

6.3.2 Eigen Split

Inspired by [4] we use the test split of 697 images which covers a total of 29 scenes. The remaining 32 scenes contain 23,488 images from which we kept 22,600 for training and the rest for evaluation as they did in [7]. To generate the ground truth depth images, Godard reprojected the 3D points viewed from the velodyne laser into the left input color camera. KITTI, however, provided already reprojected ground truth depth maps for all stereo images solving a lot of inaccuracies, but we kept Godard's reprojection for fairness sake. The provided ground truth depth maps are less then the actual number of images in the datasets. That's because they accumulate 11 frames to create those maps, then 5 frames are missing from the beginning and the end of each sequence. This has brought us to exclude those images that don't have the relative ground truth map, since our model needs to sample from ground truth to get the sparse disparity points. After this reduction, the Eigen split contains 652 images. To be consistent to Godard's evaluation, we used the same crop as [5].

KITTI		LOWER IS BETTER					HIGHER IS BETTER		
		Abs	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
VGG	Godard	2.3933	0.1121	0.9633	5.105	0.213	0.853	0.938	0.971
	Ours	**2.0428**	**0.0939**	**0.8182**	**4.737**	**0.193**	**0.882**	**0.951**	**0.977**
RES	Godard	2.2332	0.1030	0.8811	4.897	0.202	0.866	0.944	0.974
	Ours	**1.9421**	**0.0914**	**0.7388**	**4.525**	**0.188**	**0.890**	**0.954**	**0.979**

Table 6.4: Evaluation on Eigen split with models trained on KITTI.

CITYSCAPES + KITTI		LOWER IS BETTER					HIGHER IS BETTER		
		Abs	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
VGG	Godard	2.1189	0.1001	0.8125	4.686	0.198	0.874	0.947	**0.976**
	Ours	**2.1030**	**0.0999**	**0.7969**	**4.660**	**0.196**	**0.875**	**0.949**	**0.976**
RES	Godard	2.1051	0.1006	0.8186	4.645	0.195	0.876	0.949	0.976
	Ours	**1.8949**	**0.0885**	**0.7064**	**4.449**	**0.184**	**0.892**	**0.955**	**0.979**

Table 6.5: Evaluation on Eigen split with models pre-trained on Cityscapes and fine-tuned on KITTI.

In Table 6.4 and Table 6.5 we can see evaluation on Eigen split on post-processed disparity maps. Pre-training on the Cityscapes dataset improves the results over using KITTI alone only with ResNet. It seems like VggNet learns less than ResNet when pre-training on Cityscapes. In Figure 6.5 we can see some qualitative results on eigen split.

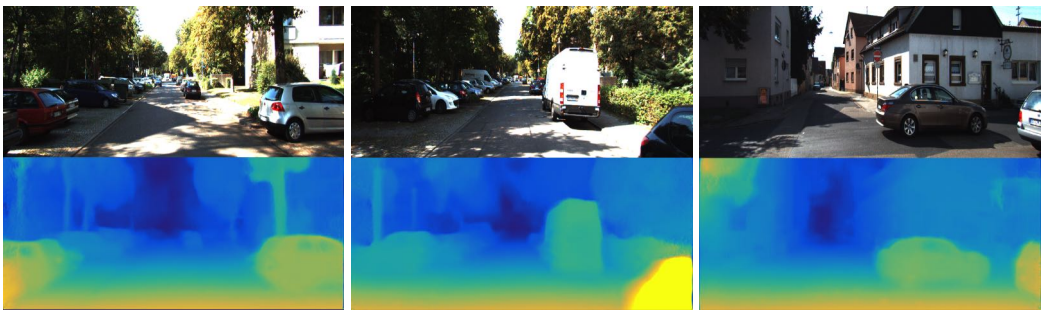

Figure 6.5: Three examples of disparity prediction on Eigen split test.

6.3.3 Odometry Sequence 03

This test-set is taken from one of the 22 sequences for odometry evaluation provided by KITTI[4]. Like every other sequence, it contains Lidar ground truth depth from velodyne laser scanner. Thus we decided to exploit it to evaluate on actual SLAM points which are obtained from ORB-SLAM as we discussed in 5.3.

In Figure 6.6 we can see some qualitative results on the sequence.

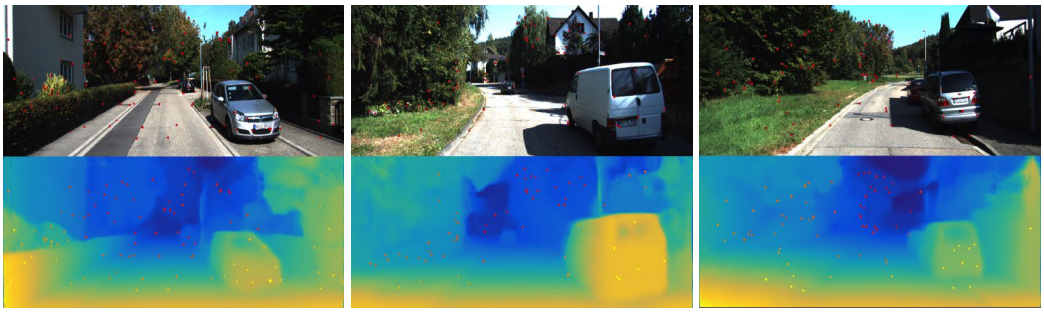

Figure 6.6: Three examples of disparity prediction on Odometry Sequence 03 test. You can see SLAM points with different color map overlapped to disparity maps.

[4] http://www.cvlibs.net/datasets/kitti/eval_odometry.php

KITTI		LOWER IS BETTER					HIGHER IS BETTER		
		Abs	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
VGG	Godard	2.8684	0.1461	1.4249	5.733	0.232	0.809	0.933	0.973
	Ours S	**2.3145**	**0.1098**	**1.0634**	**5.166**	**0.198**	**0.870**	**0.953**	**0.979**
	Ours G	*2.1825*	*0.1060*	*1.0172*	*4.958*	*0.190*	*0.876*	*0.956*	*0.981*
RES	Godard	2.7487	0.1380	1.3027	5.547	0.224	0.817	0.937	0.975
	Ours S	**2.1217**	**0.1008**	**1.0635**	**4.969**	**0.188**	**0.889**	**0.959**	**0.981**
	Ours G	*1.9955*	*0.0966*	*1.0138*	*4.750*	*0.179*	*0.894*	*0.963*	*0.984*

Table 6.6: Evaluation on Sequence 03 with models trained on KITTI. 'Ours S' is our model tested with points from SLAM and 'Ours G' is the same model but tested with points from ground truth.

CITYSCAPES + KITTI		LOWER IS BETTER					HIGHER IS BETTER		
		Abs	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
VGG	Godard	2.6265	0.1304	1.2346	5.432	0.220	0.832	0.939	0.975
	Ours S	**2.6039**	**0.1299**	**1.1750**	**5.329**	**0.215**	**0.836**	**0.942**	**0.977**
	Ours G	*2.5854*	*0.1292*	*1.1603*	*5.292*	*0.214*	*0.837*	*0.942*	*0.977*
RES	Godard	2.5839	0.1296	1.2024	5.304	0.215	0.836	0.944	0.977
	Ours S	**2.1034**	**0.1017**	**1.0479**	**4.883**	**0.187**	**0.888**	**0.960**	**0.981**
	Ours G	*2.0337*	*0.0983*	*1.0145*	*4.772*	*0.182*	*0.892*	*0.963*	*0.983*

Table 6.7: Evaluation on Sequence 03 with models pre-trained on Cityscapes and fine-tuned on KITTI. 'Ours S' is our model tested with points from slam and 'Ours G' is the same model but tested with points from ground truth.

It is easy to notice in Table 6.6 and in Table 6.7 that using pre-trained model on cityscapes doesn't increase the performance as much as with Godard's model. Anyway, our model still outperforms it both with points from SLAM and ground truth.

The Odometry Sequence 03 has been used for the Pangolin Demo, whose operation is explained in Appendix D.

6.4 Evaluation on Sparse Input

A further evaluation has been conducted on sparse input to understand how much these points could help the network to infer better disparity maps.

This test has been done on KITTI 2015 test set on two Double-Loss Autoencoder models trained with two different loss weight α_{L1}: 1 and 10. The evaluation consist on plotting the mean absolute error between the ground truth converted to disparity and resized to the input shape (256×512), $\widetilde{Gt}_i^R$ and five different values. For sake of simplicity, we will refer to the resized ground truth disparity $\widetilde{Gt}_i^R$ as D_i and to the sampled points from ground truth before the resize to input shape as SD_i.

Ground truth - prediction It is the mean absolute difference between all the non-zero points of the ground truth and the correspondent points of the predicted disparity map in the same positions.

$$\frac{1}{card(N)} \sum_{D_i>0} |D_i - Pr_i| \tag{6.9}$$

where N is the set of the ground truth points $D_i > 0$. The same test has been conducted with Godard's model predictions.

Ground truth - random sampled points It is the mean absolute difference between the points belonging to a 200 points subset of the ground truth and the correspondent points of the predicted disparity map in the same positions.

$$\frac{1}{card(M)} \sum_{\dot{D}_i>0} |\dot{D}_i - Pr_i| \tag{6.10}$$

where $\dot{D}_i$ is the i-th sampled points from the 200 points subset of the ground truth and M is the set of these points.

Ground truth - sampled points It is the mean absolute difference between the ground truth points in the same positions of the sparse disparity points given as input (thus, resized to input shape) and the points of the predicted disparity map in the same positions.

$$\frac{1}{card(N^R)} \sum_{SD_i^R>0} |D_i - Pr_i| \tag{6.11}$$

where N^R is the set of the input points resized to input shape, $SD_i^R > 0$.

Ground truth - window around sampled points It is the mean absolute difference between the mean of the ground truth values in a 11×11 window around the positions of the sparse disparity points given as input (thus, resized to input shape) and the mean of the predicted disparity values in a window of the same size and in the same positions.

$$\frac{1}{card(N^R)} \sum_{SD_i^R > 0} |W_{D_i} - W_{Pr_i}| \tag{6.12}$$

where N^R is the set of the input points resized to input shape, $SD_i^R > 0$ and W_{D_i} and W_{Pr_i} are respectively the mean of the values in the windows around D_i and Pr_i that are greater than zero.

In Figure 6.7, Equations 6.11 and 6.12 show that the values around the sparse disparity points in input are similar to ground truth, while Equations 6.9 and 6.10 show that the error gets slightly higher the further we get from the input points. This is due to the high loss weight α_{L1} that forces the network to overfit those values. We can see that the errors of our model are still lower than Godard's model (in black).

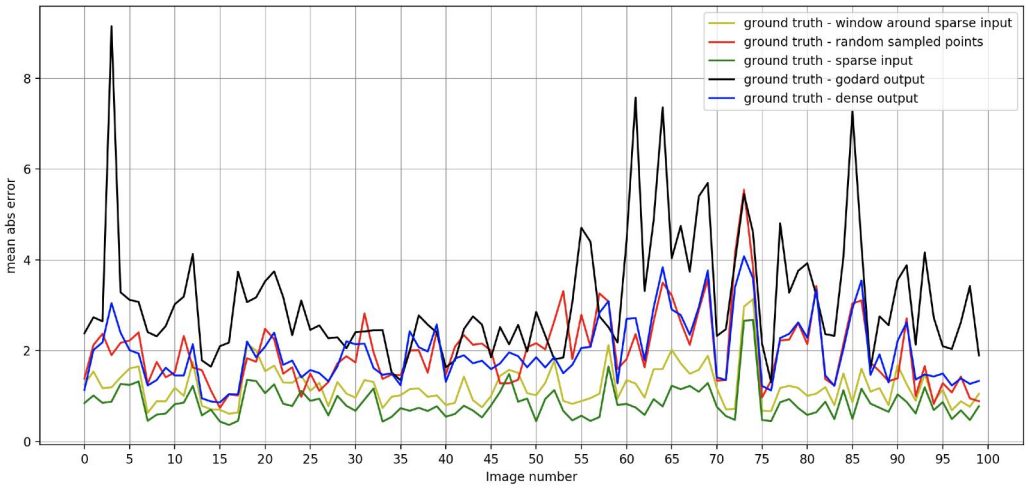

Figure 6.7: Sparse input evaluation on the first 100 images of KITTI 2015. This image represents the results on the Double-Loss Autoencoder model with loss weight α_{L1} set at 10.

Using the loss weight at 10 yields to good looking disparity maps and high results in metrics, but to be sure that of the good result with inaccurate

input points from SLAM, we should use a lower loss weight. The same test has been conducted to a model with loss weight at 1, shown in Figure 6.8.

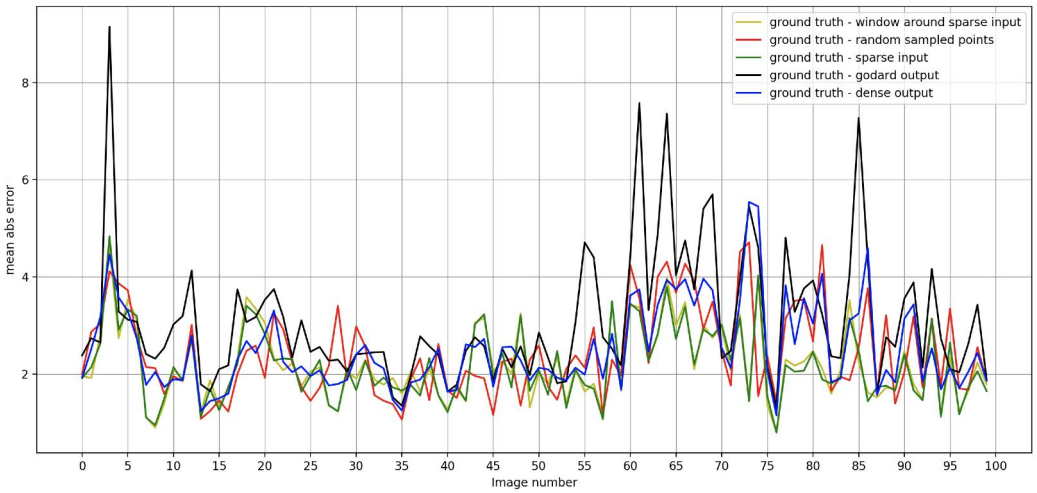

Figure 6.8: Sparse input evaluation on the first 100 images of KITTI 2015. This image represents the results on the Double-Loss Autoencoder model with loss weight α_{L1} set at 10.

It is easy to notice that there's no overfit on input points, but the metrics are slightly worse, as shown in Table 4.1. In any case, our model (blue line) performs better than Godard's (black line).

Chapter 7

Conclusions and further developments

This thesis had the aim to enhance the depth maps predictions of the already good work by Godard et al. [7] by exploiting a SLAM algorithm. The first task was to improve the results by employing ground truth depth and therefore several novel models have been tested. The best one is the Double-Loss Autoencoder which uses sparse convolutions upstream, along with a regular VggNet or ResNet downstream and it outperforms Godard's model.

Since there isn't an open source monocular SLAM with correct scale to employ, Stereo ORB-SLAM algorithm has been exploited to retrieve sparse disparity maps to input to our novel model. Several tests have been conducted to find the best loss weights and also to find the best balance between all the losses in order to obtain good looking disparity maps and high results in metrics. All the evaluation show that our model improves the predictions even with inaccurate points from SLAM.

Therefore, this thesis showed that employing sparse disparity in input could enhance the predictions of dense maps. We proposed a novel model that exploits points from a Stereo SLAM algorithm at their limits, but it needs input with correct depth scale. Therefore the further developments will cover a monocular SLAM with scale correction by IMU and GPS data, to allow a monocular system to be autonomous.

The next step would be to explore the possibility of employing monocular SLAM algorithm which has arbitrary scale for each sequence. The novel approach would be train the network using the correct scale only at training time to calibrate the arbitrary one from monocular. The network would be able to scale a monocular SLAM output without any other external help.

Bibliography

[1] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 61

[2] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *CoRR*, abs/1511.07289, 2015. 25

[3] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. *CoRR*, abs/1604.01685, 2016. 43, 45

[4] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2366–2374, 2014. 48

[5] Ravi Garg, Vijay Kumar B. G, and Ian D. Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. *CoRR*, abs/1603.04992, 2016. 18, 48

[6] Andreas Geiger. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 3354–3361, Washington, DC, USA, 2012. IEEE Computer Society. 27, 43, 44, 46

[7] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. *CoRR*, abs/1609.03677, 2016. 16, 18, 21, 26, 45, 47, 48, 55, 69

[8] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, 1988. 28

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 23

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 25

[11] Yevhen Kuznietsov, Jörg Stückler, and Bastian Leibe. Semi-supervised deep learning for monocular depth map prediction. *CoRR*, abs/1702.02706, 2017. 18

[12] Lubor Ladicky, Jianbo Shi, and Marc Pollefeys. Pulling things out of perspective. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pages 89–96, 2014. 17

[13] J. J. Leonard and H. F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, June 1991. 39

[14] Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian D. Reid. Learning depth from single monocular images using deep convolutional neural fields. *CoRR*, abs/1502.07411, 2015. 17

[15] Yue Luo, Jimmy S. J. Ren, Mude Lin, Jiahao Pang, Wenxiu Sun, Hongsheng Li, and Liang Lin. Single view stereo matching. *CoRR*, abs/1803.02612, 2018. 18

[16] Fangchang Ma and Sertac Karaman. Sparse-to-dense: Depth prediction from sparse depth samples and a single image. *CoRR*, abs/1709.07492, 2017. 18, 29

[17] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. ORB-SLAM: a versatile and accurate monocular SLAM system. *CoRR*, abs/1502.00956, 2015. 39

[18] Raul Mur-Artal and Juan D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *CoRR*, abs/1610.06475, 2016. 19, 39

[19] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, Nov 2011. 39

[20] Ashutosh Saxena, Min Sun, and Andrew Y. Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(5):824–840, May 2009. 17

[21] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 23

[22] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, ICCV '99, pages 298–372, London, UK, UK, 2000. Springer-Verlag. 40

[23] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant cnns. *CoRR*, abs/1708.06500, 2017. 18, 31, 32, 33

[24] Junyuan Xie, Ross B. Girshick, and Ali Farhadi. Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks. *CoRR*, abs/1604.03650, 2016. 17

Appendix A

Evaluation Tools

We created two tools for evaluation and for testing the sampling from ground truth. Both were created with OpenCV [1] and they allow to interact with them through the keyboard to modify some configurations or change target image.

A.1 Sparse Test

A specific tool has been created to conduct tests on finding the best configuration for the Harris Corner Detector based sampling. In Figure A.1 we can see the tool sampling disparity points from ground truth which is part of KITTI 2015 test set. It needs only the filenames file with all the paths to RGB and ground truth and it computes the sampling as explained in Chapter 4.1. The tool consists of a view with six cells which are, from top to bottom and from left to right: RGB image, result of Harris Corner Detector, ground truth depth, intersection of corners and ground truth disparity, ground truth disparity and result of random sampling from the intersection.

With keys A and D the user can change the target image and with the keyboard numbers listed below it's possible to change configuration selecting from some fixed values:

1. **BlockSize**: neighborhood size. [2, 4, 8].
2. **ksize**: aperture parameter for the Sobel Kernel [3, 5, 7, 9, 11].

3. **k**: Harris detector free parameter. See the formula below. [10^{-3}, 10^{-4}, 10^{-5}]

4. **threshold**: threshold of the local maxima for corners. [10^{-3}, 10^{-4}, 10^{-5}]

5. **Number** of the sampled points

6. Select either **disparity** or **depth** to sample from.

7. **Size** of both image and sampling when the detector and the sampling are computed.

We selected $[2, 3, 10^{-5}, 10^{-5}]$ for the Harris Corner detector configuration during training.

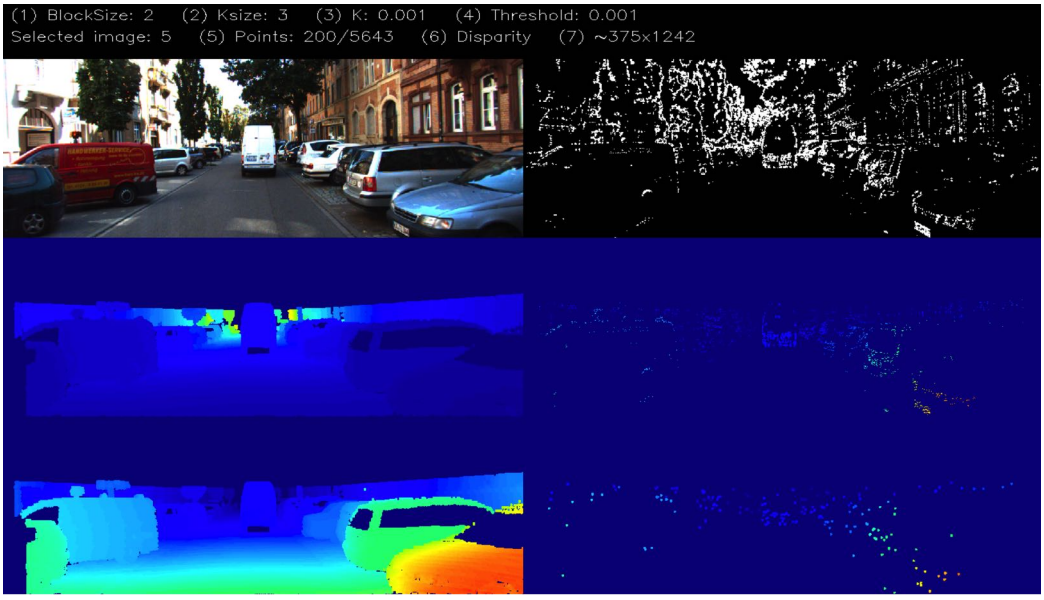

Figure A.1: Example of Sparse Test Tool

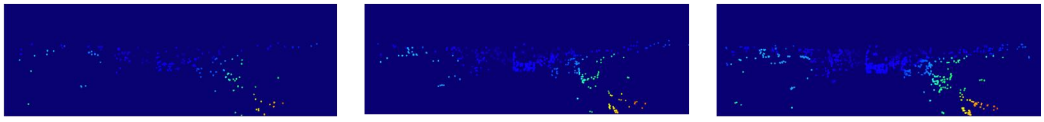

Figure A.2: Three different number of samples: 200, 500, 1000.

A.2 Show Disparities

We created a tool to visualize the results of the network's test mode: a NPY file with all the disparity maps computed using the RGB images of the test set provided to the net. After loading the file, each map needs to be 'resized' multiplying it by the original width of the RGB image. Subsequently, the ground truth is loaded from the filenames file provided along with the related RGB images, then it is dilated with a 5×5 structural element. The tool requires the NPY files of: Godard's results either normal and post-processed, as well as the NPY files of a different model's results. The latter would have NPY files for normal disparities, post-processed disparities and sparse disparity input maps. The tool view is composed of two columns. The left column shows Godard's result, while the right one shows our model's results. Each column is composed by RGB image, prediction, ground truth disparity, normalized L1 difference between the last two. With keys A and D the user can change the target image and with W and S it's possible to change the NPY file of some other model predictions.

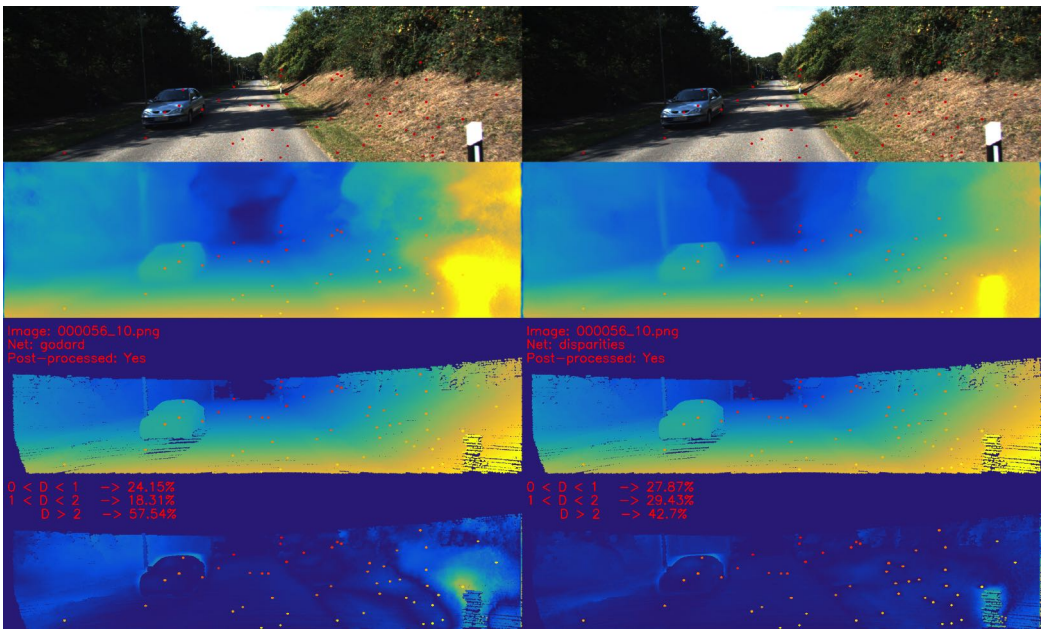

Figure A.3

In Figure A.3 you can see on ground truth some red labels which contain

the image number, the predicted disparity maps selected (Godard on the left and ours on the right) and the type of maps that are showed: regular or post-processed. The red labels on L1 difference image, show the percentage of correct predicted pixel D_i within three ranges: $0 < D_i < 1$, $1 < D_i < 2$ and $D_i > 2$.

In all the eight images, the sparse disparity points that the network gets as input are overlapped and colored by a colormap with inverted colors of the predicted disparity maps.

With the P key, the user can activate and deactivate the post-processing on the showed predicted disparity map. The difference can be seen in Figure A.4

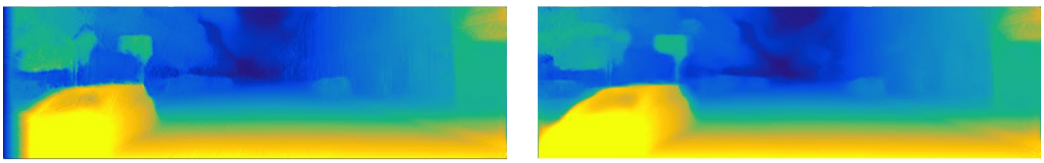

Figure A.4

With the I key, the user can hide and show the red labels and the sparse disparity points.

The user can visually enhance the disparity maps increasing a multiplication parameter with the M and lowering it with N. The effect can be seen in Figure A.5

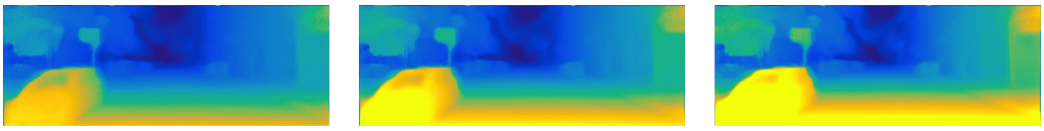

Figure A.5

Appendix B

More qualitative results

In Figures B.1 - B.5 we show some additional qualitative results from KITTI 2015.

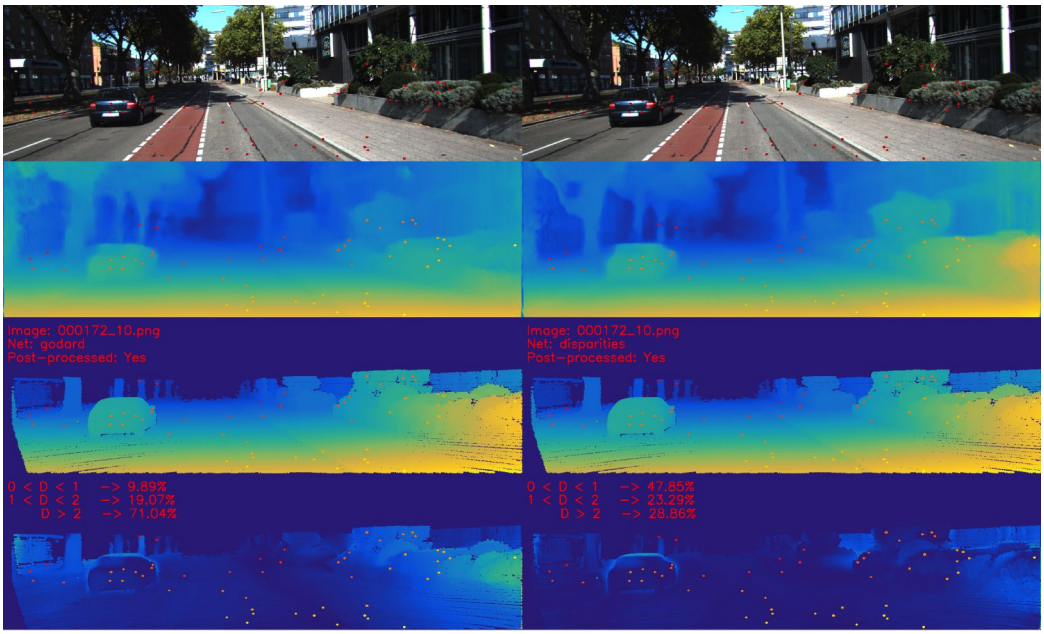

Figure B.1

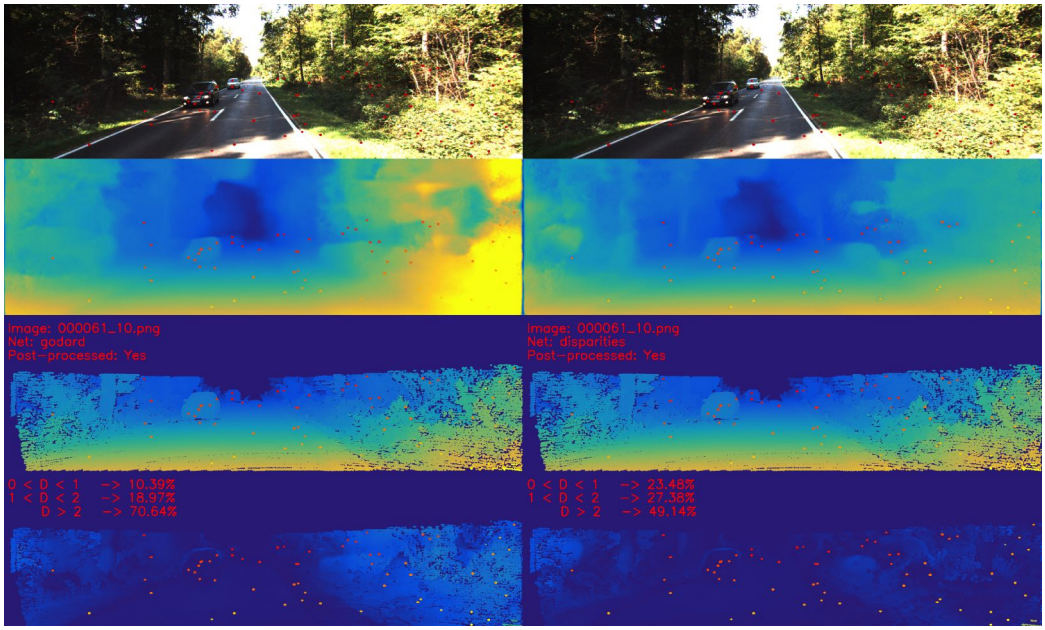

Figure B.2

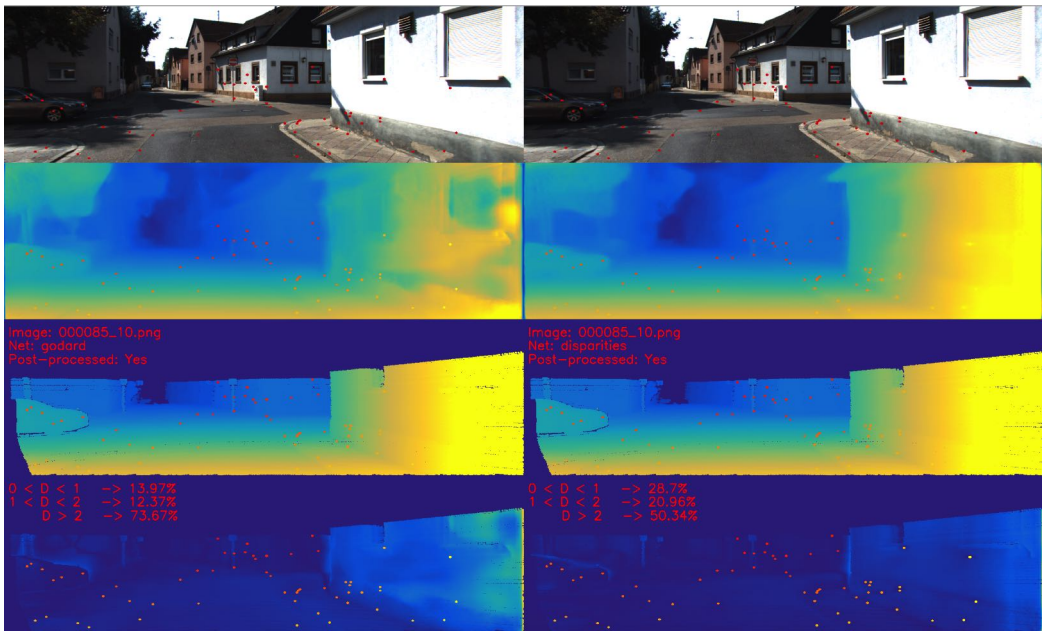

Figure B.3

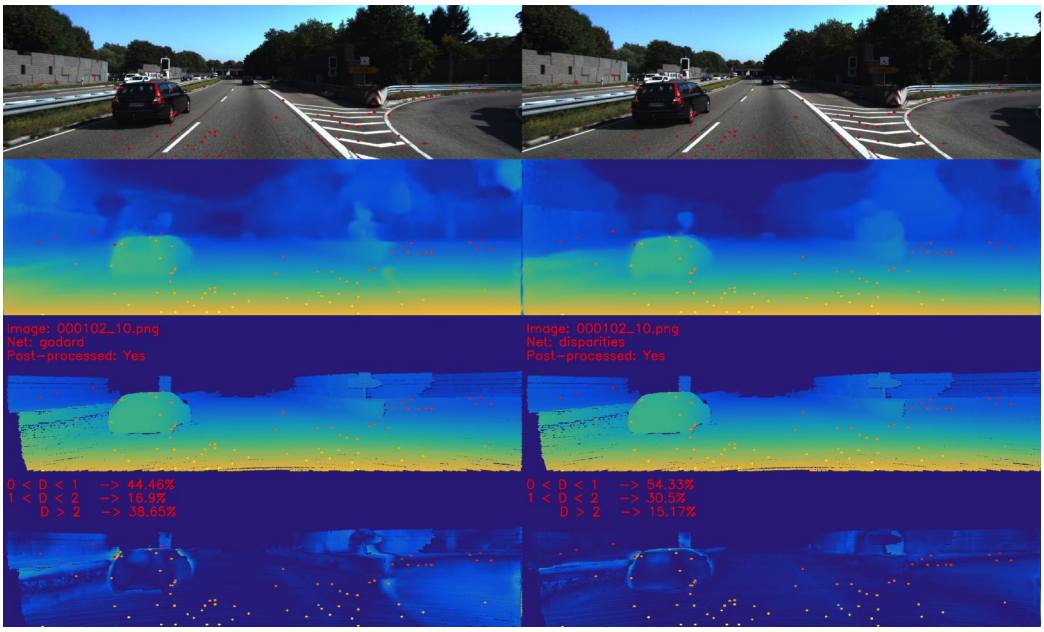

Figure B.4

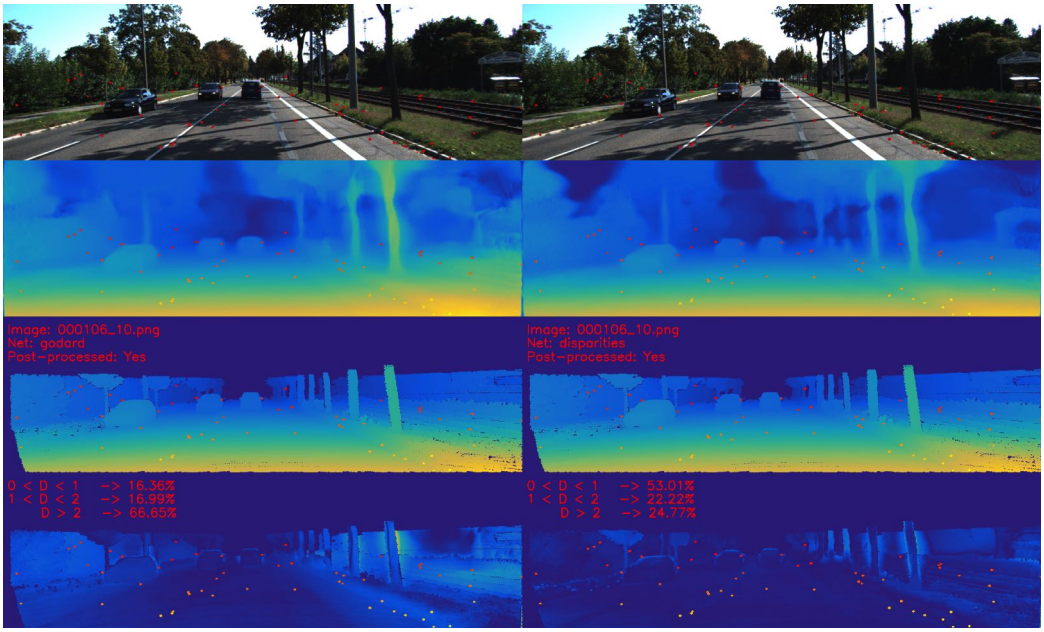

Figure B.5

Appendix C

UVR ZED dataset

Since Godard's model [7] works perfectly outside but poorly in indoor environments, we wanted to fine-tune on a self-crafted dataset.

C.1 Creating the Dataset

Since Godard's model needs just stereo images to train on, we created a dataset with a ZED stereo camera[1] which creates rectified stereo images and depth from a stereo matching algorithm. We used the depth from ZED camera as a sort of ground truth to evaluate on. It's not reliable, but it gives an idea of the real depth values.

Figure C.1: The ZED Stereo Camera

After creating a script that saves a pair of rectified stereo images and the related depth map at 10fps, we shot 12 mixed scenes indoor in the office and outdoor, around the office area, for a total of 8863 stereo pair images.

[1] https://www.stereolabs.com

The RGB images are 1280×720 PNG files, rectified and synced taken with a baseline of $12cm$. The depth maps are stored in 16 bits PNG files with a factor of 3276.75 to allow a good precision in stored values. The camera was set with a focal length at $671.6px$.

Figure C.2: An example of outdoor shoot. On top the RGB image and below the depth map, which is quite limited by the baseline: the maximum depth that the camera could get is about 14 meters.

C.2 Fine-tuning

We trained on this new dataset for 50 epochs with same configuration as we did for KITTI and Cityscapes and you can see on Figure C.3 that the resulted disparity maps are really good looking already from the 20-th epoch.

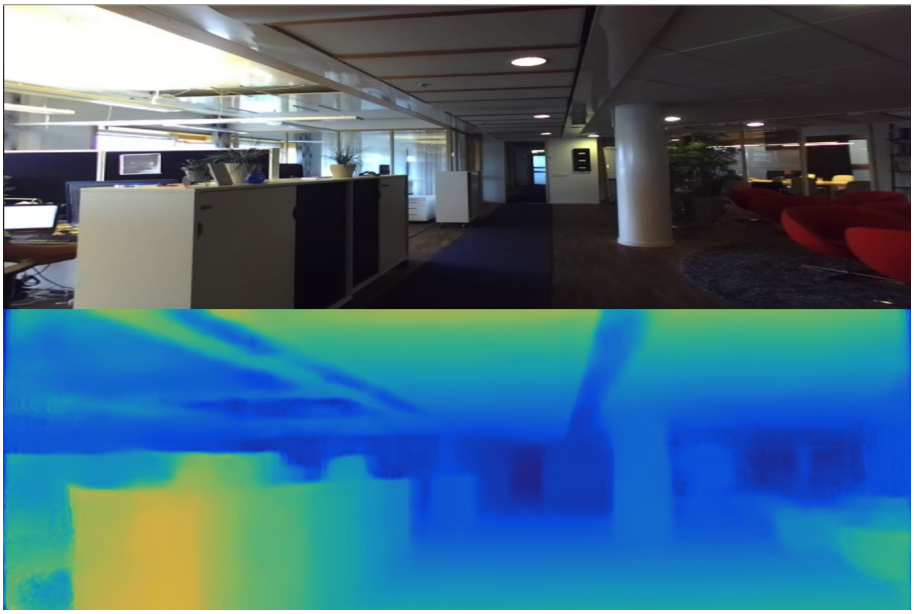

Figure C.3: Prediction on UVR dataset.

With the tool presented in Appendix A, we did a comparison between a model trained on Cityscapes and a model fine-tuned on the UVR dataset. In Figure C.4 and Figure C.6 we can see a comparison between a model trained on Cityscapes on the left and a model trained on Cityscapes and fine-tuned on UVR on the right. Specifically in Figure C.4, the disparity map predicted by the model trained on Cityscapes is good looking, but it has the wrong scale, as we can see by the L1 difference with the depth from the ZED camera. On the other hand, fine-tuning fixes the semantic errors, as we can notice in Figure C.6 where the ceiling is missing completely on the left. That's due to a semantic overfitting by the Cityscapes images: a large portion of the dataset contains images with sky and no one has a ceiling in it.

Some examples are briefly given below.

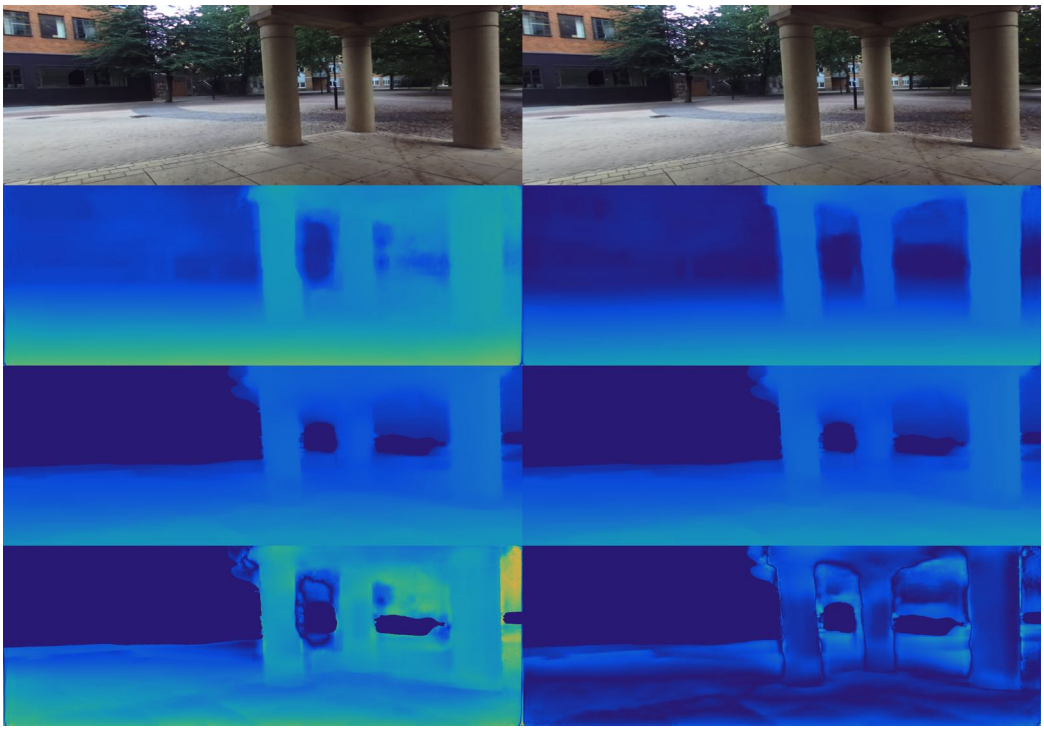

Figure C.4: Comparison Cityscapes-model and UVR-fine-tuned model.

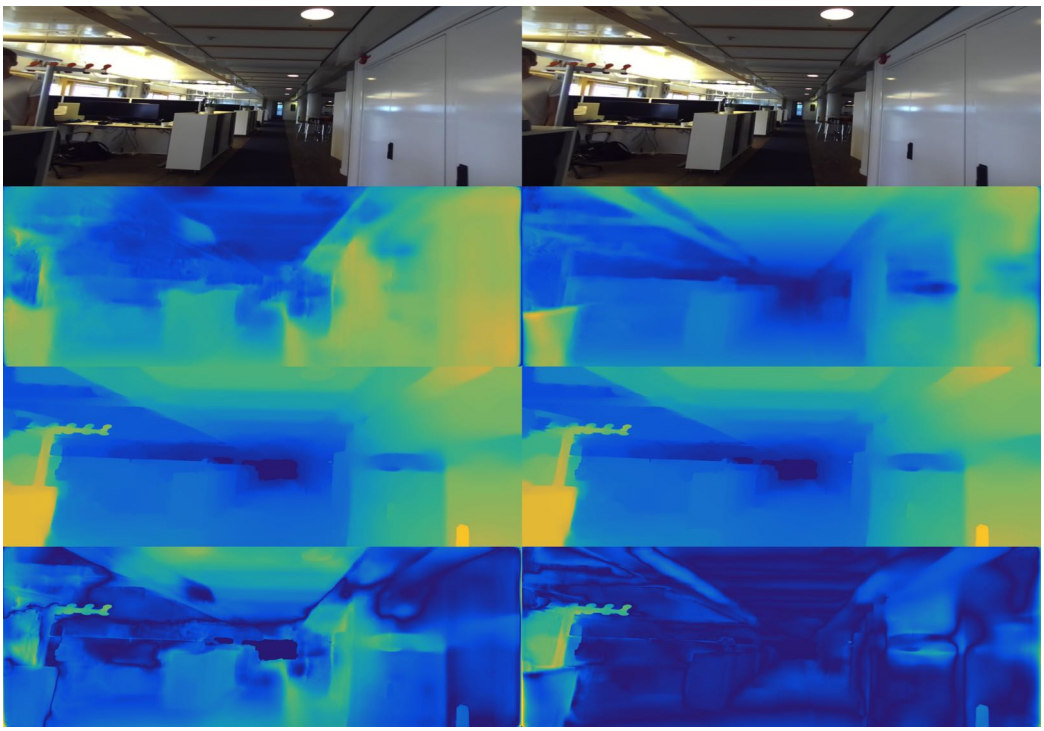

Figure C.5: Comparison Cityscapes-model and UVR-fine-tuned model.

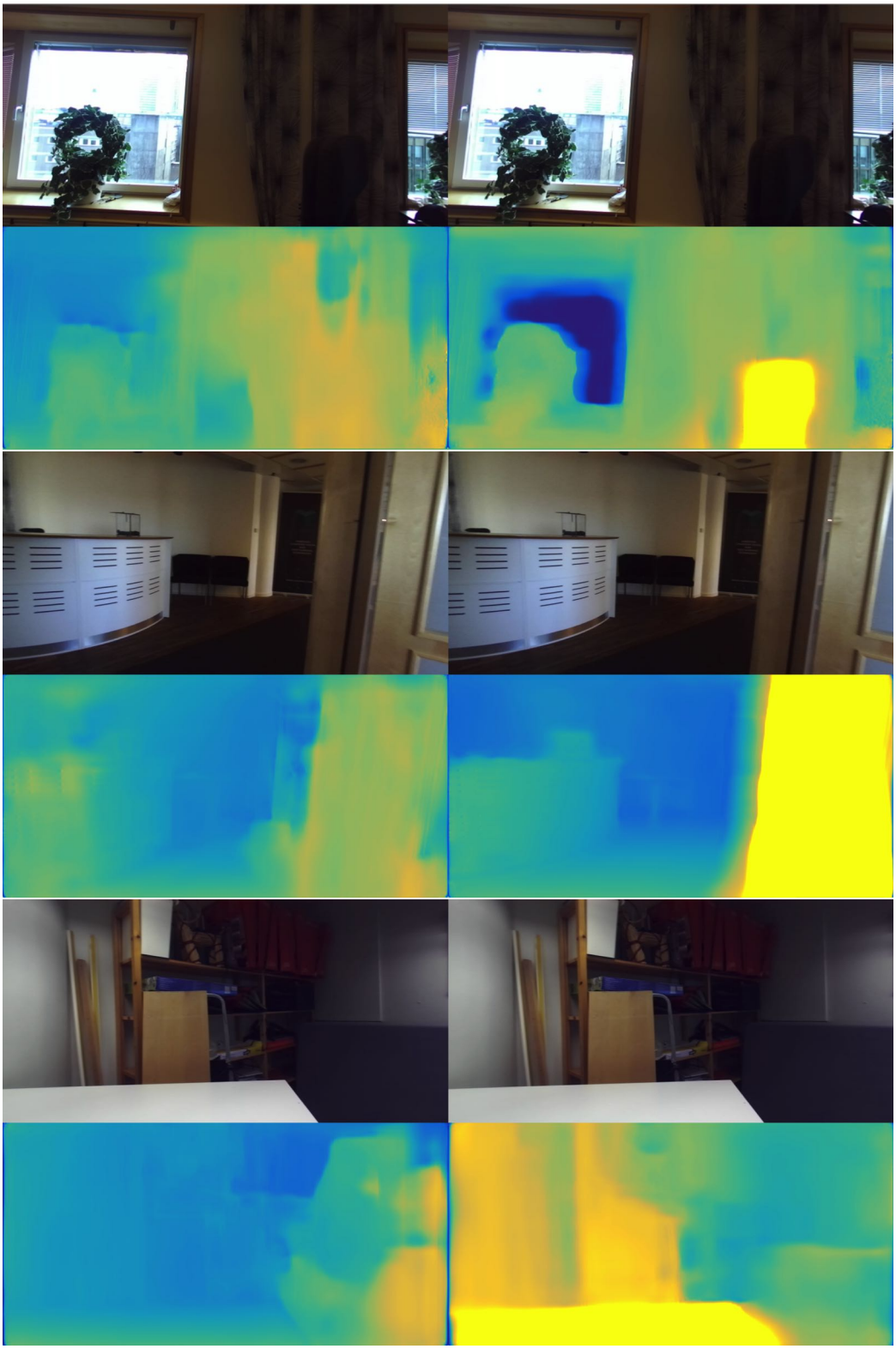

Figure C.6: Comparison Cityscapes-model and UVR-fine-tuned model.

Appendix D

Demo with Pangolin Visualizer

A dense depth map could potentially reconstruct a 3D scene just from one image. That's what we did with the Odometry Sequence 03 provided by KITTI in a demo on a 3D visualizer.

Figure D.1

We processed every RGB image with our best model to get depth maps as PNG files. With the image and the related depth map, we created a point cloud and fed it to Pangolin[1], which is a lightweight visualizer for managing

[1] https://github.com/stevenlovegrove/Pangolin

OpenGL display and interaction. The user can indeed change frame using A and D keys and interact with the point cloud using the mouse.

Figure D.2: Two different views of the same scene coming from just one image and its depth map.

The demo goes through all the sequence, showing the point clouds of all the frames along with the dense disparity map and the sparse disparity input from slam. Every point cloud is created runtime and displayed instantly after pressing either A or D to change frame.

Figure D.3: Example where the sparse disparity points from SLAM are showed.

Figure D.4: Example of the demo with the dense disparity map

The demo is available on YouTube at https://youtu.be/hEeQXNva2_s.