Single Image Plane Reconstruction using Manhattan World Constraints

Diogo Jordão Rodrigues de Oliveira

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Dr. Pedro Daniel dos Santos Miraldo

Examination Committee

Chairperson: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Dr. Pedro Daniel dos Santos Miraldo
Member of the Committee: Prof. Jacinto Carlos Marques Peixoto do Nascimento

February, 2021

Single Image Plane Reconstruction using Manhattan World Constraints

Diogo Jordão Rodrigues de Oliveira

February, 2021

Declaration:

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the *Universidade de Lisboa*.

Declaração:

Declaro que o presente documento é um trabalho original da minha autoria e que cumpre todos os requisitos do Código de Conduta e Boas Práticas da Universidade de Lisboa.

Abstract

Many indoor environments have objects with planar proprieties and are arranged as propitious to exploit their planes' normals alignment. These scenarios are ideal for a Manhattan World assumption, stating that all planes in a scene are aligned with one of the three dominant directions. In this master thesis, we propose a novel deep Neural Network, called MW-Net, for Manhattan planes detection and reconstruction, receiving a single RGB image as input. The end-to-end network learns to predict a rotation from the camera to the MW coordinate system, probabilistic segmentation masks, and an offset/depth map. The proposed method does not have a restriction on the number of planes that can predict. MW-Net was trained on ScanNet, and we extracted over 45000 ground-truth data. It uses a Dilated Residual Network for feature extraction, followed by two ramifications i) Global pooling for rotation prediction; ii) Pyramidal pooling for image segmentation and offset/depth map. MW-Net outperforms PlaneNet on segmentation accuracy, using less architectural complexity, since we do not use a DCRF, unlike PlaneNet.

Keywords: Manhattan World, Manhattan planes reconstruction, MW-Net, deep Neural Network, plane detection, Dilated Residual Network.

Resumo

Muitos ambientes interiores são constituídos por objectos com propriedades planares e a sua disposição é propícia a explorar os alinhamentos das normais dos planos. Estes cenários são ideais para a Manhattan world assumption, que afirma que todos os planos numa determinada cena estão alinhados com uma das três direções dominantes. Nesta tese de mestrado, apresentamos uma nova rede neuronal profunda, chamada MW-Net, para deteção e reconstrução de planos Manhattan recebendo unicamente uma imagem RGB como entrada. A rede "end-to-end" aprende a estimar uma rotação do referencial camera para o referencial Manhattan World, uma segmentação de imagem e um mapa offset/profundidade. O método proposto não tem qualquer restrição quanto ao número de planos que pode deduzir. A MW-Net foi treinada no dataset ScanNet, e foram extraídos mais de 45000 dados ground-truth. Foi usada uma Dilated Residual Network para extração de "features", seguida de duas ramificações i) Global pooling para prever a rotação; ii) Pyramidal pooling para a segmentação da imagem e mapa offset/profundidade. MW-Net supera o PlaneNet, um método estado de arte, e faz-lo com uma arquitetura menos complexa.

Palavras-chave: Manhattan world, reconstrução planar, MW-Net, rede neuronal profunda, deteção de planos, Dilated Residual Network.

Acknowledgements

First, I would like to thank my parents and my grandparents for all the support given. Then, I would like to sincerely thank my supervisor Pedro Miraldo and co-supervisors André Mateus and Gonçalo Pais, for their patience and availability through these days. I want to thank all me friend from IST and Figueira da Foz. A special thanks to my closer friends João Santos, Pedro Rodrigues, José Luís, André Godinho and Alexandre Abreu. Finally, I want to thank Valter, Leonardo, Diogo, Luís, and my supervisors for turning our work meetings easier and funny, during these difficult times.

Contents

List of Figures

List of Tables

Acronyms

CNN Convolution Neural Network. 6, 7, 10

Conv Convolution. 20, 21

CV Computer Vision. 1

DL Deep Learning. 1, 6, 7, 8

GPU Graphical Processing Unit. 23

HOG Histogram of Oriented Gradients. 6

IOU Intersection Over Union. 34

ML Machine Learning. 1

MW Manhattan World. xiii, 1, 2, 3, 8, 10, 11, 12, 13, 14, 15, 16, 17, 19, 23, 25, 29, 33, 37

SVM Support Vector Machine. 6

Chapter 1

Introduction

Computer vision (CV) is one of the most active research topics in computer science. In the early 2000s, several Machine Learning approaches to CV problems brought some impressive results (see [57, 9, 13]). These results sparked the interest in ML methods such as Support Vector Machines [26] and, in particular, on neural network architectures [32, 25]. With Deep Learning (DL) emergence in [32], many works have exploited these methodologies, achieving remarkable improvements [20, 19, 44, 43]. Indeed, DL methods applied to CV topics have been trending, and this relationship translated in many state-of-the-art methods on Object Detection [47, 44, 37], 3D Vision [43, 39, 51], and Tracking [54, 41].

Works [32, 25] in deep neural network architecture have been an essential role in the success of many recent methods, like [44, 47, 24]. Residual networks [25] (or Resnets) made it possible to increase the number of convolution layers, making the neural networks deeper while avoiding the undesirable vanishing gradient problem [3, 21]. This improvement on deep architectures lead to the development of state-of-the-art frameworks for object detection, *e.g.* R-CNN [20], Fast R-CNN [19], Faster R-CNN [47], YOLO [44] or YOLO 9000 [45].

This thesis focuses on the Planes' reconstruction problem, which has been extensively studied in recent works (see [36, 35, 58]). Although numerous works exploit new DL approaches, non-deep methods use more traditional approaches on Plane Detection topics, such as 3D Piecewise Planar Reconstruction [52] or Semantic Segmentation [62]. One example of a more classical approach is the Manhattan-world Stereo [16], which works under the Manhattan World (MW) constraints, an approach that we also follow.

Concerning DL approaches to Plane detection, methods like PlaneNet [36], PlaneRCNN [35] or PlaneRecover [58] brought significant improvements in terms of accuracy and run-time performance.

Usually, the Human being tends to build objects with planar surfaces on their structures. Many Deep Learning architectures ([44, 47, 24, 45]) can detect these objects, and, consequently, one can use these architectures developed for object detection and extend it to planar surfaces [35, 36]. Although many plane detection methods share some architectural similarities with object detection ones, they also share some problems. For instance, PlaneNet [36] struggles on small plane identification in a crowded planar scene. This difficulty increases with the restriction on the number of planes predicted (PlaneNet only estimates ten planes). Still, PlaneRCNN [35] is an example of significant improvement, having no restriction on the number of planes predicted, allied to an increase on accuracy/time performance. PlaneRCNN uses a more complex architecture than PlaneNet to achieve this purpose. In this master thesis, we propose a novel method that tries to overcome PlaneNet's problems with less complexity than PlaneRCNN.

Therefore, we propose the MW-Net, a novel deep neural network for detecting planes that satisfy the MW constraints. The MW Assumption states that all planes in a scene must be parallel or orthogonal between each other. These planes, Manhattan planes, have their normals aligned with one of the MW coordinate system dominant directions (basis vectors). We are aware that this approach will not recognize some planar surfaces whose normals do not respect the MW constraints. However, since many indoor scenes are composed of a large set of planes aligned with one of the dominant directions, it is possible to reconstruct almost the full planar scene with this approach, trying to neglect the less significant planes. An MW approach gives some flexibility to the proposed method by eliminating any restriction of having a pre-declared number of planes to be predicted.

MW-Net receives an RGB image and outputs: i) a rotation, represented by a quaternion, from the camera to the MW world; ii) probabilistic segmentation masks of each plane; iii) and an offset and depth maps for planar and non-planar, respectively. The quaternion is further converted to a rotation matrix 3×3 and it is constituted by the MW dominant directions. Since planes' normals are aligned with MW axis, this rotation is used to identify the planes's normal parameters.

(a) Input (b) Segmentation-gt (c) Segmentation

Figure 1.1: Example of MW assumption on an indoor scene. In figure 1.1 (a) is possible to see the RGB image of an indoor scene, and in 1.1 (b) the respective ground-truth segmentation from the input figure. The indoor scene is full of planar regions and an entire scene can be modelled using plane reconstruction. In figure 1.1 (c) it is shown an example of the MW-Net's performance.

1.1 Motivation

Indoor scenes are human-made environments built, in general, using planar surfaces. Besides, since we tend to build regular structures, it is usual to find a large number of planes that are orthogonal or parallel to each other. Also, many objects arranged in a room align with the room's layout. Figure 1.1 is an example of this property. Figure 1.1(a) shows an indoor scene where one can notice the planar and non-planar surfaces. The planes segmented in Figures 1.1(c) and 1.1(b) show the MW constraints explained before in a real-world scenario.

MW-Net is a deep neural network that receives an RGB image and outputs a rotation from the camera to the MW reference frame, four probabilistic segmentation masks (one for each MW dominant directions and one for non-planar region), and an offset/depth map. Planar reconstruction can be useful in diverse domain topics, such as robots navigation or augmented reality. It can: i) identify planar surfaces, and deliver tools for robot navigation; ii) help place objects on planar surfaces and orient them in regular spaces on augmented reality applications.

1.1.1 Contributions

One of the major difficulties of reconstructing a planar scene is the number of different planes in the environment, with different orientations and dimensions. Our method tries to overcome this problem by applying a Manhattan World approach to the scene.

Despite neglecting some planes, this approach tries to identify the most significant ones by defining MW dominant direction as propitious to it. This way, MW-Net can reconstruct almost the full planar scene. In the experimental results, we show that the proposed architecture is able to estimate the mapping of the planes more accurately than the state-of-the-art PlaneNet method. PlaneNet uses a Dense Conditional Random Field for segmentation results refinement, while MW-Net does not. MW-Net can predict Manhattan planes without any restriction and does it with less complexity than PlaneNet.

1.2 Thesis Outline

The present master thesis has five sections. Chapter 2 describes related work. It describes methods for object detection, and plane detection approaches. Chapter 3 describes the methods used, i.e., Dataset preparation, in order to allow the network's training. Also, Chapter 3 describes how planar reconstruction can be achieved through the network's output. Chapter 4 focuses on MW-Net architecture and training. Chapter 5 describes the results and compares the MW-Net with the PlaneNet [36]. Chapter 6 presents the final conclusions.

Chapter 2

Related Work

This chapter describes recent developments in object detection using Deep and Non Deep learning methods. It presents the state-of-the-art methodologies for Plane Detection in section 2.2. Finally, section 2.3 presents Manhattan World approaches on several topics, aiming at showing their utilities.

2.1 Object detection

Object Detection is one of the topics of great interest in Computer Vision. Usually, most Object Detection methods use bounding boxes to predict the object location on the image. Most of these also determine image segmentation masks, assigning a value per class for each pixel. The class with the most significant probability is the one that pixel belongs. The bounding box prediction is a regression problem (it computes a fixed number of parameters per box, from a set of features). The image segmentation is a classification problem (for each pixel it is assigned a class).

There are tons of detection methods in the Computer Vision research community, such as Object Detection (see [44, 20]), which usually is a general detection, i.e., identifies objects as well as pedestrians and animals. Some methods focus on a particular kind of detection, such as Pedestrian detection (see [6, 12]) or Plane Detection (see [36, 35]). These topics have attracted diverse industries, recognizing the potential of its use as propitious to the development of their fields. On car industries, Object Detection can deliver tools to achieve desirable results on Autonomous Driving (see [34]). Tesla is a great example of a car enterprise involvement in significant research works exploiting

Computer Vision topics (see [28]), being their Autonomous Driving work on the spotlight for the last years. Their cars have cameras for object and pedestrians recognition. They also use them for video surveillance guaranteeing the cars' security. However, most of their works are not publicly available due to car industry competitiveness. It is evident how Artificial Intelligence systems can benefit from exploiting these topics.

In the last two decades, we faced significant developments in Object Detection. For example, in the 2000s, with the development of Support Vector Machines (SVM), [26], machine learning approaches introduced SVMs in their architectures, as classifiers, leading to state-of-the-art results (see [9, 13]).

Although having great results at the time, developments on neural network architectures had a significant impact, for example by changing the way the features were extracted; the previous methods use handcrafted ones such as Histogram of Oriented Gradient (HOG) [40, 15] or SIFT [38]. Convolution Neural Network (CNN) [33] started as a trend when AlexNet [32] presented outstanding results, competing on ImageNet Large Scale Visual Recognition Challenge [48], with a top-5 test error rate of 15.3%.

Since then, several works have followed this approach and added significant contributions to the topic (see [50, 20]).

In the following sections, this evolution is further described. Section 2.1.1 reports Non Deep learning methods for object detection. Section 2.1.2 describes the state-of-the-art methods using DL approaches.

2.1.1 Non Deep methods

This section describes Non Deep approaches to object detection and in this section, it is included the Machine Learning ones. Some of the most effective methods use HOG, a feature descriptor, often used in Computer Vision. HOG descriptors are mainly used as features for object detection such as pedestrian detection [9, 13]. These descriptors usually feed a Support Vector Machine for classification. In another significant study, back in 2010, [12] claims to be the fastest detector with 6 fps, on high accuracy rates. There is also Scale Invariant Feature Transform (SIFT) [38], which descriptors can feed a classifier for object detection, as an alternative to HOG.

In face detection, several works are presenting remarkable results. For example, [27] uses two layers for classification; the first layer with component classifiers that identify

parts of the face, and the second layer with a classifier that combines the first layer's output and makes a final face detection. The method uses Support Vector Machines classifiers.

Template Matching [4] uses a template image containing one example of the desired object and slides it over the source image to detect identical ones in it. During this process, it compares the template with the patch of the source image under the template image. It uses a compare method, *e.g.* Minimum Square Difference, to evaluate the similarities, and if this difference is under a given threshold, it will consider the patch as containing a object equal or similar to the one in the template.

The evolution in the 2000s was evident, and for the last five years, significant developments were using DL. The next section describes the current state-of-the-art methods using Deep Learning.

2.1.2 Deep methods

As stated in section 1, in the last decade, DL approaches to Computer Vision problems have been seen in several works (e.g., [32, 20, 36]).

R-CNN [20] takes an RGB image as input and extracts 2000 region proposals, using Selective Search [56]. Each region proposal feeds a CNN for feature extraction, and a class-specific linear SVMs classifies the existence of an object. It also estimates four parameters for bounding box regression. R-CNN method improves the accuracy comparing to other State-Of-The-Art algorithms, but it has run-time problems, namely when training.

Instead of extracting 2000 proposal regions and feed each one to a CNN for feature extraction, without sharing computation, Fast R-CNN [19] feeds the entire image to a CNN, making a single feature extraction for the whole image, then extracts region proposals. This change improved R-CNN's run-time drastically, fixing time performance issues.

Faster R-CNN [47], removes the selective search algorithm for region proposal computation, and replaces it with a Region Proposal Network (RPN). Faster R-CNN initially takes an image and feeds it to a CNN, obtaining a feature convolution map. Then, the RPN slides through the feature map, receiving $n \times n$ windows as input and outputs a pre-determined number of region proposals for each window. The second stage is sim-

Table 2.1: Methods performance comparison taken from [37]

Method	FPS	mAP
Faster-RCNN [47]	7	73.2%
YOLO [44]	45	63.4%
SSD [37]	59	74.3%

ilar to Fast R-CNN, where is applied to a region of interest, ROI pooling, and finally, performs classification and bounding box regression.

YOLO [44] algorithm receives an RGB image and can detect objects, just resorting to a single neural network. For this purpose, YOLO divides the input into several grids, and each grid is responsible for predicting a fixed number of bounding boxes and confidence scores for each box. YOLO overcomes R-CNN and its variants in run-time performance. Further developments on YOLO, resulted in some variants, where YOLO 9000 [45] and YOLOv3 [46] are included.

In its turn, SSD [37] is a real-time object detector that uses a single network for object detection, does not need any region proposal network. It is simpler to train than Faster-RCNN, and significantly improves the speed for high-accuracy detection. Table 2.1 compares Faster-RCNN [47], YOLO [44], and SSD [37] in accuracy and run-time.

Mask R-CNN [24], that inspired PlaneRCNN's method [35], is an extension of Faster R-CNN. At first, it shares Faster R-CNN's main ideas, CNN for feature extraction and RPN for region proposal. The modifications occur on the second stage, where Mask R-CNN adds a binary mask for each ROI to the outputs of the Faster R-CNN, maintaining architecture similarities with it.

2.2 Plane detection

Object detection inspired other research works to focus on a more restricted kind of detection. Plane Detection is one of them. Plane detection is a research topic way before DL became a trend. Many research works applied a more traditional approach to this problem [16, 18, 52, 53]. Manhattan-world stereo (MWS) [16], working within the constrained space of Manhattan-world scenes, uses Multi-view Stereo (MVS) [49] to reconstruct a set of oriented 3D points (positions and normals). These Normals extract the dominant axis, and the positions generate axis aligned candidate planes.

Candidate planes are going to be used as hypotheses on Markov Random Fields depth-map reconstruction. "NYU-Toolbox" [52] is similar to MWS but does not works within the MW constrained space, and extracts its planes hypotheses using RANSAC [14].

Despite the outstanding results, there was the need to simplify the input requirements, since most of these methods require multiple views or depth information to work.

With the emergence of Deep learning, Plane detection research works [36, 35, 60] start to exploit deep neural architectures, obtaining remarkable results.

PlaneNet [36] uses a single RGB image as input and predicts plane parameters, segmentation masks, and a non-planar depth map. The network architecture consists of a Dilated residual Network [61, 5] for feature extraction, followed by two ramifications. The first ramification has a Global pooling followed by a fully connected layer for plane parameter's regression. The second ramification has a pyramidal pooling followed by a convolution layer for image classification and another convolution layer for non-planar depth map modelling. PlaneNet outputs a non-planar depth map, being the planar depth map determined using the plane parameters, only possible knowing the camera intrinsic parameters [23]. Due to its simplicity and results, our method builds on top of PlaneNet. Our network architecture differs in the output prediction from theirs, leading to changes in training. Our purpose is also different from theirs. MW-Net reconstructs Manhattan planes without any pre-declared number of planes, while PlaneNet detects unconstrained planes needing a pre-declared number of planes predicted.

PlaneRecover [58] also uses a single network for plane detection. It receives an RGB image as input and outputs a planar segmentation map, that segments the input in several planes and non-plane objects and the plane's parameters in 3D space. Similar to PlaneNet it predicts a limited number of planes, only estimating five planes per scene. PlaneRecover distinguishes from the other methods by approaching the problem with unsupervised learning, led by difficulties on dataset's ground-truth extraction. A piecewise planar 3D model of the scene can be built, using the network's output.

PlaneRCNN [35] differs from PlaneNet by using a variant of Mask R-CNN [24] for detection, and a refinement network for segmentation Mask improvement. Plane detection is made by predicting each plane parameters and segmentation mask. PlaneRCNN presents a novel loss function, which improves plane-parameter and depth map accuracy via end-to-end training. The referred method presents state-of-the-art results, overcom-

ing PlaneNet's limitation related to the restriction of the number of planes that can be predicted per scene.

Finally but not least, [60] is divided into two stages. In a first stage, it trains a CNN to obtain planar/non-planar segmentation map and pixel embeddings, followed by a mean shift clustering algorithm to generate plane instances. On the second stage, a network branch is trained to predict pixel level plane parameters. It also does not have a restricted number of planes that can be detected.

Both PlaneRCNN and [60] do not have any restriction on the number of planes but they achieve this using a more complex architecture than PlaneNet.

2.3 Manhattan World assumption

Exploiting environment geometry is not a novel approach, and a MW can take advantage of these characteristics. On 3D reconstruction, there are many MW approaches [16, 17, 10], but this topic is not the only taking advantage of it.

There are research studies using the MW constraints, for instance in navigation [7, 11], where indoor and outdoor scenes are designed on a Manhattan three-dimensional grid. In [7], they state that the important signs for navigation are aligned with one of the directions of MW, and facilitate navigation.

Chapter 3

Dataset Creation

Many objects are made of planar surfaces, and most of the time, they are arranged with other planar surfaces. An indoor environment layout usually is composed of six planes orthogonal or parallel to each other. Frequently, these planar objects are arranged according to the layout, and many times their planes are aligned with one of the three dominant directions. Situations like these attract MW approaches, which can detect a significant set of planes that have their normals aligned with MW base vectors. The MW base vectors were computed considering the most significant planes, to avoid neglecting many non-constraint planar surfaces.

The MW assumption assumes that all planes in a scene are aligned to one of the three dominant directions. If the plane's normal is aligned, then the plane will be detected. On the other hand, if the plane's normal is not aligned with one MW axis, the plane will be detected but as part of the non-planar region, not being counted as MW plane.

PlaneNet [36] detects planes unconstrained by the Manhattan World restrictions. For PlaneNet training, their authors extracted ground-truth data from the ScanNet dataset, such as the planes' parameters, image segmentation, and the image depth map.

This ground-truth data is not suitable for our network's training since it does not respect the MW constraints. Working over the extraction mentioned, we defined the MW dominant directions to distinguish the Manhattan planes from the non-planar region, in each scene. With this in mind, it is possible to adapt the dataset to our purpose, and this Chapter presents it.

3.1 Notation description

In this section it is explained the elements representation through the present Chapter.

Let us consider the plane equation

$$ax + by + cz = d, \tag{3.1}$$

where (a, b, c) are the elements of the plane normal and d the offset. A plane can be represented by three parameters, $d \times (a, b, c)$. Vectors are represents with capital letter, and their scalars are with minor letter. Consider the 3D vector X,

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \tag{3.2}$$

where x_i is the i^{th} scalar from the respective vector. Rotation quaternions are represented as Q_a^b, being the rotation from a to b. The same logical is applied to the 3×3 rotation matrices represented as R_a^b.

3.2 ScanNet dataset

The ScanNet [8] is a large-scale RGB-D video database of indoor environments. For each scene, this dataset makes available annotations with estimated calibration parameters, camera poses, 3D surface reconstructions, textured meshes, dense object-level semantic segmentations, and aligned CAD models.

For PlaneNet purpose, it was extracted 51000 ground-truth piecewise planar data (50000 for training and 1000 for testing) from ScanNet. For this process, they directly fit planes to 3D points, using RANSAC with replacement, and project them to images. The resulting dataset will make available for each RGB image the image segmentation, plane parameters, image depth, and intrinsic camera parameters.

The resulting planes are not under the MW constrains. For each scene, it becomes necessary to extract the MW planes. With this in mind, it was extracted a rotation from the camera to the MW coordinate frame. This rotation is composed of three dominant directions, and the MW planes' normals are aligned with one of those directions. Section 3.2.1 explains this extraction and the modifications made to the ScanNet dataset after PlaneNet's processing.

Algorithm 1 MW dominant directions for each ScanNet dataset scene. PlaneNet authors extracted planes' parameters, image segmentation, and image depth map for ScanNet dataset. For our purpose, we need to distinguish Manhattan from non-Manhattan planes. For this process, we estimate MW base vectors, using normals from planes that contribute to reconstructing almost the full planar scene. Planes not aligned with any of the dominant directions are going to be considered non-planar regions.

Number_of_Planes $\leftarrow$ 20
P $\leftarrow$ Planes' normals
Largest_P $\leftarrow$ None
for i in Number_of_Planes **do**
  if (Size(P[i]) $>$ Size(Largest_P) $\wedge$ (P[i] has at least one orthogonal Plane)) **then**
    Largest_P $\leftarrow$ P[i]
  end if
end for
X $\leftarrow$ Largest_P
Y $\leftarrow$ Largest plane from the list of orthogonal planes to Largest_P
Z $\leftarrow$ X $\times$ Y
R $\leftarrow \begin{bmatrix} X & Y & Z \end{bmatrix}$
$\begin{bmatrix} U & D & V \end{bmatrix} \leftarrow SVD(R)$
$R_{MW}^{C} \leftarrow Udiag(1, 1, det(UV^T))V^T$
$R_{C}^{MW} \leftarrow {R_{MW}^{C}}^{(-1)}$
Distinguish Manhattan from non-Manhattan planes

3.2.1 Manhattan World Assumption & Dataset

The MW assumption states that all planes in a specific scene are orthogonal or parallel to each other, thus planes' normals must be aligned to one of the three dominant directions. The MW coordinate frame defines these dominant directions. The MW base vectors can be arranged to obtain a rotation matrix from the MW coordinate frame to the camera coordinate frame.

To compute the MW base vectors that better suits a specific scene, we had in consideration a similar process as the one presented in [22], and Algorithm 1 shows the process. When choosing the MW base vectors, it is desirable to have ones that capture the most significant planes. These planes have the largest number of pixels assigned in the image segmentation. For instance, considering a room, and having a broader view of the division, these planes are usually a wall or the floor.

Considering that, after the PlaneNet's dataset processing, we have access up to

twenty planes per scene, a planar segmentation of the image and the image depth map. The MW base vectors were computed as follows.

From Algorithm 1, initially, it was computed how many pixels were assigned to the i^{th} plane, using the image segmentation, and the inner products between it and all the others 19 planes. Notice that this process is made to all the planes in the list P. From the inner products, we obtain the orthogonal planes to each plane.

The first base vector determined is the one associated with the MW X-axis. It is set with the normal of the most significant plane in the scene, *i.e.* with more pixels assigned, under the condition of having at least one orthogonal plane on the scene's image segmentation. If the condition is not fulfilled, this process is repeated to the second largest plane and so on. The MW Y-axis is the second base vector defined, and it assigned the normal of the largest plane from the list of planes whose normal are orthogonal to the MW X-axis. Finally, MW Z-axis is defined by the cross vector between the MW X-axis and Y-axis base vectors. As it is possible to realize, these scenes must have at least two orthogonal planes; otherwise, they are discarded.

The X and Y planes' normals are not strictly orthogonal since we gave a threshold to the inner product, 0.1, below which two planes are considered orthogonal. With this in mind, if we organize the base vectors X,Y and Z as columns, we obtain a pseudo-rotation matrix $R^{MW} = [X, Y, Z]$ which points to the need of projecting it to the closest matrix on the SO(3) group.

A SO(3) matrix must respect the following conditions,

$$\begin{aligned} R^T R = R R^T = I \\ det(R) = 1, \end{aligned} \tag{3.3}$$

and in order to do it, we applied the Singular Value Decomposition (SVD) to the pseudo-rotation $R^{MW} = [X, Y, Z]$,

$$[U \Sigma V^T] = SVD(R^{MW}). \tag{3.4}$$

Now it is trivial to obtain the rotation from the camera to the MW coordinate frame,

$$R^C_{MW} = U diag(1, 1, det(UV^T)) V^T \tag{3.5}$$

where U and V^T are unitary matrices. Notice that from Section 3.1, R_{MW}^C is the rotation from the Manhattan World to the camera coordinate frame.

Once having the R_{MW}^C is now easy to find the rotation from the camera to MW coordinate frame, which is given by its inverse,

$$R_C^{MW} = {R_{MW}^C}^{-1}. \tag{3.6}$$

It is now possible to apply the MW constraints on each scene, and distinguish which surface is planar or non-planar. This data treatment was made as follows.

At this stage, the planes' parameters available are represented regarding the camera coordinate system's origin. To verify if a plane's normal is aligned to one of the MW base vectors, we need to have the planes' parameters seen by the origin of the MW frame's origin. We can easily achieve this by applying the rotation R_C^{MW}. Considering a plane's normal seen by the camera's coordinate system, N^C, it is now possible to obtain

$$N^{MW} = R_C^{MW} N^C, \tag{3.7}$$

where N^{MW} is the plane's normal seen by the MW coordinate system.

To distinguish Manhattan planes from non-planar surfaces, we have to verify which planes' normals are aligned with one of the three MW axis. For this purpose, it was done the inner product between the planes' normals and each one of the three MW coordinate axes, $[1, 0, 0], [0, 1, 0]$ or $[0, 0, 1]$. If any of the three inner products were over a pre-established threshold, 0.9, the normal would be considered aligned with the respective axis. From that point, the plane's normal would be replaced by the MW base vector which is aligned. However, if none of the inner products was over 0.9, the plane will become part of the non-planar class. The image segmentation is then updated based on this knowledge.

The image segmentation is obtained by assigning the specific class to each pixel. There will be four classes, one for each MW dominant direction and one for the non-planar region. Pixels that are assigned with a class $C \in [1, 2, 3]$ will belong to a Manhattan plane and the pixels with a class $C \in [4]$ will belong to the non-planar region.

For PlaneNet's purpose, their authors predicted the plane's normal and offset as a three-parameter vector, and the offset was the vector's norm. Still, since we are using MW base vectors to identify planes, predicting a normalized three-parameter vector, we only know the plane's normal but not the respective offsets. Knowing the Manhattan

planes and their mapping in the image, we need to make an association class/offset to each planar pixel. We find a solution to this problem, creating an offset/depth map for each scene. Further can be found at 3.3.1. The offset/depth map ground-truth was obtained by intersecting the MW with the original image segmentation, from which we get MW planes segments.

If a plane is a Manhattan plane, we assign the respective offset to their pixels, otherwise we assign the respective depth value to the pixel. To the non-planar region pixels we assign the respective depth value. This makes it possible to obtain the offset/depth map desired.

After the ground-truth extraction, we add the rotation from the camera to the MW coordinate frame, as a quaternion, Q_C^{MW}, the image segmentation with four classes, and the offset/depth map to the ScanNet dataset, allowing the network to train as expected. This ground-truth extraction will be useful for the training losses in Section 4.2.

If the rotations were predicted as 3×3 matrices, it would not be possible to guarantee that the network's rotations outputted would fill the SO(3) group requisites, see Equation 3.3. A possible approach would be to project it to the rotation group, but this solution would be computationally more complex. For this reason, and more discussed in the next chapter, we predict a quaternion over a rotation matrix, and need a quaternion as ground-truth.

3.3 Ground-truth data

This section introduces a more detailed description of the ground-truth data and consequently MW-Net output. In section 3.3.2 is referred how to compute the depth with only the pixels coordinates, planes' parameters. This process was used to compute the depth on the offset/depth map. This is also used on the depth results in Chapter 5.

3.3.1 MW-Net outcome and ground-truth data description

The MW-Net is a novel deep neural network that receives an RGB image, and it predicts a rotation Q_C^{MW}, four probabilistic segmentation masks, and an offset/depth map, using a single network.

From [55], a quaternion is a 4 length vector, $Q \in \mathbb{R}^4$, that can be represented by

(q_1, q_2, q_3, q_4) and has the form

$$Q = q_1 + q_2 i + q_3 j + q_4 k, \tag{3.8}$$

where q_1 is the real part and q_2, q_3, q_4 are imaginary parts. To know what are the MW base vectors, it is necessary to transform the predicted quaternion in a 3×3 rotation matrix,

$$R_C^{MW}(q) = \begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2-q_3q_4) & 2(q_1q_3+q_2q_4) \\ 2(q_1q_2+q_3q_4) & 1-2(q_1^2+q_3^2) & 2(q_2q_3-q_1q_4) \\ 2(q_1q_3-q_2q_4) & 2(q_2q_3+q_1q_4) & 1-2(q_1^2+q_2^2) \end{bmatrix}, \tag{3.9}$$

in order to have access to the rotation matrix's base vectors which, as already seen, encode the planes' normal in the scene. The transformation in equation 3.9 comes from the Rodriguez formula [2], for representing a rotation matrix,

$$R(\hat{N}, \theta) = I + \sin\theta[\hat{N}]_\times + (1-\cos\theta)[\hat{N}]_\times^2, \tag{3.10}$$

where $[\hat{N}]_\times$ is the matrix form of the cross product with the vector $\hat{N} = (\hat{n}_x, \hat{n}_y, \hat{n}_z)$.

The image segmentation output consists of four probabilistic segmentation masks per pixel. Each pixel is assigned four probabilities, one for each MW axis, resulting in three probabilities, and one probability for non-planar. The largest value will determine to which class the pixel belong. Each planar class is associated to one and only one MW base vector, [1,0,0], [0,1,0] and [0,0,1], and reciprocally. If we apply the rotation R_{MW}^C to each MW base vector, we obtain the planes' normal concerning the camera coordinate frame.

Last but not least, there is the offset/depth map. This was the solution found to predict the last plane parameter missing, the offset, and still be able to reconstruct the non-planar depth. On the offset/depth map, planar pixels are assigned with the offset and non-planar by the respective depth. To conclude, for each pixel it is known to which plane it belongs and which offset it is and, as a result, one can compute a depth per pixel for the entire image.

3.3.2 Planar Depth

Since our network predicts an offset/depth map, described on the previous section, we only know the non-planar surfaces depth. To compute the depth for each planar

surface, we need to solve the equation 3.11 in order to λ, for all the planar surfaces' pixels with coordinates (u,v).

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} R|T \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.11}$$

Equation 3.11 represents a 3D to 2D projection, where (u, v) are the pixel coordinates, (x, y, z) are 3D cartesian coordinates, R is a rotation matrix from the world to the camera coordinate frame, and T is a translation matrix. K is the intrinsic matrix that is given by

$$K = \begin{bmatrix} fs_x & 0 & c_x \\ 0 & fs_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.12}$$

where f is the focal distance, (s_x, s_y) are conversion factors from meter to pixel coordinates, and (c_x, c_y) are the coordinates from the principal point. The principal point is where the image plane intersects with the camera coordinate frame axis. The focal distance is the distance from the origin of the camera coordinate frame to the image plane.

On the equation 3.11 there are four unknown variables (λ, x, y, z). When we solve equation 3.11 for (x, y, z), these variables will be in function of λ, which is the desired depth. Considering $(x, y, z) = \lambda(x_\lambda, y_\lambda, z_\lambda)$, and knowing that pixel with coordinates (u,v) belongs to a plane with normal (a,b,c) and offset d, we can easily find the desired depth using the planes' equation,

$$\begin{aligned} &\lambda(ax_\lambda + by_\lambda + cz_\lambda) = d \\ \Leftrightarrow\ &\lambda = \frac{d}{ax_\lambda + by_\lambda + cz_\lambda}. \end{aligned} \tag{3.13}$$

This depth is not in meters, and it has a scale factor of 0.001.

Chapter 4

MW-Net: A plane detection network with Manhattan World constraints

MW-Net is a novel method for plane detection. It takes a 192×256 RGB image, 3 channels, as input and outputs a rotation quaternion, from the camera to the Manhattan World (MW) coordinate frame, four probabilistic plane segmentation masks, and an offset/depth map.

The rotation quaternion, a 1×4 vector, guarantees a rotation belonging to the SO(3) group. The rotation quaternion can be further converted to a 3×3 matrix, as seen in Section 3.3.1. This rotation is composed by the MW dominant directions seen by the camera coordinate frame, and we can easily obtain the Manhattan planes' normals making use of it.

For the image segmentation, the network predicts four probabilistic segmentation masks, where each pixel is assigned with four probabilities. There is one probability for each possible class: i) planes' normal is aligned with MW X-axis; ii) planes' normal is aligned with MW Y-axis; iii) planes' normal is aligned with MW Z-axis; iv) planes' normal is not aligned with any of the MW axes. The pixel belongs to the class that has the largest value. The resulting segmentation image output shape is 192×256.

The offset/depth map is a 192×256 matrix where each planar pixel is assigned the offset of the plane it belongs to. For non-planar pixels, it is assigned a depth value.

The network predicts all Manhattan planes, without any conditionality on the number of planes that can be estimated. Our network only detects and reconstructs planes aligned with one of the MW dominant direction. Indoor environments are propitious to this approach due to the large set of orthogonal and parallel planes in each scene, lifting

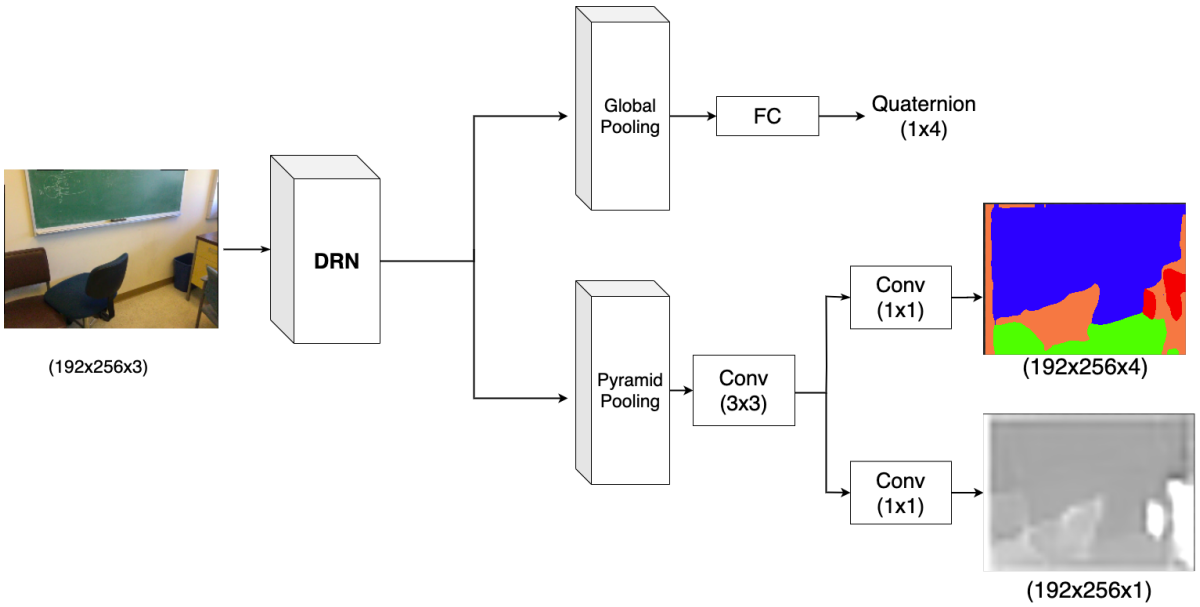


Figure 4.1: MW-Net's Architecture. It is constituted by a Dilated Residual Network (DRN), for feature extraction, and by two ramifications, a global pooling for rotation quaternion prediction and a pyramidal pooling [63] for the plane segmentation mask and a offset map prediction.

the mentioned restriction.

MW-Net uses a single neural network for the whole process, implemented in Pytorch (see [42, 29]).

4.1 Architecture

MW-Net's is a novel deep neural network, and its structure can be seen in Figure 4.1. It is constituted by a Dilated Residual Network (DRN), for feature extraction, and two ramifications: i) a global pooling for quaternion rotation prediction; ii) and a pyramid pooling [63], which, for its turn, ramifies in plane segmentation mask and an offset/depth map prediction branches. Over this chapter, when talking about convolution layers, it is represented its kernel size between parentheses. For example, a convolution layer with kernel size 3 is represented as Conv(3×3). From now on, when talking about tensors shapes, it is referred to their shapes as $width \times height \times channels$.

As already referred, the DRN in Figure 4.1 is a feature extractor block. DRN has a similar structure to Resnet, but instead of using standard convolutions in some layers, it

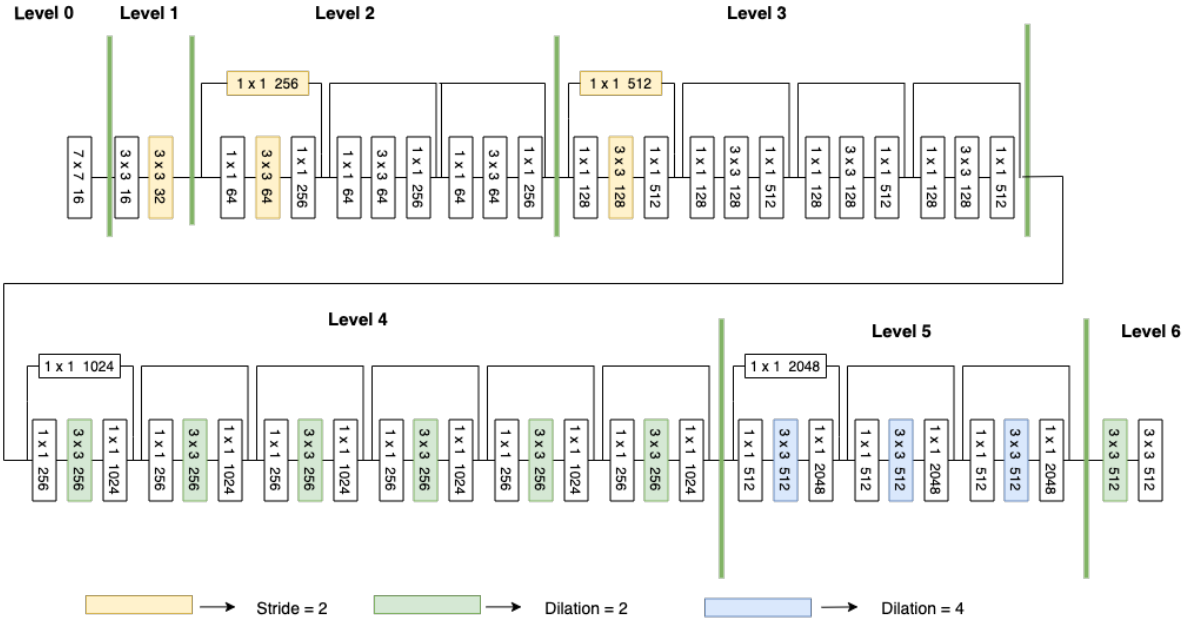


Figure 4.2: DRN-D-54 architecture. It is divided in seven levels. Bottleneck blocks form levels 2, 3, 4, and 5. Convolution with color yellow have stride 2, color green have dilation 2, and blue have dilation 4.

applies dilated convolution [59]. Dilated convolutions are convolutions where the kernel elements are spaced from each other, skipping some input points. For example, $D = 2$ means that the kernel elements have a gap between them; $D = 3$ means they are spaced by two. The DRN used is a DRN-D-54, which has 35.8M parameters.

The DRN-D-54 structure, represented in figure 4.2, was divided into seven levels. In level 0, there is a single standard Conv(7×7), and it is the starting layer that receives the RGB image. Level 1 has two convolutions, being the first one a standard Conv(3×3) and the second one is a Conv(3×3), with stride 2.

Bottleneck blocks form levels 2, 3, 4, and 5. A bottleneck block [25] has 3 convolution layers, starting with a Conv(1×1), followed by a Conv(3×3) and a Conv(1×1). Generally, the last layer outputs four times more channels than the first two, i.e., if the first two layers output 64 channels, the last layer will output $4 \times 64 = 256$ channels. The bottleneck block has a shortcut from its input to the outcome, allowing to skip the three convolution layers described, giving the block some flexibility to be just the identity if needed. This property helps to avoid the undesirable vanishing gradient problem.

Table 4.1 shows the number of output channels for each level, and the number of

Table 4.1: DRN-D-54 level description

Level	Channels in	Channels out	Bottleneck blocks
0	3	16	0
1	16	32	0
2	32	256	3
3	256	512	4
4	512	1024	6
5	1024	2048	3
6	2048	512	0

bottleneck blocks. Level 2 and 3 have 3 and 4 bottleneck blocks respectively, and their first block has the second Conv layer with stride 2. Level 4 and 5 have six and three bottleneck blocks, respectively, and the second convolution layer of each block is a dilated one, by 2 and 4 respectively. Finally, layer six is formed by two convolution layers. The DRN outcome will be of shape $24 \times 32 \times 512$ and will feed the Global pooling and Pyramid pooling block.

Global pooling in Figure 4.1 is simply an average pooling in two dimensions. The average pooling kernel has shape 24×32, resulting in an output of shape of $1 \times 1 \times 512$. Making this result going through a fully connected layer with 512×4 parameters, it results in the desired quaternion. This is another reason for using the quaternion for predicting the rotation. If we predicted the rotation as a 3×3 matrix, the fully connected would have 512×9 parameters, increasing its complexity.

Pyramid Pooling is a more complex block than Global Pooling, see figure 4.3. It receives the DRN's feature map and applies four different average pooling to it, using different kernels sizes. After pooling, each output goes through a standard convolution, with size kernel of 1, that outputs 128 channels. Each output goes through an upsample to obtain shapes equal to the input one, 24×32. Finally, the pooling's outputs are concatenated to each other, and with the Pyramid pooling input, obtaining a final tensor shaped as $24 \times 32 \times 1024$.

Finally, Pyramid's output goes through a standard convolution, with a kernel of size 3, outputting 512 channels. For the Segmentation Masks, there is a standard convolution with a kernel size of 1, and since there are four different classes, it outputs four channels. Bearing in mind that the output is of shape $24 \times 32 \times 4$, it is necessary to upsample it so that the resulting shape is $192 \times 256 \times 4$. For the Offset map, instead of a convolution

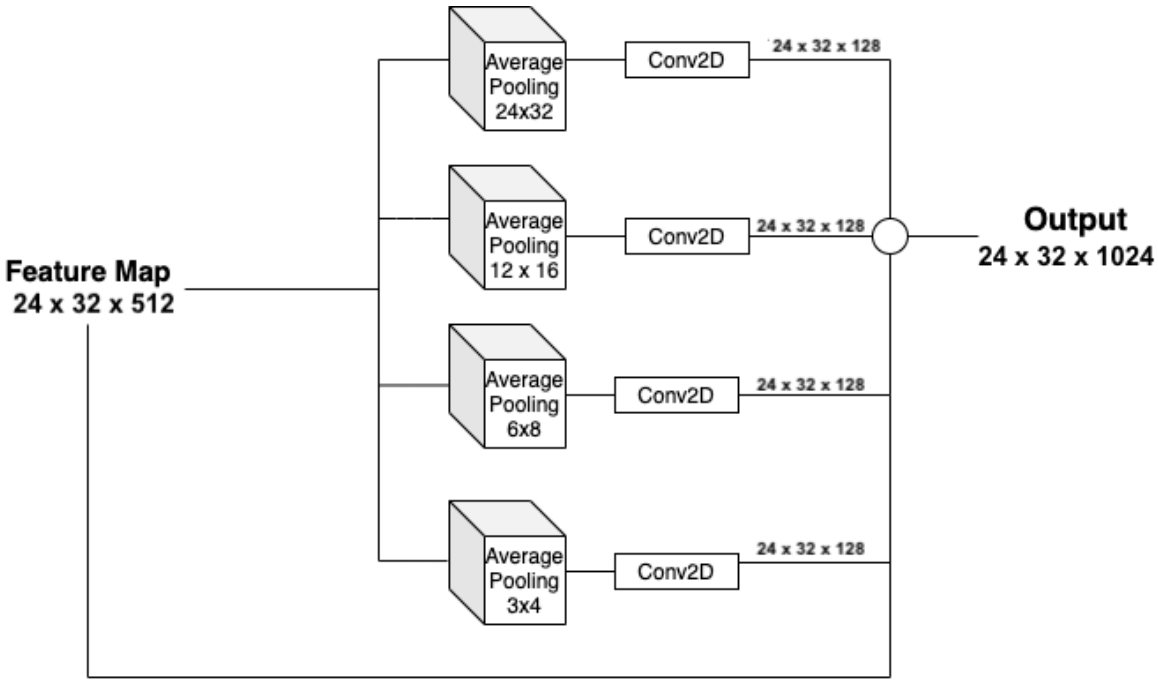


Figure 4.3: Pyramid pooling architecture. It receives the DRN's feature map and applies four different average pooling to it, using different kernels sizes. After pooling, each output goes through a standard convolution, with size kernel of 1, that outputs 128 channels. Each output goes through an upsample to obtain shapes equal to the input one, 24×32. Finally, the pooling's outputs are concatenated to each other, and with the Pyramid pooling input, obtaining a final tensor shaped as $24 \times 32 \times 1024$.

that outputs four channels, a convolution outputs a single output followed by the bilinear interpolation.

4.2 Training

MW-Net was trained on an indoor environment dataset, ScanNet, with ground-truth extraction for the sake of MW assumption. For training, it was used a pre-trained model of PlaneNet, trained by us for 50 epochs, since the only pre-trained model available was for their network implemented on Tensorflow [1] . This pre-trained model does not use the Dense Conditional Random Fields (DCRF) [31], used on PlaneNet, to refine the Segmentation prediction.

The proposed model was trained on a GPU GeForce GTX 1070 over 40 epochs, and

it was used 46710 samples for training. Network's learning was mini-batch learning, and each batch has eight data samples. Using mini-batch learning, the model updates its weights on a higher frequency; in the MW-Net case, this happens in every mini-batch, i.e., the model updates every eight samples from training data. Each sample includes the RGB image and the respective ground-truth data (quaternion, image segmentation, and offset/depth map), and the camera intrinsic parameters. The RGB image will be the network's input, and the ground-truth data will be part of the loss, as a reference to the network's outputs. Since GPU RAM is limited, Mini-batch learning is useful because it allows good memory management.

The optimizer used for training was the Adam Optimizer [30] with an initial learning rate set to 3×10^5. The optimizer is responsible for minimizing the loss, updating the model weights, and Adam computes adaptive learning rates per parameter. The loss is a multi-task one, and it is described in section 4.2.1.

4.2.1 Loss

The problem at hand is a multi-task learning problem, our network predicts three different outputs at the same time, resulting in three different losses, one for each output. As it is possible to see in section 4.2, the network learning is a mini-batch one, and the batch size, B, is 8 samples of data training. The overall loss is given by the sum of the three losses, regarding the mini-batch in question, leading to

$$Loss = \sum_{i}^{B} Loss^i_{quat} + Loss^i_{seg} + Loss^i_{offset}, \tag{4.1}$$

where $Loss_{quat}$ is the quaternion loss, $Loss_{seg}$ is the segmentation loss, and $Loss_{offset}$ is the offset loss. The Losses are represented as $Loss^i$ that is the Loss of the i^{th} sample of the mini-batch. Since the referred problem is multi-task learning, there are many problems adjacent to it. It is required to be careful with each loss that composes the overall loss. If a loss is much larger than the other two, it may lead to one or more losses being neglected. One can avoid this problem by applying weights to the different losses that compose the overall loss.

In figure 4.4, it is possible to observe the evolution of the loss per steps. It is obvious that was used mini-batch training just by looking to the figure, because of the "stairs" effect that causes to the loss. This effect is due to high frequency weights update,

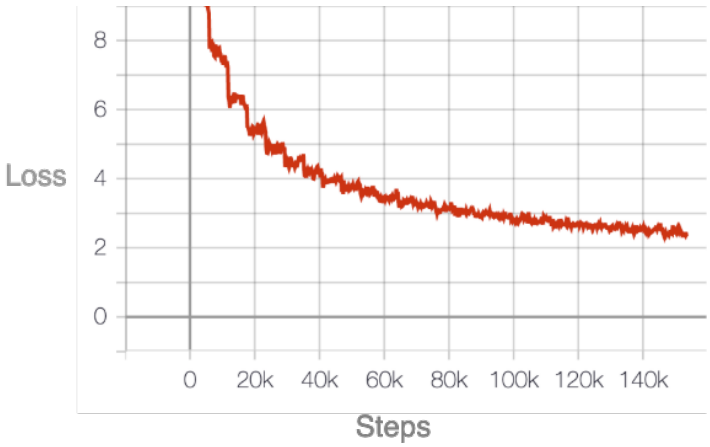


Figure 4.4: Evolution of Loss per step.

making the loss to stabilize, with small increases and decreases, for some steps, and falling suddenly.

Below, it is described the losses that compose the overall loss.

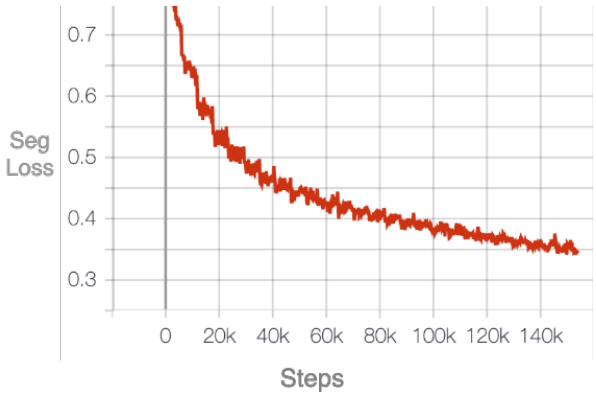
4.2.2 Quaternion Loss

Quaternion prediction is a simple regression problem and its approach is made with a l_1 norm,

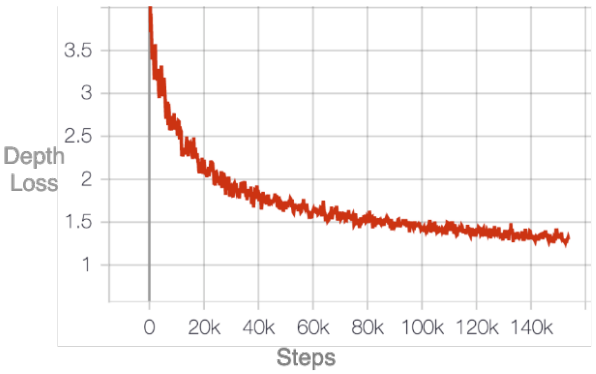
$$Loss_{quat} = \|(Q - Q^*)\|_1, \tag{4.2}$$

where Q is the quaternion predicted, Q^* is the quaternion ground-truth. The reason of the prediction being a quaternion, over being a rotation matrix 3×3, is because the quaternion guarantees that the rotation predicted belongs to the SO(3) group, adding to the fact that there are less network parameters that need to be predicted. Considering each prediction's number of parameters, it will be more efficient to predict four parameters than nine.

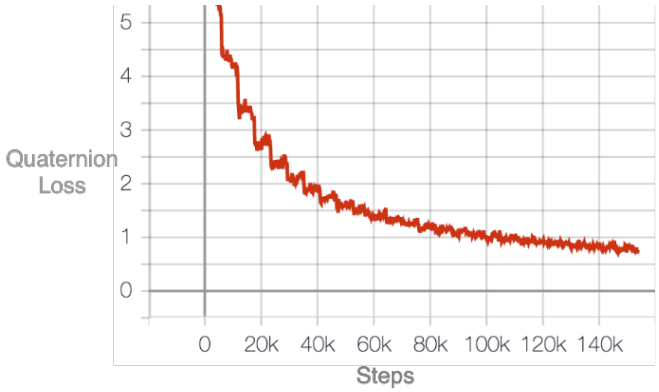
Figure 4.5 (c) represents the evolution of the quaternion loss over the steps. As expected, the loss is converging, and it has the same shape has the overall loss, i.e., the "stairs" effect due to the mini-batch learning.



(a) Segmentation Loss



(b) Offset/Depth Loss



(c) Quaternion Loss

Figure 4.5: The overall loss is composed by three losses. In figure 4.5 (a) it is represented the evolution of segmentation loss. Figures 4.5 (b) and 4.5 (c) present the offset/depth and quaternion losses, respectively. All of the three losses are converging, which is the objective.

4.2.3 Segmentation Loss

Image segmentation is a classification problem where each pixel has four probabilities associated. These four probabilities encode how certain the network thinks the pixel belongs to a specific plane. Classes 1, 2 and 3 correspond to planes that have their normals aligned to the axis [1,0,0], [0,1,0] and [0,0,1] from MW coordinate frame, and class 4 corresponds to a non-planar region.

For this purpose it was used a cross entropy loss,

$$Loss_{seg} = \frac{1}{K}\sum_{p=0}^{K} -\log\left(\frac{\exp\left(m_{class}^{(p)}\right)}{\sum_{j=0}^{C}\exp\left(m_{j}^{(p)}\right)}\right) \tag{4.3}$$

which has a softmax inside the logarithm operation. In equation 4.3 ($m_{class}^{(p)}$ is a probabilistic value predicted by the network for pixel p and class is the class target which the pixel belongs to. ($m_j^{(p)}$ is a probabilistic value predicted by the network for pixel p and class j, C is the number of classes, 4, and K is the number of pixels $K = 192 \times 256$.

The segmentation loss is represented in figure 4.5 (a), and is the loss which contribute with the lowest value to the overall loss. The loss is converging and it presents the same "effect" as the others.

4.2.4 Offset/depth map Loss

For the offset/depth map loss, we use a squared l2 norm,

$$Loss_{offset} = \frac{1}{K}\|(O - O^*)\|_2^2 \tag{4.4}$$

where O is the offset/depth map, O^* is the offset/depth map ground-truth, and $K = 192 \times 256$ pixels. On 4.5 (b), it is represented the offset/depth map loss when training.

From figure 4.5, we can conclude all losses are balanced, being the offset/depth loss the highest from them all at the end of the training. This is due to the complexity of it, since it is a regression problem to predict 192×256 pixel offsets/depths.

Chapter 5

Results

This chapter conducts experiments regarding the comparison of the proposed approach against the PlaneNet. It is explained the metrics applied and justified the comparisons made. As previously mentioned, MW-Net is able to reconstruct a planar scene with a single RGB image. It is a competitive method and is an innovative method for MW planes detection. To compare MW-Net against PlaneNet, it was applied the two recall metric from their paper. The comparisons were made with the original PlaneNet model, from their Github repository, which includes the dense conditional random field (DCRF), for segmentation refinement. MW-Net was trained for 40 epochs, but the model with the best results was on the 26^{th} epoch. For the comparisons made through this Chapter, we used the best model.

5.1 MW-Net outputs

This section describes and illustrates the output of the MW-net and compares it with the ground-truth. In figure 5.1, it is possible to see the MW-Net inputs and respective outputs. From left to right, the first image column presents the inputs images, followed by the segmentation predicted by the network and the segmentation ground-truth, on the second and third column, respectively. The fourth column consists of the offset/depth-map predicted, and in the last column, there are the respective ground-truth.

The output description can be consulted on section 3.3.1, where it was already referred that network predicts four probabilistic segmentation masks classes, three planar classes, one for each MW axis, and one non-planar class. To each pixel, it is assigned

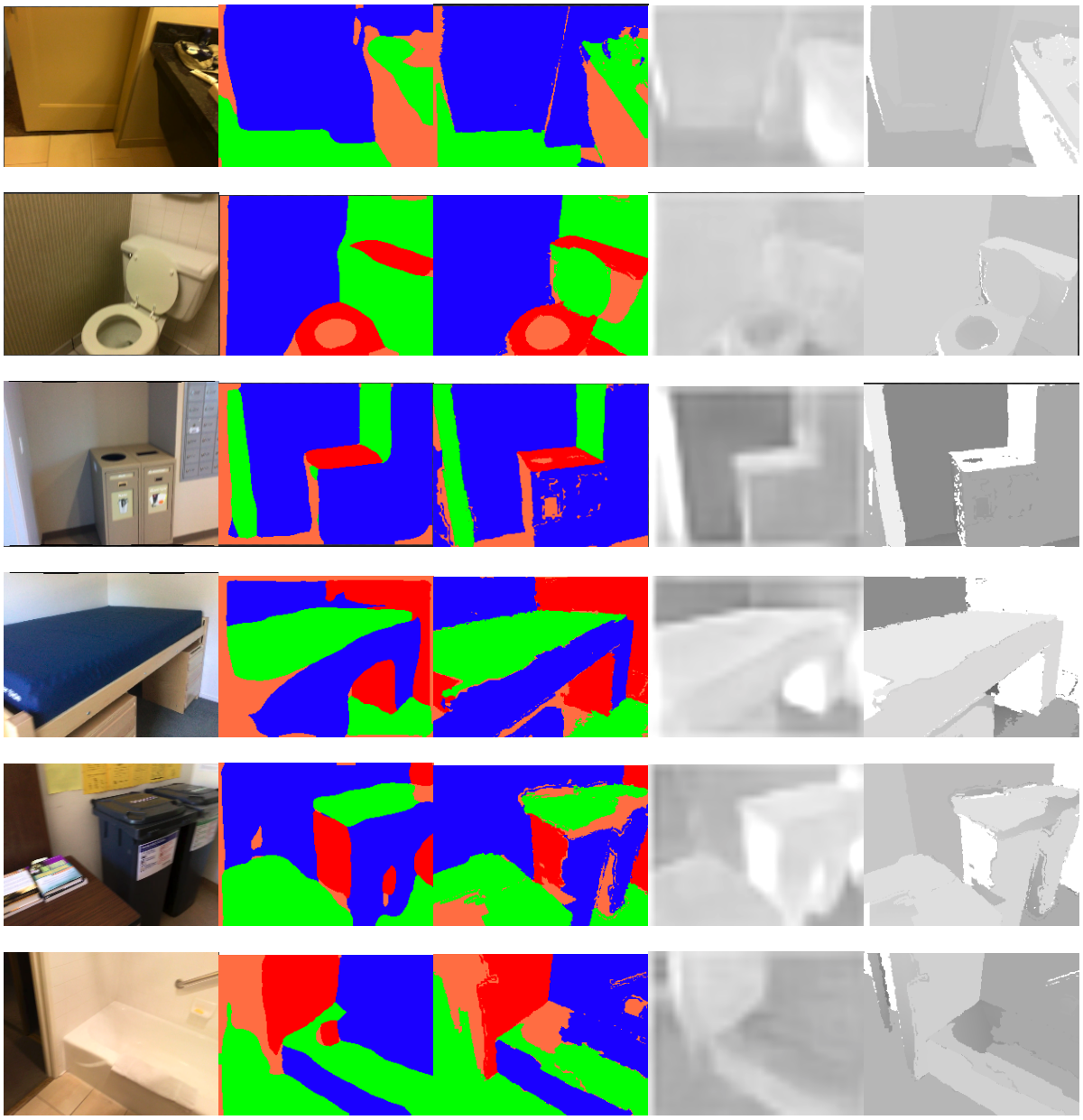
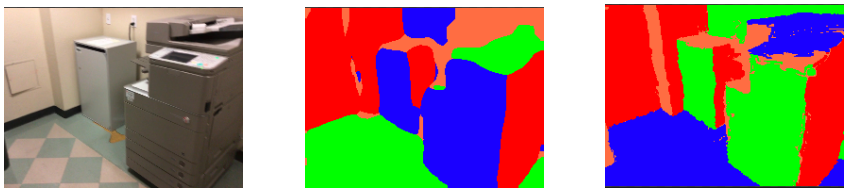


Figure 5.1: MW-Net outputs and respective ground-truths. The first images column, from left to right, is the inputs of the network, followed by segmentation predictions, second column, and segmentation ground-truths, third column. The last two columns, from left to right, are the offset/depth-map prediction and the offset/depth-map groundtruth. As it is possible to infer, the segmentation results are very close to the ground-truth.



(a) input (b) Segmentation (c) Segmentation-gt

Figure 5.2: Example of image segmentation when classes are swapped comparing to ground-truth. It is possible to see that in image segmentation prediction the classes represented by the color green and blue are swapped, relatively to the ground-truth.

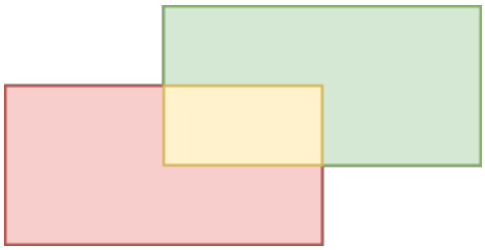
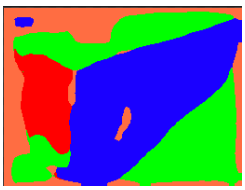


Figure 5.3: IOU example. Representation of two planes intersecting. The IOU between these two planes is the ratio between the number of pixels on the intersection of both planes (yellow region) over the number of pixels on the union (red + yellow + green)

the class with a higher value between the four classes. Looking at the segmentation on figure 5.1, it is possible to distinguish four colors, corresponding each to a class. The blue color correspond to the MW X-axis base vector, the green color to the MW Y-axis, and the red color to the MW Z-axis. The orange color corresponds to the non-planar class. The network presents incredible results on the segmentation branch, but there is margin to improve. It is possible to see, in many segmentation images, that there are some pixels on the left side that are classified incorrectly, being those pixels classified as non-planar. This can happen for many reasons, one of them may be due to the training dataset, that can have many data elements that are promoting over-fitting. This is one of the aspects that need to be improved in further developments. Nonetheless, MW-Net presents 80,75% of planar accuracy, while PlaneNet has 73,52%.

The metric applied for this accuracy is based on the IOU, see legend figure 5.3, between the ground-truth with the inferred plane. Taking in count the figure 5.3, the metric applied for each plane is the number of pixels that are in the intersection of the plane ground-truth with the plane inferred, number of pixels of yellow surface, over the


(a) input


(b) Segmentation


(c) Segmentation-gt

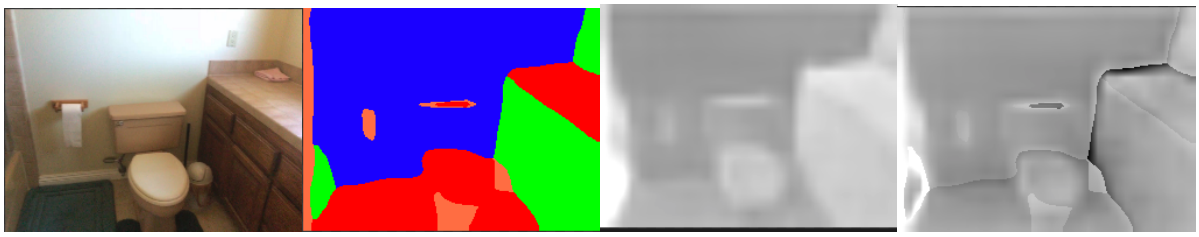
Figure 5.4: In this figure it is represented a bad labeled example. In figure 5.4 (a) it is represented the networks input, and on 5.4b (b) and 5.4 (c) it is represented the segmentation predicted by the network and the ground-truth, respectively. Cases like this decreases the network segmentation accuracy.

total number of pixels that belong to the plane ground-truth, represented by the number of pixel on yellow+green surfaces.

To be fair with the network's real performance, we have to pay attention to cases illustrated in figure 5.2, where the segmentation is well made but the classes are swapped. In figure 5.2 it is possible to verify the RGB image of a scene. In 5.2 (b), it is evident that the class represented by the blue color is swapped with the class represented by the green color, when comparing with the segmentation ground-truth in figure 5.2 (c). This image segmentation, although having classes swapped, it is segmenting planes correctly.

In order to the metric work as supposedly, to ground-truth planes we have to infer predictions that overlap the most with them. Obviously, two different ground-truth planes cannot have the same plane inferred. The association is made through the IOU, i.e., given a plane prediction it is computed the IOU with all the planes ground-truth, and this plane is associated to the ground-truth with which has the highest IOU. Although the high segmentation accuracy rates, the segmentation results may be harmed by some bad labelled data such as the one in figure 5.4. In the figure there are three images, figure 5.4 (a) is the network's input, and in figure 5.4 (b) and 5.4 (c) are the image segmentation from the network and ground-truth, respectively. It is obvious that the ground-truth is not accurate, and the segmentation by the network is under the expectation.

In figure 5.1, the fourth column and fifth column, counting from left to right, are the offset/depth map prediction and ground-truth, respectively. The offset/depth map is basically a offset mapping for planar surfaces and depth for non-planar surfaces, as it was explained. Being a regression problem, the network is responsible to predict 192×256 values, one for each pixel, resulting, naturally, in some outliers. In addiction



(a) Input (b) Segmentation (c) offset/depth (d) depth

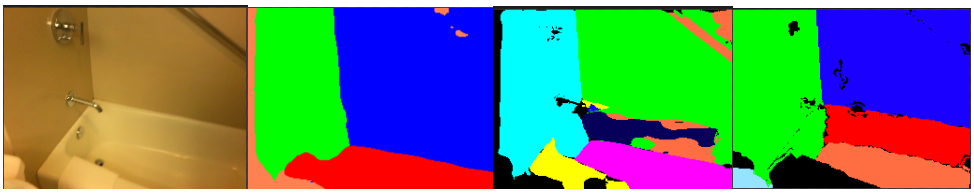
Figure 5.5: In this figure it is represented an example of depth computation. In figure 5.5 (a) is the RGB image of the scene, figure 5.5 (b) is the network's segmentation of the scene, and in figure 5.5 (c) is the offset/depth map. Using the network's outputs is possibble to build the deth image in figure 5.5 (d)

to this problem, for planar regions, since it is predicted an offset for each pixel, it is natural that a large number of pixels will have the same offset, and if this prediction it is not correct the error will increase since it is spread to the remaining pixels.

The overall depth is now easy to predict, knowing the network outputs. For that, we need to know the MW base vectors in reference to camera reference frame. Applying the rotation from MW to camera reference frame, inverse of the rotation predicted by the network, it is possible to obtain the desired base vectors. For instance, the pixels assigned with the first class, corresponding to the blue color, are associated to the MW X-axis base vector. Applying the rotation referred and following the steps on section 3.3.2 for every planar pixel, the proposed objective is achieved. In figure 5.5 it is possible to verify a depth image 5.5 (d) obtained using the outputs in 5.5 (b) and 5.5 (c). Concluding, it is possible to notice that innumerable indoor scenes presents many planar surfaces that are parallel and/or orthogonal between each other. Having a MW approach to the problem can simplify proposed objective, improving some results. In the images presented, the majority of planes were well predicted and it is possible to achieve remarkable results with it. In the next section it is made the comparison with the PlaneNet method.

5.2 MW-Net vs PlaneNet

Both MW-Net and PlanNet are deep methods that allow planes' reconstruction using a single network. In the following paragraphs it is referred the main differences between them.



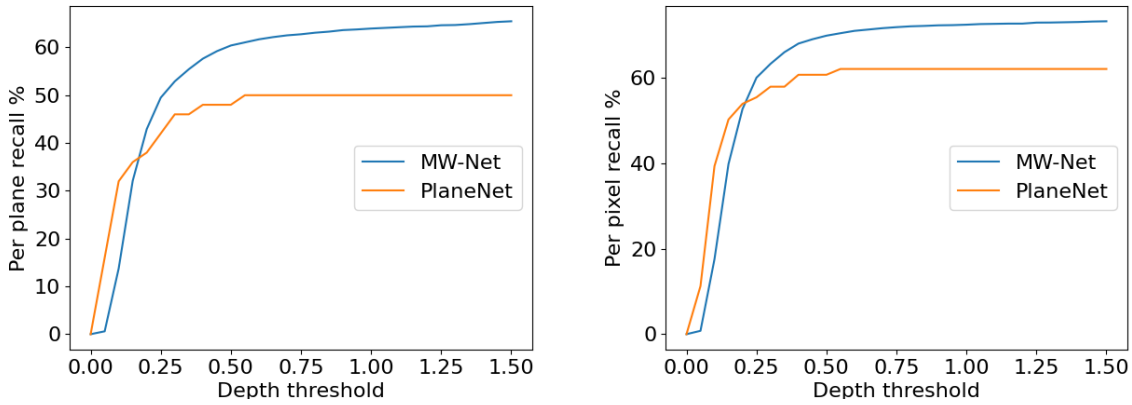
(a) Input (b) MW-Net (c) PlaneNet [36] (d) Groudtruth

Figure 5.6: In this figure it is possible to see the comparison of segmentation between MW-Net and PlaneNet, figures 5.6 (b) and 5.6 (c) respectively. It is possible to see that the segmentation does not miss any plane having the indoor scene totally identified.

PlaneNet outputs are the plane parameters, the image segmentation, and a non-planar depth-map. The plane parameters are the plane's normal and offset, in the form offset×normal, only needing three parameters to identify the planes (see equation 3.1). But since their parameters depend on the offset, it struggles to distinguish parallel planes with different offsets, when these planes are close from each other, harming their segmentation results. This can happen, for instance, due to network difficulties on distinguishing different textures. PlaneNet only predicts ten planes per scene, if there are a crowded planar scenes, it will fail to perform as expected. In its turn, MW-Net uses MW base vectors to identify the planes' normals, being the segmentation independent of the offset, and the offset is offered by the offset/depth map prediction. The network do not have a limitation on the number of planes that can be predicted.

In figure 5.6, it is possible to see the comparison between PlaneNet and MW-Net. The ground-truth represented in the figure 5.6 (d), segments the image in the same way PlaneNet does, where planes with different offsets are identified by different classes. In figures 5.6 (c) and 5.6 (d), the non-planar class is represented by the black color. The difficulty on identifying parallel planes with different offsets, when they are close from each other, is evident in the figure 5.6 (c), where it struggles to identify distinguish the planes represent by the colors red and blue on figure 5.6 (d), while MW-Net do not face this problem.

To compare against PlaneNet, it is applied two recall metrics to both methods, the same as in [36]. To understand the metrics, it is necessary to have a notion of what it is the Intersection Over Union IOU, see caption from figure 5.3. For the figure 5.7 (a), the metric presented is the percentage of the correctly predicted ground-truth planes. A



(a) Plane Recall Accuracy (b) Pixel Recall Accuracy

Figure 5.7: In this figure it is possible to compare segmentation evolution in function of Depth threshold using two recalll metrics. The comparison was made between MW-Net PlaneNet. MW-Net significantly outperforms PlaneNet in both metrics.

ground-truth plane is correctly predicted if the IOU with the inferred plane is over 0.5 and the mean offset/depth difference, from the overlapping region, is less than a given threshold. The offset/depth difference is the difference between the offset/depth of plane prediction pixels and the corresponding plane ground-truth pixels.

The second metric, figure 5.7 (b) is the number of pixels, that are in the overlapping regions, over the total number of pixels from all the planar surfaces in the scene, being similar to the metric presented in section 5.1. This measure it is not the same as the one in the previous section, because only the pixels that are in the planes well predicted by the metric in figure 5.7 (a), will count as pixels well predicted. This means that if the IOU between the planes ground-truth and the plane inferred is lower than 0.5, and they have pixels in common, this pixels will not count as well predicted. Although the denominator still is the total number of planar pixels in the scene.

In figure 5.7, it is presented the MW-Net and PlanNet performance when applies the metrics described previously. We vary the depth threshold from 0 to 1.50, and it is possible to verify that MW-Net significantly outperforms PlaneNet, except when the depth threshold is small and PlaneNet can fit planes accurately for those thresholds, lower than 0.2. It is seen that, considering threshold values above 0.2, the MW-net outperforms significantly, meaning that our image segmentation is much better than

PlaneNet, but PlaneNet outperforms MW-Net on depth prediction.

MW-Net obtains these results with less architecture complexity than PlaneNet. Although PlaneNet uses a single network for planar reconstruction, on segmentation branch, it uses a dense conditional random field (DCRF) (see [31], and train the DCRF module with precedent layers (see [64]), as a way to refine segmentation results. MW-Net outperforms PlaneNet without using any DCRF, as it is possible to verify in the comparisons made.

Chapter 6

Conclusion

This thesis presented a novel method for planar reconstruction using a MW approach. MW-Net receives an RGB image as input and outputs a rotation matrix from camera to MW coordinate frame as a quaternion, four image segmentation probabilistic masks and an offset/depth map. MW-Net predicts planes segments with high accuracy rates, and without any restriction on the number of MW planes that can predict. It was proven that MW approach is reliable since the innumerable quantity of planes that are parallel/orthogonal to each other, and almost all planar surfaces were detected. MW-Net outperforms PlaneNet, a state-of-the-art method, in terms of segmentation, achieving remarkable results. MW-Net not just outperform PlaneNet, also it does it with less network architecture complexity.

As future work, a comparison with PlaneRCNN and PlaneRecover should be made. There are space for improvements, such as on offset/depth map. On segmentation, there are miss-classified pixels, on most scene images left side, that need to improve.

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016. 23

[2] N. Ayache. *Vision stéréoscopique et perception multisensorielle: application à la robotique mobile*. 1989. 17

[3] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994. 1

[4] R. Brunelli. *Template matching techniques in computer vision: theory and practice*. John Wiley & Sons, 2009. 7

[5] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014. 9

[6] H. Cho, P. E. Rybski, A. Bar-Hillel, and W. Zhang. Real-time pedestrian detection with deformable part models. In *IEEE Intelligent Vehicles Symposium*, pages 1035–1042, 2012. 5

[7] J. M. Coughlan and A. L. Yuille. Manhattan world: Compass direction from a single image by bayesian inference. In *IEEE Int'l Conf. Computer Vision (ICCV)*, volume 2, pages 941–947, 1999. 10

[8] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 5828–5839, 2017. 12

[9] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893. IEEE, 2005. 1, 6

[10] E. Delage, H. Lee, and A. Y. Ng. Automatic single-image 3d reconstructions of indoor manhattan world scenes. In *Robotics Research*, pages 305–321. 2007. 10

[11] P. Denis, J. H. Elder, and F. J. Estrada. Efficient edge-based methods for estimating manhattan frames in urban imagery. In *European Conf. Computer Vision (ECCV)*, pages 197–210, 2008. 10

[12] P. Dollár, S. Belongie, and P. Perona. The fastest pedestrian detector in the west. 2010. 5, 6

[13] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2008. 1, 6

[14] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 9

[15] W. T. Freeman and M. Roth. Orientation histograms for hand gesture recognition. In *International workshop on automatic face and gesture recognition*, volume 12, pages 296–301, 1995. 6

[16] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Manhattan-world stereo. In *cvpr*, pages 1422–1429. IEEE, 2009. 1, 8, 10

[17] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Reconstructing building interiors from images. In *IEEE Int'l Conf. Computer Vision (ICCV)*, pages 80–87, 2009. 10

[18] D. Gallup, J.-M. Frahm, and M. Pollefeys. Piecewise planar and non-planar stereo for urban scene reconstruction. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1418–1425, 2010. 8

[19] R. Girshick. Fast r-cnn. In *IEEE Int'l Conf. Computer Vision (ICCV)*, pages 1440–1448, 2015. 1, 7

[20] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014. 1, 5, 6, 7

[21] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010. 1

[22] R. Guo, K. Peng, D. Zhou, and Y. Liu. Robust visual compass using hybrid features for indoor environments. *Electronics*, 8(2):220, 2019. 13

[23] R. I. Hartley. Self-calibration from multiple views with a rotating camera. In *European Conf. Computer Vision (ECCV)*, pages 471–478. Springer, 1994. 9

[24] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *IEEE Int'l Conf. Computer Vision (ICCV)*, pages 2961–2969, 2017. 1, 2, 8, 9

[25] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 1, 21

[26] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998. 1, 6

[27] B. Heisele, T. Serre, and T. Poggio. A component-based framework for face detection and identification. *Int'l J. Computer Vision (IJCV)*, 74(2):167–181, 2007. 6

[28] S. Ingle and M. Phute. Tesla autopilot: semi autonomous driving, an uptick for future autonomy. *International Research Journal of Engineering and Technology*, 3(9), 2016. 6

[29] N. Ketkar. Introduction to pytorch. In *Deep learning with python*, pages 195–208. Springer, 2017. 20

[30] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 24

[31] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. *Advances in neural information processing systems*, 24:109–117, 2011. 23, 36

[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 1, 6, 7

[33] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 6

[34] G. Lewis. Object detection for autonomous vehicles, 2014. 5

[35] C. Liu, K. Kim, J. Gu, Y. Furukawa, and J. Kautz. Planercnn: 3d plane detection and reconstruction from a single image. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 4450–4459, 2019. 1, 2, 5, 8, 9

[36] C. Liu, J. Yang, D. Ceylan, E. Yumer, and Y. Furukawa. Planenet: Piece-wise planar reconstruction from a single rgb image. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 2579–2588, 2018. 1, 2, 4, 5, 7, 9, 11, 34

[37] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European Conf. Computer Vision (ECCV)*, pages 21–37. Springer, 2016. xv, 1, 8

[38] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int'l J. Computer Vision (IJCV)*, 60(2):91–110, 2004. 6

[39] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, pages 922–928, 2015. 1

[40] R. K. McConnell. Method of and apparatus for pattern recognition, Jan. 28 1986. US Patent 4,567,610. 6

[41] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 4293–4302, 2016. 1

[42] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019. 20

[43] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017. 1

[44] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *cvpr*, pages 779–788, 2016. 1, 2, 5, 8

[45] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 7263–7271, 2017. 1, 2, 8

[46] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 8

[47] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99, 2015. 1, 2, 7, 8

[48] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *Int'l J. Computer Vision (IJCV)*, 115(3):211–252, 2015. 6

[49] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 519–528, 2006. 8

[50] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013. 6

[51] S. Shi, X. Wang, and H. Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 770–779, 2019. 1

[52] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *European Conf. Computer Vision (ECCV)*, pages 746–760, 2012. 1, 8, 9

[53] S. Sinha, D. Steedly, and R. Szeliski. Piecewise planar stereo for image-based rendering. 2009. 8

[54] N. Soans, E. Asali, Y. Hong, and P. Doshi. Sa-net: Robust state-action recognition for learning from observations. In *IEEE Int'l Conf. Robotics and Automation (ICRA)*, pages 2153–2159, 2020. 1

[55] R. Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010. 16

[56] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *Int'l J. Computer Vision (IJCV)*, 104(2):154–171, 2013. 7

[57] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages I–I. IEEE, 2001. 1

[58] F. Yang and Z. Zhou. Recovering 3d planes from a single image via convolutional neural networks. In *European Conf. Computer Vision (ECCV)*, pages 85–100, 2018. 1, 2, 9

[59] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. 21

[60] Z. Yu, J. Zheng, D. Lian, Z. Zhou, and S. Gao. Single-image piece-wise planar 3d reconstruction via associative embedding. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 1029–1037, 2019. 9, 10

[61] Yu, Fisher and Koltun, Vladlen and Funkhouser, Thomas. Dilated residual networks. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 472–480, 2017. 9

[62] C. Zhang, L. Wang, and R. Yang. Semantic segmentation of urban scenes using dense depth maps. In *European Conf. Computer Vision (ECCV)*, pages 708–721, 2010. 1

[63] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 2881–2890, 2017. 20

[64] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr. Conditional random fields as recurrent neural networks. In *IEEE Int'l Conf. Computer Vision (ICCV)*, pages 1529–1537, 2015. 36