José María Fácil Ledesma

Deep Learning for 3D Visual Perception

Director/es

Civera Sancho, Javier
Montesano del Campo, Luis

ISSN 2254-7606

ISSN 2254-7606

Tesis Doctoral

DEEP LEARNING FOR 3D VISUAL PERCEPTION

Autor

José María Fácil Ledesma

Director/es

Civera Sancho, Javier
Montesano del Campo, Luis

UNIVERSIDAD DE ZARAGOZA
Escuela de Doctorado

Programa de Doctorado en Ingeniería de Sistemas e Informática

2021

Deep Learning for 3D Visual Perception

José M. Fácil Ledesma

Advisors: Javier Civera & Luis Montesano

Departmento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza

This dissertation is submitted for the degree of
Doctor of Philosophy

October 2020

A mi hermana,
mis padres,
mis abuelas
y
Mª Luisa.

Acknowledgements

I started my thesis four years ago but I made the decisions that brought me here around a decade ago. Naturally, during all these years there has been many people influencing my life, supporting and helping me even if they or I did not notice. It is going to be impossible to mention them all; they simply are too many. I thank you all in advance. There are some major players I would like to mention, for whom I am deeply thankful and in debt.

First, I want to save a special mention to those people to who I owe such amazing opportunities and for whom I have great respect. To my advisors Javier and Luis for motivating and advising me through all these years. To Thomas Brox for letting me join his lab for a few months. To Alejo for always teaching and helping me since my first days; even if he did not have to. To Lina, for letting me be part of her team. To my high school teacher Ramon for opening this door for me.

Thanks to the external reviewers Alberto and Manuel and to the PhD committee Ruben, Jesus, Gabriel, Ana Cris and Taihú.

I would not like to forget all the PhD students and Professors at Unizar and specially those that I have directly work with. I would like to specially mention all my co-authors Alejo, Clara, Berta, Alejandro, Josechu, Jose Neira and my two super-advisors. There are some that I would like to have included in the previous list – I still want to – and that I want to thank for there discussions and opinions: Jose, Iñigo, Jesus, Seong and Carlos. My former lab mates Jason, Edu and the two Carlos and also my three outstanding Bachellor and Master students I have co-supervised: Dani, Mikel and Juan for who I hope the best in the next professional steps.

The day which comes to mind, while writing these lines, as one of my best days of work was with Jose, Clara and Alejandro. This was the best wine-ending conference deadline, for CVPR 2019. Exhausting experience, that finished around 5AM, but one of the best moments of team work I have ever experienced. If I have half as good teammates and friends as you in the future I will feel I have found the right place to be. For more Juepinchos with you guys.

Thanks to all my friends from Zaragoza. To all the Upside Downs with Lorenzo and Jose, any after-work routine with you two is unmatched. Thanks to Toño, Rafa, Alex, Dani, Slavi and Claudiu, you have definitely help to build the foundations of the professional I am today. To Laura and Dante, best duo. Also to all my great flatmates Santi, Andrea, Alessandra & Andrea, Patri, Diego and Irene.

I would be remiss to forget all my friends in Huesca. I am very happy to say I continue having all my good friends from this small town and how proud I am of still being part of your group: Acher, Lorenzo, Miguel, Lampi, Dani, Garasa, Nacho, Gavin, Olivan, Josan, Elias, Cristian, Ruben and Olmo. Also thanks to the best *cordada Carnicraba* Arturo, Toño and Edu for teaching me how to fight my fears. The experiences I have had with you these last years on the walls are incredible and still hard to believe I accomplished them.

I had a very special time in Freiburg and I do not want to miss a big thanks to all my friends there. Thanks Özgün, Tonmoy, Huizong and Benjaming. Thanks to my flatmates Kristen and Sebas. Thanks to Abel and Paula for admitting me in that great small group, it meant a lot.

Summer 2019 was a blast in Zurich. Thanks to all the great people that happened to be there during that time. For all the *culebreo* to Alber, Ruben and Alejo, so fun. For all the Toro bars to Mariano, Clara and whoever the company was that day. Thanks to all the people that I worked with me and taught me, to all the Facebook Zurich team.

I must not forget my San Francisco buddy Rigley, you and Rich really have been the most welcoming friends ever. Thanks to my homie Srinath, for your lessons and conversations during improvised exotic dinners are unforgettable. To the Apple team I am so lucky to be part of. I will see you all soon.

I would not forgive myself if I don't save a special mention to these very important people. Thanks to to my best bro Jake, no day I am with you I don't learn a new lesson; your never-ending will to improve and change is inspirational. To Maggie, that has become my best adventure companion, your unstoppable motivation and curiosity are exemplar. Your ability to group and befriend are lovable, and you always bring the best with you. Thanks for your patience, love and support, specially these last months. You always make the right question to solve the puzzle.

To conclude I would like to thank my family, to all members of it but specially my parents and my sister Irene for their love, lessons and support during all these years. Irene it is an honor to have you as sister, not a single person does not have something good to say about you and that is great to hear when you are in a mistrust moment. *Mamá, las conversaciones contigo siempre me han enseñado, a menudo pones mis pensamientos en su debido lugar, guardo estos momentos como un tesoro. Papá, eres mi gran maestro, al que siempre admiraré y del que siempre querré saber la opinión. Sentir tu apoyo y orgullo ha sido un combustible inagotable. Estoy muy orgulloso y feliz de contar con todos vosotros. Gracias.*

P.S. Pablo, thanks for all the book recommendations and video calls, you have done COVID-19 lockdown much more entertaining, *I think*. I can't wait to read a book produced by you man.

"Every scholar knew that one of the greatest dangers in research was the desire to find a specific answer." – Sazed

— Brandon Sanderson, Mistborn: The Well of Ascension (2007)

Abstract

3D Visual Perception refers to the set of problems that involve gathering information trough a visual sensor and estimating the three-dimensional position and structure of the objects and formations surrounding the sensor. Functionalities like ego-motion estimation or map building are essential to enable higher-level tasks such as autonomous driving or augmented reality. In this thesis we have addressed several 3D perception challenges, all of them meant to be useful from the perspective of Simultaneous Localization and Mapping (usually refereed with its acronym SLAM), which is itself a 3D Perception problem.

Simultaneous Localization And Mapping –SLAM– aims to track the pose of a device (for example a robot, a phone or augmented-reality glasses) with respect to a map that is simultaneously built while the sensorial platform is exploring the environment. SLAM is a relevant technology in several applications, for instance, Virtual Reality (VR), Augmented Reality (AR) or Autonomous Driving. Visual SLAM is the term used when only visual sensors (cameras) are used. Many of the pieces in an ideal SLAM system are nowadays well-known, mature and some are even present in applications. However, for other pieces there are still significant research challenges. In particular, the ones that we address in this thesis are: the estimation of the 3D structure around the camera from just one single image, place recognition under drastic appearance changes or semantic reconstruction; all using deep neural networks.

Monocular depth estimation is the task of perceiving the distance to the camera from every pixel in an image, just from the information on that single image. This is a ill-posed problem, therefore it is very difficult to infer the exact depth distance of points just from one single image. It requires knowledge of what we see and the sensor we use to see. For example if we know that a certain car model has certain height and we know the internal parameters of the camera we used; we can estimate how far the car is if its height is 50 pixels in the image. For this we present the first approach capable of estimating depth from one single view that is able to have a reasonable performance with pictures taken by different cameras, such as your

phone's but also your wearable camera.

We also present how to estimate, from one single image, the structure of indoor rooms or room layout. For this second approach we leverage from spherical panoramic images in equirectangular representation. Equirectangular image is an image format that displays all directions in an spherical image. The equirectangular format is one single stitched image of 360° horizontally and 180° vertically. From these images we aim to recover the room layout, our goal being to learn the image cues that define the main structure of an indoor room. We focus on recovering the simplest layout model, which are the lines that delimit the floor, ceiling and walls.

Long-term localization and mapping require modeling appearance changes in the environment, as the way a particular place looks in winter or summer, for example, can change greatly. We introduce a multi-view condition-invariant place recognition model that is robust to this cases. Visual Place Recognition aims to identify a place that we have visited already by associating the visual clues in image taken now and those taken before at different times. It is preferred to be invariant to view-point, light changes, dynamic objects moving and long-term appearance changes such as seasons, day and night or weather conditions.

Finally, and also related to long-term functionality, we also developed DynaSLAM, a SLAM system able to distinguish between dynamic and static objects in a scene and that tracks the position of the device with respect to the static parts only. It is very useful to not include the dynamic objects in the map that simultaneously we are building so that if later we visit the same scene and the dynamic objects have moved we can still function correctly in the long term.

Resumen

La percepción visual 3D se refiere al conjunto de problemas que engloban la reunión de información a través de un sensor visual y la estimación la posición tridimensional y estructura de los objetos y formaciones al rededor del sensor. Algunas funcionalidades como la estimación de la ego moción o construcción de mapas are esenciales para otras tareas de más alto nivel como conducción autónoma o realidad aumentada. En esta tesis se han atacado varios desafíos en la percepción 3D, todos ellos útiles desde la perspectiva de SLAM (Localización y Mapeo Simultáneos) que en si es un problema de percepción 3D.

Localización y Mapeo Simultáneos –SLAM– busca realizar el seguimiento de la posición de un dispositivo (por ejemplo de un robot, un teléfono o unas gafas de realidad virtual) con respecto al mapa que está construyendo simultáneamente mientras la plataforma explora el entorno. SLAM es una tecnología muy relevante en distintas aplicaciones como realidad virtual, realidad aumentada o conducción autónoma. SLAM Visual es el termino utilizado para referirse al problema de SLAM resuelto utilizando unicamente sensores visuales. Muchas de las piezas del sistema ideal de SLAM son, hoy en día, bien conocidas, maduras y en muchos casos presentes en aplicaciones. Sin embargo, hay otras piezas que todavía presentan desafíos de investigación significantes. En particular, en los que hemos trabajado en esta tesis son la estimación de la estructura 3D al rededor de una cámara a partir de una sola imagen, reconocimiento de lugares ya visitados bajo cambios de apariencia drásticos, reconstrucción de alto nivel o SLAM en entornos dinámicos; todos ellos utilizando redes neuronales profundas.

Estimación de profundidad monocular is la tarea de percibir la distancia a la cámara de cada uno de los pixeles en la imagen, utilizando solo la información que obtenemos de una única imagen. Este es un problema mal condicionado, y por lo tanto es muy difícil de inferir la profundidad exacta de los puntos en una sola imagen. Requiere conocimiento de lo que se ve y del sensor que utilizamos. Por ejemplo, si podemos saber que un modelo de coche tiene cierta altura y también sabemos el tipo de cámara que hemos utilizado (distancia focal, tamaño de pixel...); podemos decir que si ese coche tiene cierta altura en la imagen,

por ejemplo 50 pixeles, esta a cierta distancia de la cámara. Para ello nosotros presentamos el primer trabajo capaz de estimar profundidad a partir de una sola vista que es capaz de obtener un funcionamiento razonable con múltiples tipos de cámara; como un teléfono o una cámara de video.

También presentamos como estimar, utilizando una sola imagen, la estructura de una habitación o el plan de la habitación. Para este segundo trabajo, aprovechamos imágenes esféricas tomadas por una cámara panorámica utilizando una representación equirectangular. Utilizando estas imágenes recuperamos el plan de la habitación, nuestro objetivo es reconocer las pistas en la imagen que definen la estructura de una habitación. Nos centramos en recuperar la versión más simple, que son las lineas que separan suelo, paredes y techo.

Localización y mapeo a largo plazo requiere dar solución a los cambios de apariencia en el entorno; el efecto que puede tener en una imagen tomarla en invierno o verano puede ser muy grande. Introducimos un modelo *multivista* invariante a cambios de apariencia que resuelve el problema de reconocimiento de lugares de forma robusta. El reconocimiento de lugares visual trata de identificar un lugar que ya hemos visitado asociando pistas visuales que se ven en las imágenes; la tomada en el pasado y la tomada en el presente. Lo preferible es ser invariante a cambios en punto de vista, iluminación, objetos dinámicos y cambios de apariencia a largo plazo como el día y la noche, las estaciones o el clima.

Para tener funcionalidad a largo plazo también presentamos DynaSLAM, un sistema de SLAM que distingue las partes estáticas y dinámicas de la escena. Se asegura de estimar su posición unicamente basándose en las partes estáticas y solo reconstruye el mapa de las partes estáticas. De forma que si visitamos una escena de nuevo, nuestro mapa no se ve afectado por la presencia de nuevos objetos dinámicos o la desaparición de los anteriores.

En resumen, en esta tesis contribuimos a diferentes problemas de percepción 3D; todos ellos resuelven problemas del SLAM Visual.

Table of contents

Chapter 1

Introduction

Computer vision has experienced a huge growth during the last decades. It can be broadly defined as extracting information from images with similar goals to humans, making computers see as we do. In order to meet this ambitious goal, computer vision addresses a vast number of different perception problems like 3D reconstruction, object detection or object tracking. And it does that by using many different models, strategies, algorithms and techniques, such as epipolar geometry, probabilistic models or machine learning to name a few examples. Computer vision is every time more present in our lives and we have grown to naturalize it, benefiting from its advantages and getting used to its ubiquitous presence in many modern technologies like face recognition in phones, lane detection in cars or camera tracking in virtual and augmented reality (VR/AR) headsets.

1.1 3D Visual Perception and Visual SLAM

3D visual perception consists on recovering information of the 3D structure behind the images taken by cameras. There is a wide range of research problems grouped under the general term perception, that goes from recognizing the objects around the camera to estimating the ego-motion of the camera itself. Sometimes we may dispose of multiple views of a scene taken at the same time (e.g. stereo cameras) or taken sequentially at different times; and sometimes we will perceive our environment from just one single image.

Many different applications may benefit from 3D perception. Some include, cars with pedestrian detection as extra safety feature, medical image for precise surgery or people tracking in video vigilance among many others. In the last years, many of these perception solutions have been transferred to commercial products and they are every time more present. However, there are still many points to improve both in resolution, like complete scene understanding, and precision, like object classification or dense depth estimation. Some of these problems have not yet been addressed successfully or still have a poor performance.

There are two particular perception problems that humans resolve very well, which are the estimation of our trajectory – tracking our motion – and the 3D perception of our environment and elements around us – mapping the scene. In the robotics community this problem is usually referred to as SLAM (which stands for Simultaneous Localization And Mapping). Another acronym is used by the computer vision community for a similar problem, SfM (which stands for Structure from Motion). The difference lies in that SLAM, differently to SfM, assumes that the data will appear sequentially and aims to complete the computation within the sampling time of the sensor(s) (real time). SLAM is a very relevant problem that is essential for a wide range of applications such as virtual, augmented or mixed reality (Newcombe et al., 2011, Klein and Murray, 2007), Micro Aerial Vehicles (MAVs) navigation (Shen et al., 2011), autonomous driving (McManus et al., 2013) and assisted surgery (Lamarca et al., 2020). Visual SLAM addresses the localization and mapping tasks using only visual sensors, *e.g.,* RGB cameras.

Visual SLAM is currently a mature problem, but still a relevant and challenging research area (Cadena et al., 2016). Literature has proposed many different approaches to SLAM, the two main families being the feature-based models – using salient points and matching them across views – (Klein and Murray, 2007, Mur-Artal et al., 2015) and direct approaches – that rely directly on the pixel colors – (Newcombe et al., 2011, Engel et al., 2014). Although there are excellent approaches to the general SLAM problem from visual clues, the problem is not fully solved yet. There are many open challenges that can actually appear very often: surfaces with no texture and therefore difficult to match between different views (Concha and Civera, 2015a), small camera motions that do not create enough parallax to triangulate points, revisiting places after their appearances have changed (*e.g.* day and night) (Gomez-Ojeda et al., 2015), dynamic elements of the scene (e.g., people walking on the street) that violate the usual rigidity constraints of the algorithm (Alcantarilla et al., 2012) or adding semantic information to certain parts of the scene (*e.g.* road lanes or room layout) (Fernandez-Labrador et al., 2018b).

As the technology for SLAM has become more known and present in applications like VR/AR, its challenges have become more evident. Despite tackling the general problem with great performance, large errors due to cases like day/night or small motions limit the potential of the technology in critical applications, for example autonomous driving. In order to have a reliable perception system, these cases must be addressed as well. Concurrently with the growth in use of SLAM systems, the technology for recording and storing data has become better and more accessible. As result of this, more data with better ground truth is available. This has allowed the community to create better benchmarks and evaluation tools.

The large amount of data available, in this as in many other problems, has made machine learning algorithms gain relevance. A machine learning algorithm is an algorithm that learns from data (Goodfellow et al., 2016). Specifically, it learns to improve its performance (measured by a certain metric) on a task, based on experience. An example of a task can be *object classification* where previous experience are *images* that have been *labeled* (or not) and the measurement of performance might be the classification *accuracy*.

The machine learning technique that has benefited more from having a vast amount of data, as well as more computational power, is deep learning. Deep learning is a family of algorithms that forms part of the machine learning and artificial intelligence field. Deep learning is based on artificial neural networks, being deep feed-forward networks or multi-layer perceptron the quintessential model for deep learning (Goodfellow et al., 2016). Deep neural networks (DNNs) are a set of interconnected processing units able to approximate complex functions based on a performance measurement normally referred to as loss.

It is in this context of deep learning growing in popularity and being used to tackle perception problems that this thesis takes place. In particular, we have contributed to several challenges in different 3D perception problems: dense monocular depth estimation, place recognition under drastic appearance changes, semantic reconstruction and ego-motion estimation in dynamic environments. All of them are connected to visual SLAM to a greater or lesser extent.

1.2 How Deep Learning is Improving Visual Perception

The increase of data and computational power has made DNNs very popular in many different applications. Some of the most popular examples are found in the natural language processing (NLP) domain where we can differentiate several tasks: speech recognition (Amodei et al., 2016, Wang et al., 2019), sentiment analysis (Zhang et al., 2018) or hate speech detection (Gomez et al., 2020) among many others.

Deep neural networks have also been widely used in computer vision for many different tasks, using mainly the convolutional layers introduced in LeCun et al. (1989). Convolutional neural networks (CNNs) are a specialized type of neural network for processing data that has a fixed-size grid-like topology. They are only locally connected, sharing the weight on each local connection that is normally referred as kernel and allow the same pattern to be learned in any part of the grid. Other network architectures were later used with a CNN backbone, for example Generative Adversarial Networks (GANs) introduced by Goodfellow et al. (2014).

GANs are a proposal for training generative models that consist in two networks competing in a game where the success of one implies the other one's loss. These models have been used in many applications with different purposes, such as training context encoders as unsupervised pretraining for object classification and semantic segmentation (Pathak et al., 2016). In parallel, Ronneberger et al. (2015) introduced the U-Net architecture, adding extra connections at different levels between encoder and decoder, consequently better propagating high-level high-resolution features. Another relevant architectural design are the Siamese networks, used for tasks such as image retrieval (Gordo et al., 2016). A Siamese neural network consists of two DNNs that share their weights during training, while being fed with two different input images to compute comparable output vectors.

When working with data that has a temporal dimension, like language or video (Donahue et al., 2014), recurrent neural networks (RNN) have demonstrated a good performance. Differently from a normal DNN, a RNN defines a directed graph with temporal connections along a sequence of inputs (Hochreiter and Schmidhuber, 1997). These connections allow the network to capture temporal patterns from sequential data, and they have been used in tasks like speech recognition, video captioning and handwriting recognition among others.

Visual perception has greatly benefited from the use of deep learning models. Some popular problems that have made use of deep networks are object classification (Simonyan and Zisserman, 2014, He et al., 2016), object detection (Liu et al., 2020), image and video semantic segmentation (Garcia-Garcia et al., 2018), and face recognition (Parkhi et al., 2015). More aligned with the purpose of this thesis, in 3D visual perception many tasks have also been addressed using deep learning, starting from pure monocular depth estimation (Eigen et al., 2014), camera pose estimation (Kendall et al., 2015), object pose tracking (Tompson et al., 2014) and flow estimation (Dosovitskiy et al., 2015). Geometric models began to be added to the network architectures and losses, for self-supervision (Godard et al., 2017) or to be learned (Ummenhofer et al., 2017).

In this thesis we have identified several challenges that needed to be addressed inside 3D perception. We have proposed solutions and contributed to different areas, one of them being monocular depth estimation. Along this manuscript we will present works that discuss some of these architectural models mentioned before. For instance, RNN have been used for generating distinctive descriptors for sequence of images to perform visual place recognition. Siamese networks have been used to compare these descriptors for several images and also to train single-view depth estimation with images of different sizes. In the next section we briefly explain each one of these problems and our contributions.

1.3 Our contributions in 3D Visual Perception

"I suppose it is tempting,
if all you have is a hammer, everything looks like a nail"

— Abraham Maslow, The Psychology of Science (1966)

We have tackled the following significant challenges on 3D visual perception: dense monocular depth estimation, place recognition under drastic appearance changes, semantic reconstruction and ego-motion estimation in dynamic environments. We contribute in all of these challenges using deep learning. We make use of convolutional networks to work with images and address these challenges by extracting and processing visual clues present in the input images. We have also contributed to make deep learning and specifically CNNs more adaptable to different cameras or image representations by introducing CAM-Convs and EquiConvs, two types of convolutions that will be further detailed in Chapters 2 and 5. In the following sections we introduce the research problems, the related literature and our contributions for all the different 3D visual perception challenges we have addressed.

1.3.1 Visual Mapping without Motion

Visual SLAM is mainly based on matching points across different views and jointly estimate the relative movement of the camera and the triangulation of these points (Triggs et al., 1999). However, that procedure's accuracy depends on the parallax angle generated by the camera motion. When the camera does not move, the trajectory does not need to be calculated but the estimation of the 3D of the scene becomes challenging as it is an ill-posed problem. This problem is referred to in the computer vision community as single-view depth estimation or monocular depth estimation. Depth perception from one single image has usually been addressed using machine learning algorithms (Saxena et al., 2009).

One of the most impactful advances in single-view depth estimation was the use of deep learning by Eigen et al. (2014), that trained in a supervised manner a deep neural network to predict depth from single RGB images. Their proposal is a convolutional neural network (CNN) trained with RGB-D examples to predict a depth value for each RGB pixel.

Following works farther improved predictions by using deeper models and networks pretrained in other tasks, mainly classification (Eigen and Fergus, 2015, Laina et al., 2016). Different training procedures allowed to train with different datasets despite not being supervised – meaning that they do not have depth values for all the pixels. Godard et al. (2017) proposed a self-supervised scheme by forcing left-right stereo camera consistency during training, their original network architecture based on Mayer et al. (2016) DispNet. Zhou et al. (2017) took the self-supervision a bit further by not using stereo but monocular ego motion. In this case the ego motion is also predicted by the network, therefore requires much less supervision for training. A combination of both ideas was also proposed to learn depth and ego motion jointly (Zhan et al., 2018).

Despite all the advances in single-view depth estimation there is one attribute of traditional geometry-based multi-view model that has been largely unaddressed: incorporating the camera intrinsic parameters into the models. There has been two ways of dealing with the camera geometry in deep-learning single-view depth. The first and most generalized way has been to train on a single dataset where all the images were captured using the same camera; the common practice being to ignore the camera geometry and simply assuming that all the cameras have the same one (Eigen and Fergus, 2015, Laina et al., 2016). The second approach has been to use one single dataset (*e.g.* MegaDepth) with images from different types of cameras but still ignoring the camera geometry in the model (Li and Snavely, 2018). Unfortunately, this forces the predicted depth to be up-to-scale and ignores the projection model.

Chapter 2 in this thesis focuses on providing deep neural networks with awareness of the camera geometry when estimating depth from single images. Inspired by Liu et al. (2018) we propose CAM-Convs a type of convolution that adds pixel-wise information of the camera intrinsic parameters and allows the network to train and infer depth for different cameras. In this data-hungry era our proposal adds the possibility of training with multiple cameras without sacrificing performance. Also, our depth estimation approach has proved to generalize across different cameras never seen during training. Figure 1.1 shows an example of depth predictions made by our deep neural network with and without using our CAM-Convs. Previous approaches normally train and evaluate on the same type of data both in domain, for example indoor images, and in camera, training and testing on the same camera. We have proved that it is possible for a CNN to be aware of the camera and to build a model capable of working with different types of cameras. Continuing our proposal, López-Antequera et al. (2020) introduces a camera normalization that also allows CNN to train with multiple cameras. Preliminary unpublished results of our work were presented to the Robust Vision Challenge in 2018 in the International Conference in Computer Vision and Pattern Recognition (CVPR) and achieved the third position in the ranking, see Figure 1.2.

1.3.2 Visual Mapping with Little Motion

While estimating depth from a single view (hence no motion) is the most challenging case, visual SLAM still suffers when there is very little camera motion. As mentioned before, the

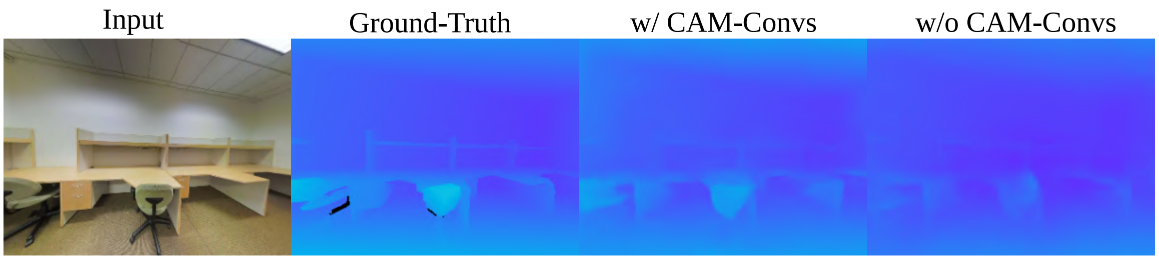
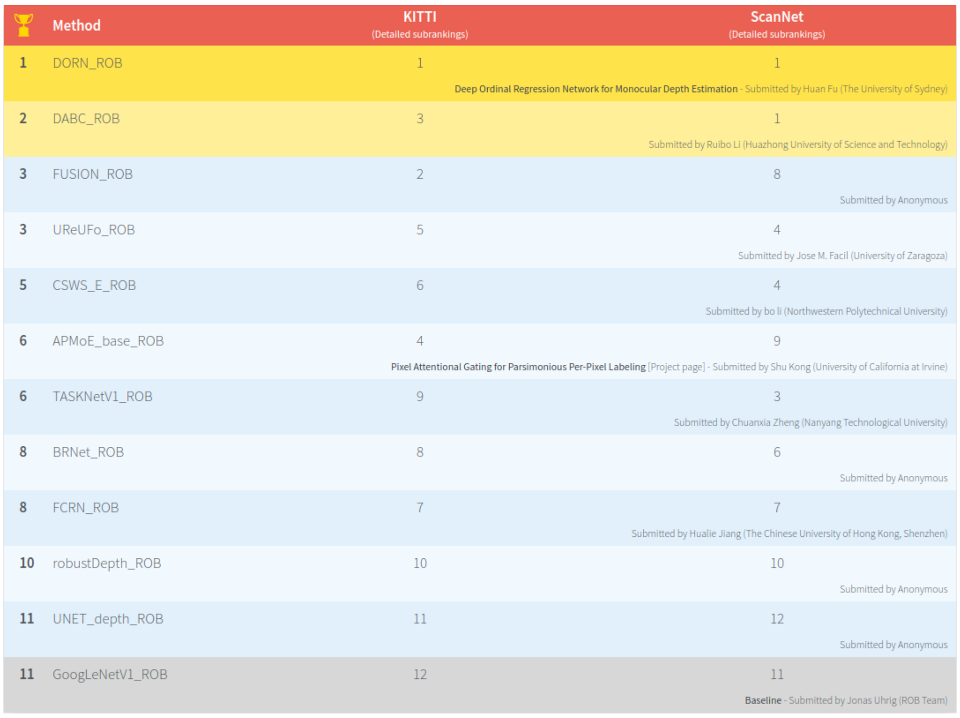


Fig. 1.1 Depth prediction from single image. 1st column shows the original RGB image and input for the network. 2nd column shows the ground-truth depth. 3rd column shows the prediction by a CNN using CAM-Convs. 4th column shows the prediction of the same CNN but this time not using CAM-Convs.



🏆	Method	KITTI (Detailed subrankings)	ScanNet (Detailed subrankings)	
1	DORN_ROB	1	1	**Deep Ordinal Regression Network for Monocular Depth Estimation** - Submitted by Huan Fu (The University of Sydney)
2	DABC_ROB	3	1	Submitted by Ruibo Li (Huazhong University of Science and Technology)
3	FUSION_ROB	2	8	Submitted by Anonymous
3	UReUFo_ROB	5	4	Submitted by Jose M. Facil (University of Zaragoza)
5	CSWS_E_ROB	6	4	Submitted by bo li (Northwestern Polytechnical University)
6	APMoE_base_ROB	4	9	**Pixel Attentional Gating for Parsimonious Per-Pixel Labeling** [Project page] - Submitted by Shu Kong (University of California at Irvine)
6	TASKNetV1_ROB	9	3	Submitted by Chuanxia Zheng (Nanyang Technological University)
8	BRNet_ROB	8	6	Submitted by Anonymous
8	FCRN_ROB	7	7	Submitted by Hualie Jiang (The Chinese University of Hong Kong, Shenzhen)
10	robustDepth_ROB	10	10	Submitted by Anonymous
11	UNET_depth_ROB	11	12	Submitted by Anonymous
11	GoogLeNetV1_ROB	12	11	**Baseline** - Submitted by Jonas Uhrig (ROB Team)

Fig. 1.2 Image taken from `http://www.robustvision.net/rvc2018.php`. The Robust Vision Challenge 2018 was a full day event held in conjunction with CVPR 2018 in Salt Lake City. The goal of this challenge is to foster the development of vision systems that are robust and consequently perform well on a variety of datasets with different characteristics.

parallax angle is very small in this case and this makes the estimation inaccurate. One of the hypotheses in this thesis is that multi-view algorithms can benefit from single-image depth estimation. Mapping from multiple views can leverage from having a first guess from a single view. Despite being a ill-posed problem, the literature has shown (Eigen et al., 2014, Eigen and Fergus, 2015, Ummenhofer et al., 2017) that it is possible to learn visual clues from which we can successfully predict a reasonably accurate 3D structure. Chronologically, the first work of this thesis, presented in Chapter 3, successfully combines both mapping techniques; single-view deep-learning-based depth estimation (Eigen and Fergus, 2015) and multi-view direct-based dense depth estimation (Newcombe et al., 2011). Our proposal is based on the fusion of both estimations, and our results show that neither mapping technique gets affected by the other and both can be complementary and achieve better results when they are combined. Figure 1.3 shows an example that summarizes the results of this work.

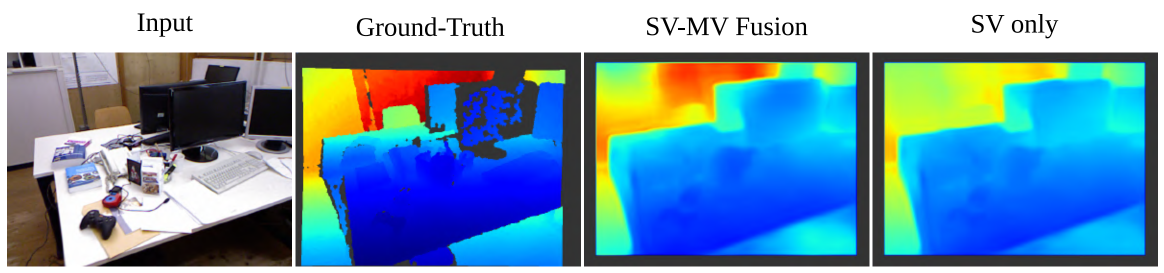


Fig. 1.3 Depth prediction combining single-view predicted with a CNN and geometry-based multi-view depth. 1st column shows the original RGB image and input for the network. 2nd column shows the ground-truth depth. 3rd column shows the single-view (SV) prediction by a CNN and then fused with a semi-dense depth predicted by traditional multi-view (MV) depth estimation. 4th column shows the single-view (SV-only) prediction of the same CNN but this time not adding extra information.

Following works have shown similar conclusions, combining learning and geometry based multi-view estimation. Tateno et al. (2017), similarly to us, directly combines the raw results of deep neural network (Laina et al., 2016) and a semi-dense visual SLAM system (Engel et al., 2014). Ummenhofer et al. (2017) developed a learning-based model that leverages from stereo views. They propose a CNN that predicts the motion between two views as well as the a dense depth estimation of one of them. Bloesch et al. (2018) introduced a way to optimize learning-based single-view predictions by minimizing a photo metric energy term that depends on a sequence of images. They train a Variational Auto-Encoder (VAE) with single images and, in test time, optimize the latent space generated by the VAE to produce the best depth representation that minimizes the photo-metric error. Zhou et al. (2018) presented a totally learning based multi-view tracking and mapping, in which they sequentially feed their model with images and the Convolutional Neural Network interactively minimizes the error.

A similar line of work is depth in-painting or depth completion, where deep learning is used to fill in a dense depth map starting from a sparse depth map and an image. It has been used to complete depth images captured by a RGB-D sensor (Zhang and Funkhouser, 2018) or Lidar (Qiu et al., 2019, Ma et al., 2019) but can potentially be used to fill semi-dense (Engel et al., 2014) or sparse reconstructions (Mur-Artal et al., 2015) for visual SLAM.

1.3.3 Place Recognition under Appearance Changes

When we revisit a place that we have visited before, we can use such knowledge in a wide array of tasks. For example, being able to be back on track after being lost or disoriented, a familiar room or street can be of so much help. Another application of place recognition is navigation and decision making. If we have map memorized we can decide the best way to reach our goal in the fastest way, *e.g.* the kitchen when we are seeking food. In visual SLAM, place recognition plays an important role and is essential to reuse and/or update previously mapped places, also called *loop closure* in robotics, and to re-localize the camera when the tracking has failed, *re-localization* (Gálvez-López and Tardos, 2012, Lee and Civera, 2019). Place recognition has other applications such as image retrieval (Noh et al., 2017), autonomous driving (McManus et al., 2014) or augmented reality (AR) (Middelberg et al., 2014).

One of the problems and main challenges in visual place recognition (Lowry et al., 2016) is that the scenes rarely stay intact over time; meaning that they suffer changes and they do not look exactly as they were before (Garg et al., 2019). Examples of changes can go from dynamic objects like people, that appear or disappear and produce confusing visual cues, to movable objects that have been moved like furniture or vehicles. Another example include time, weather or seasons: light changes produce very different visual effects during day and night, rainy days produce reflections and a winter day can lead to images that are very different from the same scene during the summer. Examples of visual changes in the appearance of a scene can be seen in the Figure 1.4.

In Chapter 4 we propose a deep learning model for place recognition under appearance changes. We propose several networks that generate a descriptor for a given image or, for the first time, a sequence of images that can easily be matched with another previously taken in the same place by using a nearest neighbor algorithm. A descriptor is a vector of numbers, each image generates its own and our goal is to ensure that images taken at the same place generate similar vectors while the images that are taken at different places do not. We have also proposed a partition of the Nordland dataset and made it free access. We proposed this partition to make the most challenging bias-free data splits. Details of the dataset are shown in Chapter 4. Recently Warburg et al. (2020) introduced a large dataset for long-term localization and have evaluated our ideas of sequence descriptors, ultimately confirming our findings.

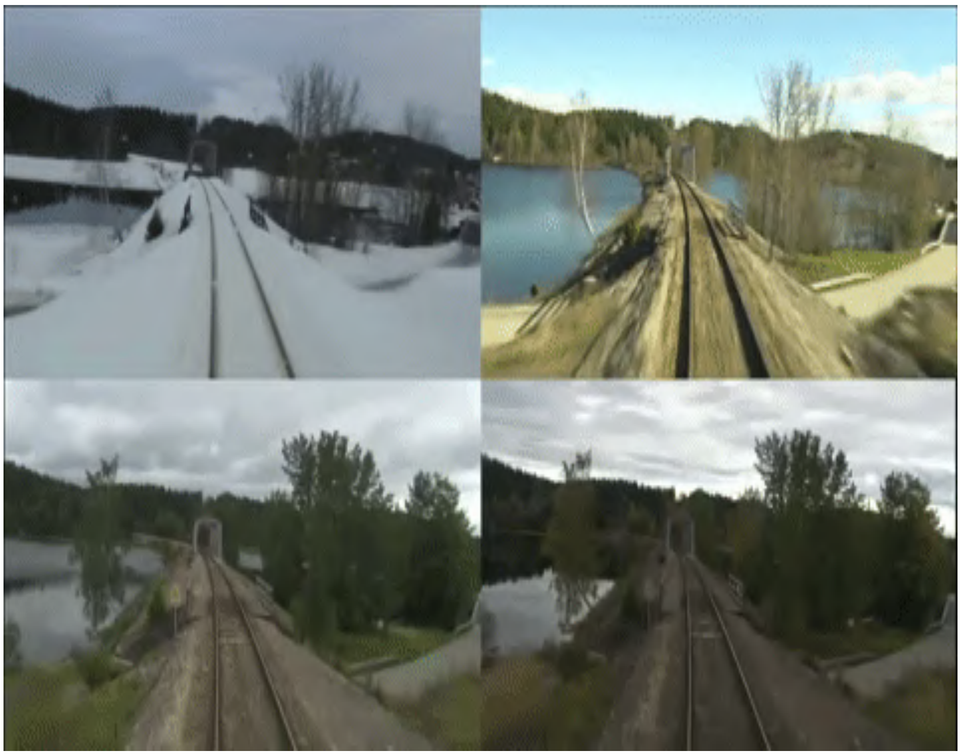


Fig. 1.4 Images from the Nordland Dataset. The four images were taken from the same point but despite looking very different we are capable of telling that they are the same place. From top-left in clockwise order the images were taken in winter, summer, fall and spring.

1.3.4 Visual Reconstruction of High-Level Structures

It is incredibly useful to recognize the patterns of the structure around us. Having certain knowledge of the layout can help us navigate or reconstruct the environment better. A easily seen example of this is driving; we, humans, leverage the structure of a scene to track our position and simultaneously build a map of the surrounding environment. We correctly assume that the lines marking lanes are parallel and that the road between these lanes is a plane or quite planar.

Previous works have used scene priors in Visual SLAM as well. Concha et al. (2014) proposes a piece-wise-planar regularization that leverages from recognizing planar parts of the map that are very common on man-made scenes. Another common pattern in man-made structures are straight lines, successfully used and combine with salient points by Gomez-Ojeda et al. (2016). Fouhey et al. (2013) recovers 3D data-driven primitives that were later used by Concha et al. (2015) to improve monocular mapping. Pérez-Yus et al. (2014) detects a more specific and high-level structure: stairs; a work that was later extended in Perez-Yus et al. (2017) by adding a visual odometry module and detecting the stairs also during traversal motion. Salas et al. (2015) build a Visual SLAM system that is aware of the layout of a room, tracking it through different views. Recovering the layout of an indoor room is the detection of the lines that delimit the structure of it, *i.e.* the separation between walls, floor and ceiling.

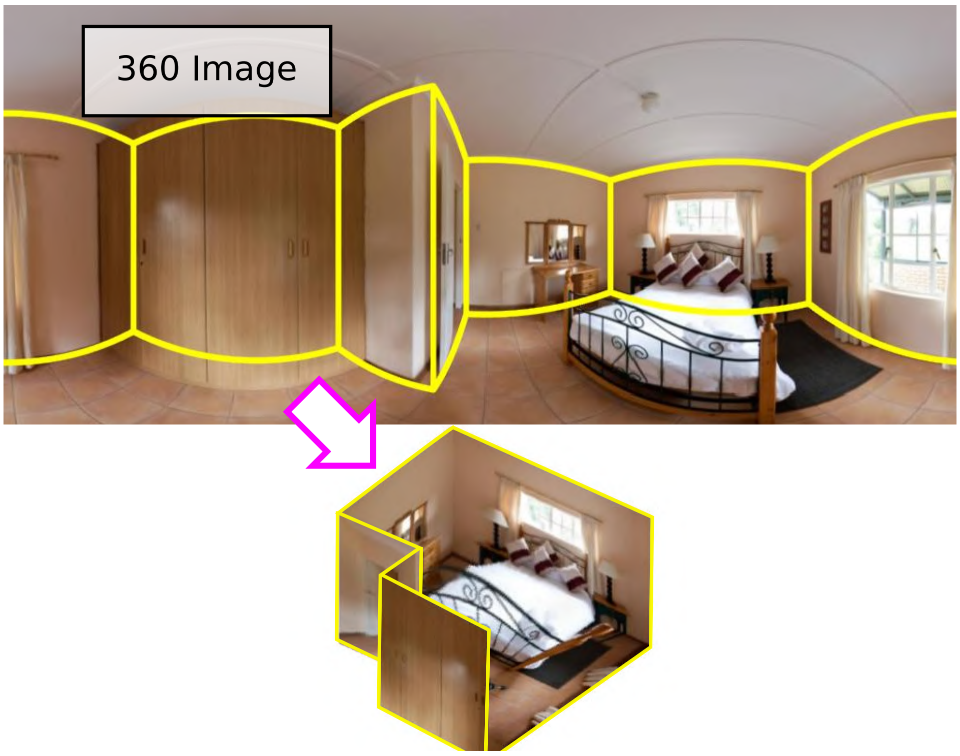


Fig. 1.5 Image taken from Fernandez-Labrador et al. (2018a). This layouts were recovered from a single image taken from a 360 camera and it is represented as a equirectangular image.

In this thesis, specifically in Chapter 5, we have focused on the improvement of this layout detection for indoor rooms. We have proposed the first corner prediction end-to-end neural network for layout recovery from 360 panoramas. See Figure 1.5 for an illustrative example. We have presented a model that directly recovers the corners of the room with no pre- or post-processing, from there with very little computation we are capable of recovering a very simple and schematic 3D model. Another contribution of this work is the development of EquiConvs, a convolution type that deforms the shape of the kernel to match the distortion created by the equirectangular representation of a panoramic image.

1.3.5 Visual SLAM on Dynamic Environments

The last of the topics addressed in this thesis is SLAM in dynamic environments (see Figure 1.6 for an illustration of the problem). Including dynamic objects in our reconstruction can be significantly challenging since they move, their changing positions can confuse our tracking if we use them to estimate our pose. As humans, we experiment a similar effect when we sit inside a train and through the window we see another train in the station. Sometimes one of the two trains starts to move and we do not really understand which one of the two is moving. Sometimes we wrongly perceive as if we were moving because we use a moving image to locate ourselves. For visual tracking we locate the camera with respect to the parts of the scene we are viewing, but if we are not careful of using only parts of the scene that are static as an anchor to estimate the pose of the device, the recovered pose can be erroneous. Riazuelo et al. (2017) proposed a RGB-D SLAM system with an embedded real-time human tacker module. Li and Lee (2017) also proposed a similar RGB-D SLAM system, but differently to previous approaches, they use depth edge points as an indicator of the probability that parts of the image belong to a dynamic object.

We propose, in Chapter 6, a combination of traditional geometry and deep learning techniques to detect and consider dynamic objects during tracking and mapping for monocular, stereo and RGB-D SLAM. This system combines the use of a semantic segmentation CNN with a classical feature-based SLAM system. More recent works have advanced more in the topic of SLAM with dynamic objects. Yang and Scherer (2019) build an RGB-D SLAM system where by retrieving 3D bounding boxes of objects they jointly optimize the poses of the camera, objects and points among different views. Xu et al. (2019) are able to maintain a map tracking all the instances of objects by building an object-level octree-based volumetric representation, providing a robust RGB-D camera tracking as well.

1.4 List of Publications

The work developed during this PhD thesis has produced the following publications.

- Facil, J. M., Concha, A., Montesano, L., & Civera, J. (2017). **Single-view and Multi-view Depth fusion.** IEEE Robotics and Automation Letters, 2(4), (pp. 1994-2001). With Oral Presentation at International Conference on Intelligent Robots and Systems (IROS) 2017.

- Facil, J. M., Ummenhofer, B., Zhou, H., Montesano, L., Brox, T., & Civera, J. (2019). **CAM-Convs: CAM-Convs: Camera-Aware Multi-scale Convolutions for Single-View Depth.** IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2019 (pp. 11826-11835).

- Olid, D., Fácil, J. M., & Civera, J. (2018). **Single-view place recognition under seasonal changes.** Planning, Perception and Navigation for Intelligent Vehicles Workshop at International Conference on Intelligent Robots and Systems (IROS) 2018.

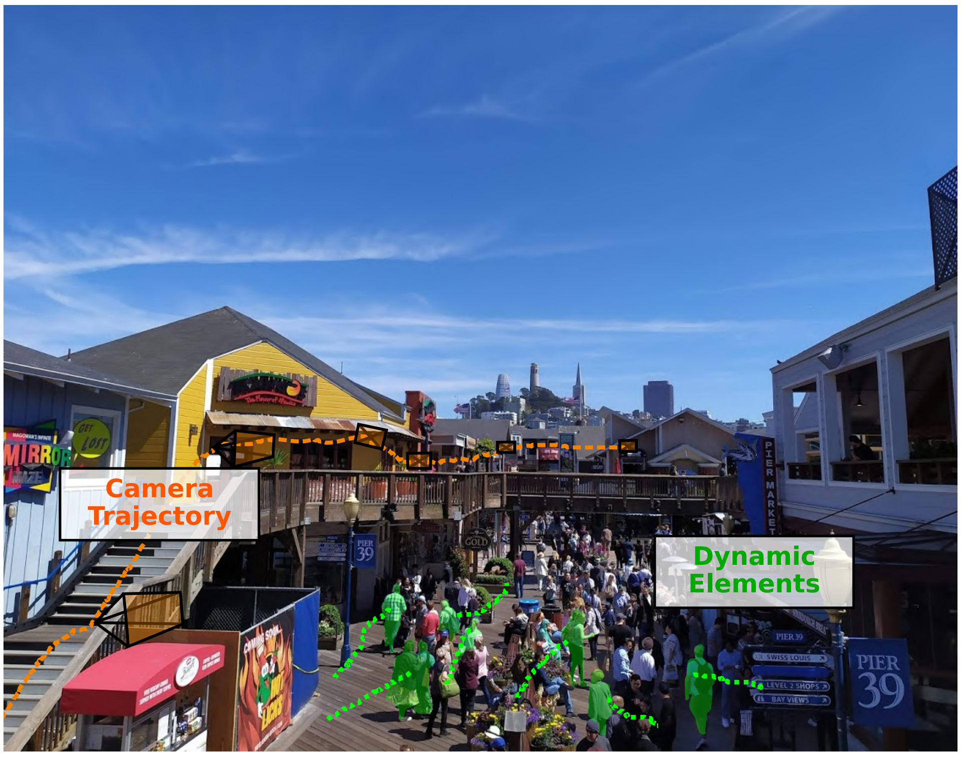


Fig. 1.6 This image shows an example of how dynamic objects are detected over time and taken into account when tracking the camera position. At the same time, the map could be built, not considering the dynamic objects because they should not be part of it.

- Facil, J. M., Olid, D., Montesano, L., & Civera, J. (2019). **Condition-Invariant Multi-View Place Recognition.** Technical Report 2019 – arXiv preprint arXiv:1902.09516.

- Fernandez-Labrador, C., Facil, J. M., Perez-Yus, A., Demonceaux, C., & Guerrero, J. J. (2018). **PanoRoom: From the Sphere to the 3D layout**. 3D Meets Semantics at European Conference in Computer Vision (ECCV) 2018.

- Fernandez-Labrador, C.*, Facil, J. M.*, Perez-Yus, A., Demonceaux, C., Civera, J., & Guerrero, J. J. (2020). **ThreeSixty End-to-End Layout Recovery**. Woman in Computer Vision at IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2019. * - *Equal Contribution*

- Fernandez-Labrador, C.*, Facil, J. M.*, Perez-Yus, A., Demonceaux, C., Civera, J., & Guerrero, J. J. (2020). **Corners for layout: End-to-end layout recovery from 360 images**. IEEE Robotics and Automation Letters, 5(2), (pp. 1255-1262). * - *Equal Contribution*

- Bescos, B., Facil J.M., Civera, J. & Neira, J., **Detecting, Tracking and Eliminating Dynamic Objects in 3D Mapping using Deep Learning and Inpainting**, Oral and Poster Presentation within the Workshop at ICRA 2018: Representing a Complex World: Perception, Inference, and Learning for Joint Semantic, Geometric, and Physical Understanding

- Bescos, B., Facil J.M., Civera, J. & Neira, J., **Robust and Accurate 3D Mapping by combining Geometry and Machine Learning to deal with Dynamic Objects**, Poster Presentation within the Workshop at IROS 2017: Learning for Localization and Mapping

- Bescos, B., Facil J.M., Civera, J. & Neira, J., **DynaSLAM: Tracking, Mapping and Inpainting in Dynamic Scenes**, IEEE Robotics and Automation Letters 3 (4), (pp. 4076 - 4083). With Oral Spotlight and Poster at International Conference on Intelligent Robots and Systems (IROS) 2018.

1.5 Code Released

During the realization of this thesis we have released the following code repositories.

- Source code for CAM-Convs: Camera-Aware Multi-Scale Convolutions for Single-View Depth in TensorFlow 1.4 and 2.0. `https://github.com/jmfacil/camconvs`

- Source code for Single-View Place Recognition in Caffe. `https://github.com/jmfacil/single-view-place-recognition`

- Source code of Corners-for-Layout and EquiConvs (EquiRectangular Convolutions) for TensorFlow 1.4. `https://github.com/cfernandezlab/CFL`

- Source code of DynaSLAM, build upon ORB-SLAM2, Tracking, Mapping and Inpainting in Dynamic Scenes for Monocular, Stereo and RGB-D Cameras. `https://github.com/BertaBescos/DynaSLAM`

1.6 Manuscript Organization

The structure of this Ph.D. thesis is as follows. In the Chapter 2 we present CAM-Convs for monocular single-image depth estimation with different cameras. In Chapter 3 we propose a fusion between single and multi-view depth to improve mapping with very little motion and texture for Visual SLAM. Chapter 4 presents several deep neural network architectures for robust visual place recognition considering appearance changes. In Chapter 5 we introduce Corners for Layout (CFL), an end-to-end network to extract corners of indoor rooms, and the Equirectangular Convolutions (EquiConvs), a type of convolution that allows CNNs to adapt to equirectangular distortions. Finally, Chapter 6 presents DynaSLAM, a monocular, stereo and RGB-D SLAM system that is aware of dynamic objects on the scene and ignores them when tracking the camera pose and mapping the scene.

Chapter 2

Camera-Aware Multi-Scale Convolutions for Single-View Depth

As we have mentioned in the introduction chapter, structure without motion is an ill-posed problem that is not solvable in general with traditional geometric techniques. For that reason many learning approaches have been presented over the last years trying to tackle this problem. Deep learning, as in many other problems, have proven to show the best results. However, single-view depth estimation suffers from the problem that a network trained on images from one camera does not generalize to images taken with a different camera model. Thus, changing the camera model requires collecting an entirely new training dataset. In this chapter, we propose a new type of convolution that can take the camera parameters into account, thus allowing neural networks to learn calibration-aware patterns. Our experiments confirm that this improves the generalization capabilities of depth prediction networks considerably, and clearly outperforms the state of the art when the train and test images are acquired with different cameras.

2.1 Introduction

Recovering 3D information from 2D images is one of the fundamental problems in computer vision that, due to recent advances and applications, is receiving nowadays a renewed attention. Among others, there has been recent relevant results on problems such as 6D object pose detection (Kehl et al., 2017, Shrivastava et al., 2017, Sundermeyer et al., 2018), 3D model reconstruction Fan et al. (2017), Tatarchenko et al. (2017), depth estimation from single (Laina et al., 2016, Fu et al., 2018, Li et al., 2018) and multiple views (Ummenhofer et al., 2017, Huang et al., 2018), 6D camera pose recovery (Kendall et al., 2015, 2017) or camera tracking and mapping (Zhou et al., 2018, Bloesch et al., 2018, Tang and Tan, 2018,

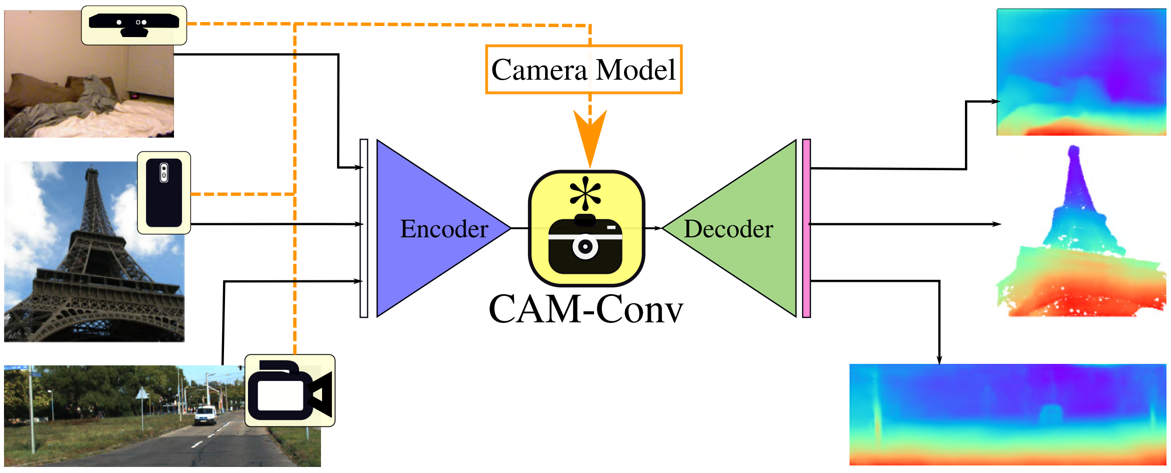


Fig. 2.1 CAM-Convs allows efficient specialization of a camera-generic network for various camera models by feeding camera-specific parameters into the network.

Tateno et al., 2017). While traditional multi-view methods (Schonberger and Frahm, 2016) are mostly based on geometry and optimization and, thus, are largely independent of the data, these recent deep learning approaches depend on training data that demonstrates the mapping from images to depth.

The common strategy to collect such data is by using an RGBD sensor, like the Kinect camera, which conveniently provides both the RGB image and what can be considered ground truth depth. It is implicitly assumed that training on this type of data will generalize to other RGB sensors that do not provide depth. However, the evaluation of recent learning-based methods relies largely on public benchmarks where images have been recorded with the same RGBD camera as the training data. Thus, evaluation on these benchmarks does not reveal whether a depth estimation method generalizes to RGB images from another camera.

Overfitting to a benchmark is a common problem in computer vision research. Torralba and Efros (2011) have shown that datasets may have strong biases that make researchers over-confident regarding the performance of their method. In particular, train-test divisions of the same kind of data are not enough to prove generalization. In this work we show that, indeed, state-of-the-art single-view depth prediction networks do not generalize when the camera parameters of the test images are different from the training ones.

Moreover, we show that for single-view depth prediction the problem of missing generalization to images from different cameras is even more severe: it cannot be solved by training on images from a diverse set of cameras with different parameters. For present methods to adapt to a different camera model, they require changes in the architecture.

We present a deep neural network for single-view depth prediction that, for the first time, addresses the variability on the camera's internal parameters. We show that this allows to use images from different cameras at train and test time without a performance degradation. This is of particular interest, as it enables the exploitation of images from *any* camera for training the data-hungry deep networks. Specifically, within our proposed network, our main contribution is a novel type of convolution, that we name as CAM-Convs (Camera-Aware Multi-scale Convolutions), that concatenates the camera internal parameters to the feature maps, and hence allows the network to learn the dependence of the depth from these parameters. Figure 2.1 shows an illustration of how CAM-Convs act in the typical encoder-decoder depth estimation pipeline. The network can be trained with a mixture of images from different cameras without overfitting to specific intrinsics. We show that the network generalizes also to images from cameras it has not been trained on. A comparison with the state of the art in single-image depth estimation demonstrates that the better generalization properties do not reduce the accuracy of the depth estimates.

2.2 Related Work

Estimating 3D structure and 6 degrees-of-freedom motion using deep learning has been addressed recently from several angles: Supervised (Laina et al., 2016) and unsupervised (Zhou et al., 2017), from single (Eigen and Fergus, 2015) and multiple views (Tang and Tan, 2018), using end-to-end networks (Laina et al., 2016) or fusing with multi-view geometry (Fácil et al., 2017), completing depth maps (Zhang et al., 2018, Weerasekera et al., 2018), and estimating geolocation (Weyand et al., 2016, Kendall et al., 2015), relative motion (Ummenhofer et al., 2017), visual odometry (Wang et al., 2017, 2018), and simultaneous localization and mapping (SLAM) (Tateno et al., 2017, Bloesch et al., 2018, Zhou et al., 2018).

In this work we deal with single-view supervised depth learning, so we will focus our literature review in this case. Among the pioneering work we can reference Hoiem et al. (2005), that similarly to pop-up illustrations, cut and fold a 2D image based on a segmentation into geometric classes and some geometric assumptions. Saxena et al. (2009) is another seminal work that, with minimal assumptions on the scene, learned a model based on a MRF. Eigen et al. (2014) was the first paper that used deep learning for single-view depth prediction, proposing a multi-scale depth network. Its results were improved later by Eigen and Fergus (2015), Liu et al. (2015b), Laina et al. (2016), Chakrabarti et al. (2016) and He et al. (2018).

Many methods focus on specific datasets which enable to train learning-based methods for specific tasks. For instance, Eigen and Fergus (2015) extend the multi-scale architecture in Eigen et al. (2014) to the prediction of surface normals and semantic labels on the NYU dataset (Silberman et al., 2012). Similarly, Wang et al. (2015) train a network that jointly predicts depth and segmentation on the same dataset. For depth, Laina et al. (2016), Liu et al. (2015b) and Eigen and Fergus (2015) show that their methods can be adapted to other datasets like Make3D (Saxena et al., 2009) or KITTI (Geiger et al., 2012). However, they treat datasets like different tasks and require retraining for each dataset to achieve state-of-the-art performance.

Chen et al. (2016), inspired by Zoran et al. (2015), introduce the Depth in the Wild dataset and train a CNN using ordinal relations between point pairs. While the images stem from internet photo collections taken with many different cameras, they do not make use of the camera parameters during training. Li and Snavely (2018) use a structure from motion pipeline to extract depth from internet photo collections and use this to train a CNN predicting depth up to a scale factor. Again, information about camera parameters is not exploited and generalization is solely driven by large diverse datasets. Extrinsic parameters have been considered for other tasks such as stereo estimates (Ummenhofer et al., 2017) or synthesis of view point changes (Zhou et al., 2016). Intrinsic parameters are usually left out in deep learning pipelines, with the exception of He et al. (2018). They embed focal length information in a fully-connected approach, making it impossible to train and test in different image sizes, while our proposal is flexible and can deal with different image sizes. Posterior to the publication of this work, López-Antequera et al. (2020) have proposed a canonical camera model, they show that converting every image to this model it is a simpler method to train than the one we propose, despite of having some drawbacks like forcing the images to have same size and optical center can force to drop parts of them.

In the next section we describe how to explicitly implement the internal camera parameters into the network and thereby improve generalization by CAM-Convs.

2.3 Camera-Aware Multi-scale Convolutions

CAM-Convs (standing for Camera-Aware Multi-scale Convolutions), is the variant of the convolution operation that we present in this thesis. CAM-Convs include the camera intrinsics in the convolutions, allowing the network to learn and predict depth patterns that depend on the camera calibration. Specifically, we add CAM-Convs in the mapping from RGB features to 3D information–*e.g.* depth, normals–, that is, between the encoder and the decoder. As shown in Figure 2.2, we add them at every level, such that we include CAM-Convs on every skip-connection too. Notice that all the CAM-Convs are added after the encoder, allowing the use of pretrained models.

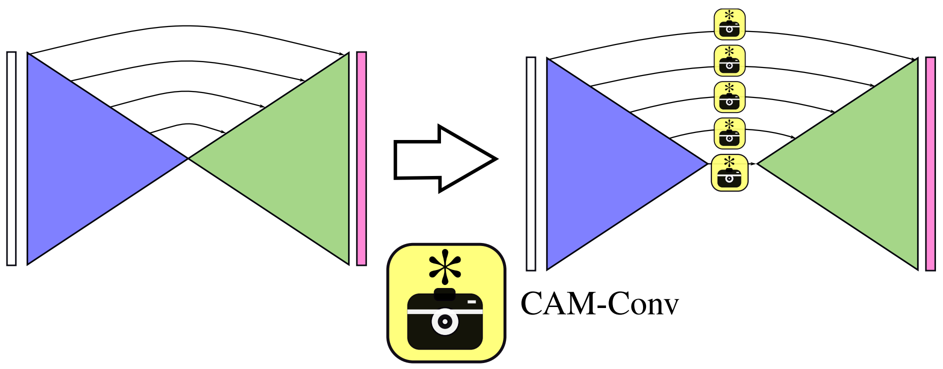


Fig. 2.2 Adding CAM-Convs to an Encoder-Decoder U-Net architecture.

The basics of CAM-Convs are as follows: We pre-compute pixel-wise coordinates and field-of-view maps and feed them along with the input features to the convolution operation. CAM-Convs use the idea behind Coord-Convs presented in Liu et al. (2018), on adding normalized coordinates per pixel, but incorporating information on the camera calibration. An illustrative scheme of how CAM-Convs extra channels work is shown in Figure 2.3. The different maps included are computed using the camera intrinsic parameters (focal length f and principal point coordinates (c_x, c_y)) and the sensor size (width w and height h):

Centered Coordinates (cc)**:** To add the information of the principal point location to the convolutions, we include cc_x and cc_y coordinate channels centered at the principal point–i.e. the principal point has coordinates $(0,0)$. Specifically, the channels are

$$cc_x = \begin{bmatrix} 0-c_x \\ 1-c_x \\ \vdots \\ w-c_x \end{bmatrix}_{w\times 1} \cdot \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}_{h\times 1}^{\mathsf{T}} = \begin{bmatrix} -c_x & \cdots & w-c_x \\ \vdots & \ddots & \vdots \\ -c_x & \cdots & w-c_x \end{bmatrix} \tag{2.1}$$

$$cc_y = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}_{w\times 1} \cdot \begin{bmatrix} 0-c_y \\ 1-c_y \\ \vdots \\ h-c_y \end{bmatrix}_{h\times 1}^{\mathsf{T}} = \begin{bmatrix} -c_y & \cdots & -c_y \\ \vdots & \ddots & \vdots \\ h-c_y & \cdots & h-c_y \end{bmatrix}. \tag{2.2}$$

We resize these maps to the input feature size using bilinear interpolation and concatenate them as new input channels. These channels are sensitive to the sensor size and resolution (pixel size) of the camera, as their values depend on it. We assume the sensor size is measured in pixels. In Figure 2.3 we represent cc with a color gradient from red (for negative coordinates) to blue (for positive coordinates), white for 0. Notice in the figure how cc values change when camera sensor size, principal point or pixel size change.

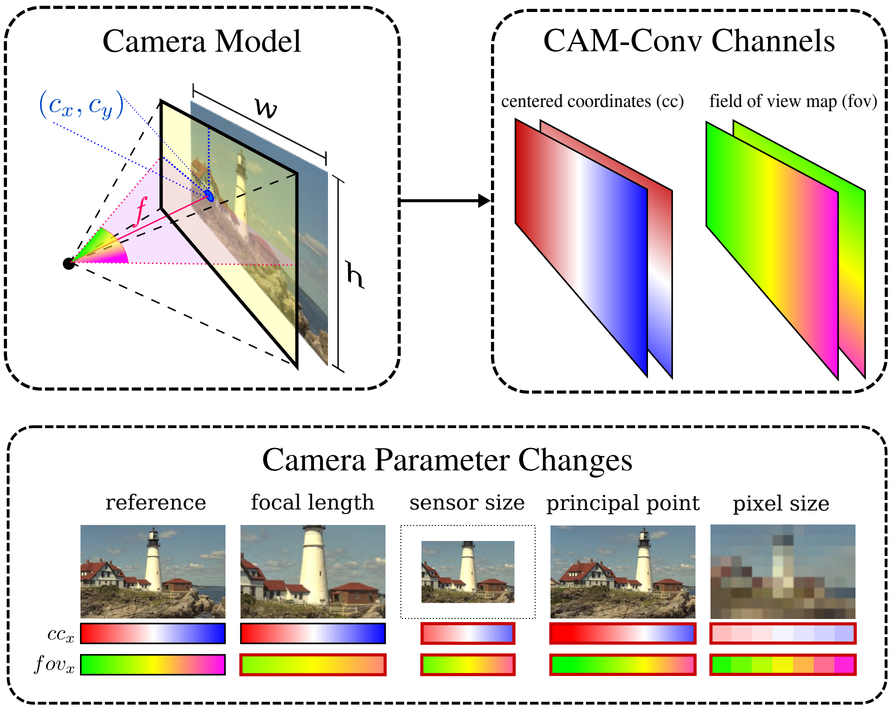


Fig. 2.3 Overview of the additional channels of our CAM-Convs (Camera-Aware Multi-scale Convolutions). We compute Centered Coordinates (*cc* from red to blue) and Field of View (*fov* from green to pink) maps. We concatenate these maps with the input features before applying the convolution. Both *cc* and *fov* depend on the camera model and are sensitive to camera changes. The Bottom part shows how *cc* and *fov* maps change with the camera parameters (a red border means the map has changed from the original).

Field of View Maps (*fov*): The horizontal and vertical *fov* maps are calculated from the *cc* maps and also depend on the camera focal length *f*

$$fov_{ch}[i,j] = \arctan\left(\frac{cc_{ch}[i,j]}{f}\right), \tag{2.3}$$

where *ch* can be *x* or *y* (see Eq. 2.1 and 2.2). They give information about the captured context and the focal length. These maps are sensitive to sensor size and focal length. In Figure 2.3 we represent *fov* with a color gradient from green to pink; yellow represents an angle of 0 in the field of view map. Notice in the bottom part of the figure how the *fov* map values change when changing camera focal length, sensor size or principal point. Changes on the pixel size change the resolution of the map but the field of view and thus the available context in the image stays the same.

Normalized Coordinates (*nc*): We also include a Coord-Conv channel of normalized coordinates (Liu et al., 2018). The values of Normalized Coordinates vary linearly with the image coordinates between $[-1,1]$. This channel does not depend on the camera sensor.

However, it is very useful to describe the spatial extent of the context (in feature space) that is left in each direction (*e.g.*, if the value on the x channel is close to -1, it means the feature vector at this position is close to the left border and there is almost no context on the left side).

Notice that nc is not shown in Figure 2.3 as it remains constant.

2.3.1 Focal Length Normalization

An instance of an object imaged by two cameras with different focal length appears with different image sizes although the depth is the same. Focal length normalization is an alternative to avoid such inconsistencies. To this end, we predict depth values normalized to a default focal length f_n. Given a metric depth map d we get the normalized depth values as $\frac{f_n}{f}d$ with f as the actual focal length. Note that the normalized depth values depend on the focal length. For the raw inverse depth predictions $\tilde{\xi}$ of our network we denormalize the values as

$$\xi = \frac{f_n}{f}\tilde{\xi} \tag{2.4}$$

where $\xi = \frac{1}{d}$ is the inverse depth map. Tateno et al. (2017) used a similar approach to correct depth values at test time, in this chapter we propose for the first time to use it during training.

This normalization can be used together with our CAM-Convs. Although CAM-Convs allows the network to learn this normalization on its own, we found in our experiments that using this normalization accelerates the convergence. It should be remarked, though, that focal length normalization assumes a constant pixel size over the whole image set, and therefore can only be used in such cases. CAM-Convs are a more general model that overcomes this limitation.

2.4 Model and Training

2.4.1 Network Architecture

The network we use in this work has an encoder-decoder architecture inspired by DispNet in Mayer et al. (2016). Hence, we add skip-connections from the low-level feature maps of the encoder to the feature maps of the same size in the decoder, and concatenate them (Ronneberger et al., 2015). Withal, we also estimate intermediate pyramid-resolution predictions, which converge faster and ensure that the network's internal features are more aimed for the task. As it is common in the literature (Laina et al., 2016, Kuznietsov et al., 2017), our network's backbone is ResNet-50, pretrained on the ImageNet Classification Dataset (He et al.,

INPUT	LAYER	K	S	FUN	CH	OUTPUT
			encoder			
image	conv+BN	7	2	ReLU	64	conv1
conv1	maxpool	3	2	-	64	pool
pool	conv-block	3	1	ReLU	256	res2a
res2a	id-block	3	-	ReLU	256	res2b
res2b	id-block	3	-	ReLU	256	res2c
res2c	conv-block	3	2	ReLU	512	res3a
res3a	id-block	3	-	ReLU	512	res3b
res3b	id-block	3	-	ReLU	512	res3c
res3c	id-block	3	-	ReLU	512	res3d
res3d	conv-block	3	2	ReLU	1024	res4a
res4a	id-block	3	-	ReLU	1024	res4b
res4b	id-block	3	-	ReLU	1024	res4c
res4c	id-block	3	-	ReLU	1024	res4d
res4d	id-block	3	-	ReLU	1024	res4e
res4e	id-block	3	-	ReLU	1024	res4f
res4f	conv-block	3	2	ReLU	2048	res5a
res4a	id-block	3	-	ReLU	2048	res5b
res4b	id-block	3	-	ReLU	2048	res5c
			skip-connections			
conv1	CAM-Convs	3	1	L-ReLU	64	conv1
res2c	CAM-Convs	3	1	L-ReLU	256	res2c
res3d	CAM-Convs	3	1	L-ReLU	512	res3d
res4f	CAM-Convs	3	1	L-ReLU	1024	res4f
res5c	CAM-Convs	3	1	L-ReLU	2048	res5c
			decoder			
res5c	upconv	4	2	L-ReLU	1024	upconv4
upconv4	conv	3	1	L-ReLU	24	inconv4
inconv4	conv	3	1	-	5	**LR-1**
LR-1,upconv4	concat	-	-	-	-	up-pre4
res4f,up-pre4	concat	-	-	-	-	in3
in3	upconv	4	2	L-ReLU	512	upconv3
upconv3	conv	3	1	L-ReLU	24	inconv3
inconv3	conv	3	1	-	5	**MR-1**
MR-1,upconv3	concat	-	-	-	-	up-pre3
res3d,up-pre3	concat	-	-	-	-	in2
in2	upconv	4	2	L-ReLU	256	upconv2
upconv2	conv	3	1	L-ReLU	24	inconv2
inconv2	conv	3	1	-	5	**MR-2**
MR-2,upconv2	concat	-	-	-	-	up-pre2
res2c,up-pre2	concat	-	-	-	-	in1
in1	upconv	4	2	L-ReLU	64	upconv1
upconv1	conv	3	1	L-ReLU	24	inconv1
inconv1	conv	3	1	-	2	**HR-1**
HR-1,upconv2	concat	-	-	-	-	up-pre1
conv1,up-pre1	concat	-	-	-	-	in
in	upconv	4	2	L-ReLU	32	upconv
upconv	conv	3	1	L-ReLU	24	inconv
inconv	conv	3	1	-	2	**HR-2**

Table 2.1 Network Architecture Details. We present three different blocks here. **1st**: *encoder* taken directly from ResNet-50, pretrained on the ImageNet Classification Dataset (He et al., 2016), sub-blocks of the *encoder* are more detailed in Table 2.2. **2nd**: *skip-connections* include also a CAM-Convs block (Table 2.2). **3rd**: *decoder* uses as inputs all the *skip-connections* and produce 5 different pyramid-resolution predictions (**bold** in the OUTPUT column).

INPUT	LAYER	K	S	BN	FUN	CH	OUTPUT
	conv-block						
I	**conv-block**	**k**	**s**	**-**	**f**	**N**	**O**
I	conv	1	s	✓	f	N/4	O2a
O2a	conv	k	1	✓	f	N/4	O2b
O2b	conv	1	1	✓	-	N	O2c
I	conv	1	s	✓	-	N	O-1
O-1,O2c	add	1	1	-	f	N	O
	id-block						
I	**id-block**	**k**	**-**	**-**	**f**	**N**	**O**
I	conv	1	1	✓	f	N/4	O2a
O2a	conv	k	1	✓	f	N/4	O2b
O2b	conv	1	1	✓	-	N	O2c
I,O2c	add	1	1	-	f	N	O
	CAM-Convs-block						
I	**CAM-Convs**	**k**	**s**	**-**	**f**	**N**	**O**
$_{fov,cc,nc}$	resize	-	-	-	-	-	$_{fov,cc,nc}$
I,$_{fov,cc,nc}$	concat	-	s	-	-	-	intern
intern	conv	k	s	-	f	N	O

Table 2.2 Sub-Blocks for the Network. The first two are the standard conv and id block from ResNet, commonly used in the literature. The **CAM-Convs-block** is the one we introduce. It takes as input 3 different maps and feeds a convolution with them, together with the input features.

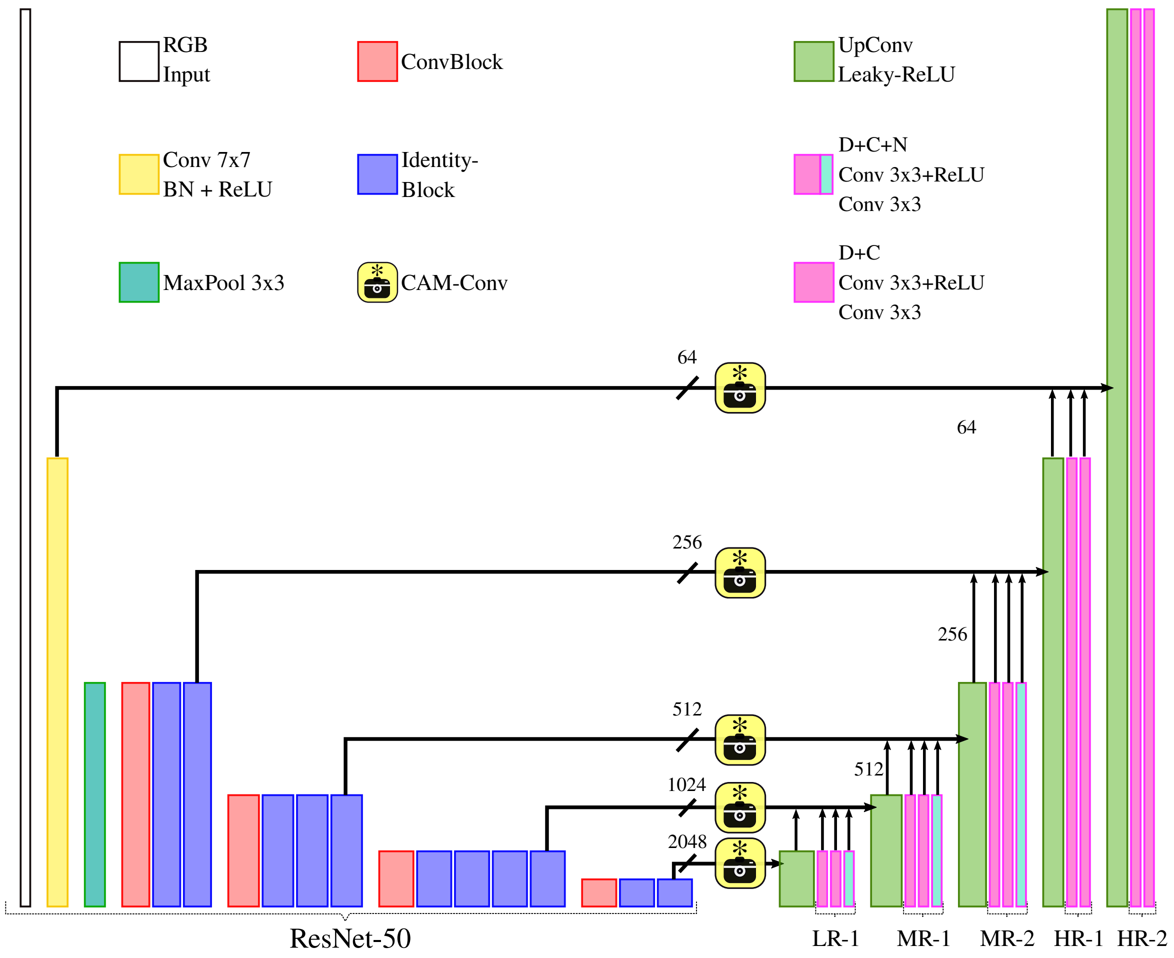


Fig. 2.4 Our network architecture, inspired by DispNet by Mayer et al. (2016), to which we added CAM-Convs connecting the encoder and decoder. We predict depth, confidence and normals (D+C+N) in the first three intermediate resolution levels (LR-1,MR-1 and MR-2) and only depth and confidence (D+C) in the last two resolution levels (HR-1 and HR-2).

2016). As suggested in Godard et al. (2018) and our experiments, pretraining the encoder on general image recognition tasks, as ImageNet, helps in both accuracy and convergence time reduction. A schematic of our network architecture can be seen in Figure 2.4. A more detailed view of the network implementation can be seen in Table 2.1 (some of the blocks used in this table are further detailed in Table 2.2). Bold output columns represent different pyramid-resolution predictions of the network.

The network predictions are composed by:

ξ**: Inverse depth** $\xi = \frac{1}{d}$**.** We chose inverse depth for its linear relationship with pixel variations.

c**: Depth confidence.** As Ummenhofer et al. (2017), we enforce the network to predict a confidence map for every depth prediction.

n: Surface normals. The normals are predicted only for small resolutions (all except the last two), as the ground-truth normals are too noisy at full resolution.

Predictions LR-1, MR-1, and MR-2 are composed of inverse depth, depth confidence and surface normals (a total of 5 channels see Table 2.1). HR-1 and HR-2 are composed only of inverse depth and depth confidence (a total of 2 channels see Table 2.1).

2.4.2 Losses

In this section we will present all the losses and their combination for the training.

Depth Loss: We minimize the L1 norm of the predicted inverse depth ξ minus the ground truth inverse depth $\hat{\xi}$, that is

$$\mathscr{L}_d = \sum_{i,j} \left| \xi(i,j) - \hat{\xi}(i,j) \right|. \tag{2.5}$$

Note that for experiments with focal length normalization we scale depth values accordingly (see section 2.3.1).

Scale-Invariant Gradient Loss: We use the scale-invariant gradient loss proposed in Ummenhofer et al. (2017), in order to favor smooth and edge preserving depth estimations. The loss based on the depths is

$$\mathscr{L}_g = \sum_{h=\{1,2,4,8,16\}} \sum_{i,j} \left\| \mathbf{g}_h[\xi](i,j) - \mathbf{g}_h[\hat{\xi}](i,j) \right\|_2. \tag{2.6}$$

For the gradients, we use the same discrete scale-invariant finite differences operator **g** as defined in their work, which is

$$\mathbf{g}_h[d](i,j) = \left(\frac{d(i+h,j) - d(i,j)}{|d(i+h,j) + d(i,j)|}, \frac{d(i,j+h) - d(i,j)}{|d(i,j+h) + d(i,j)|} \right)^\top, \tag{2.7}$$

and we apply the scale-invariant loss to cover gradients at 5 different spacings h.

Confidence Loss: The ground truth for the confidence map must be calculated online as it depends on the prediction. The confidence ground truth is calculated as

$$\hat{c}(i,j) = e^{-|\xi(i,j) - \hat{\xi}(i,j)|}, \tag{2.8}$$

and its corresponding loss function is defined as

$$\mathscr{L}_c = \sum_{i,j} \left| c(i,j) - \hat{c}(i,j) \right|. \tag{2.9}$$

Normal Loss: For the normal loss, we use the L2 norm. The ground truth for the normals ($\hat{\mathbf{n}}$) is derived from the ground truth depth image. The loss for the normals is as follows:

$$\mathscr{L}_n = \sum_{i,j} \left|\left|\mathbf{n}(i,j) - \hat{\mathbf{n}}(i,j)|\right|\right|_2. \tag{2.10}$$

Total Loss: The individual losses are weighted by factors obtained empirically, so the total loss $\mathscr{L}$ is

$$\mathscr{L} = \lambda_1 \mathscr{L}_d + \lambda_2 \mathscr{L}_g + \lambda_3 \mathscr{L}_c + \lambda_4 \mathscr{L}_n, \tag{2.11}$$

where λ_1, λ_2, λ_3 and λ_4 are 150, 100, 50 and 25 respectively.

2.4.3 Training Schedule

We train all our networks using the TensorFlow framework (Abadi et al., 2016). We start from ResNet-50 (pre-trained) and a randomly initialized decoder. For optimization we use Adam Optimizer (Kingma and Ba, 2014) with a momentum of 0.9. The complete training of the network is composed by three different stages, each adding more layers and predictions to the decoder (SR,MR and HR respectively in Table 2.1).

1st stage. We train until the first two resolutions of the decoder (LR-1 in Table 2.1). This stage is the shortest one, trained only for $10k$ iterations with batch size 16. We only train the encoder layers and the decoder until the smallest prediction. We do not apply the scale-invariant loss for this resolution.

2nd stage. We train until the next two predictions (LR-1, MR-1 and MR-2 in Table 2.1). As in the previous stage we train only the layers that affect the outputs. This stage is trained for $50k$ iterations with batch size 16. We apply a scale-invariant loss to prediction MR-2 after $25k$ iterations.

3rd stage. We train the whole network, for $200k$ iterations with batch size 16. We apply a scale-invariant loss to predictions MR-2, HR-1 and HR-2 after $25k$ iterations.

Learning rate. The learning rate policy for the three stages is shown in Figure 2.5. The base learning rates for the three stages are $1e-3$, $5e-4$ and $1e-4$. The learning rate drops along the iterations, never being less than a minimum of $1e-6$. As we train multiple models we use a fixed automatic learning rate decay.

Weighting losses. For all stages we minimize the losses for several resolutions. We scale the losses according to the resolution level. Specifically, we multiply the losses by a factor $\frac{1}{k}$, where k denotes the resolution level. Starting with the finest resolution of the active stage. *I.e* in the 1st stage LR-1 prediction loss would be multiplied by 1. While in the 3rd stage LR-1

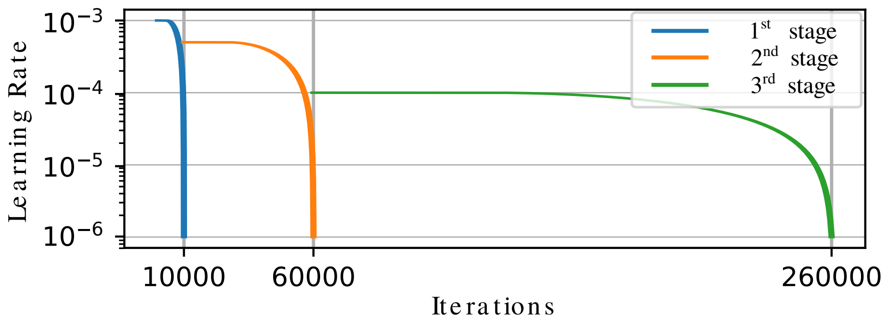


Fig. 2.5 Learning rate policy for the three-stages training.

would be multiplied by $\frac{1}{5}$, MR-1 would be multiplied by $\frac{1}{4}$ and so on until HR-2 that would be multiplied by 1.

2.5 Multi-Camera Experiments and Results

Most of the single-view depth prediction networks have been trained and tested using the same or very similar camera models. Generalizing to different camera models has several implications that are not straightforward. For this reason, we first present a thorough analysis on the generalization capabilities of current approaches. To this end we apply naïve generalization techniques (focal normalization and image resizing) during training on a network without our special convolutions (as Figure 2.4 but without CAM-Convs) and examine the limitations. Finally, we train and evaluate our network with CAM-Convs (as Figure 2.4) and show the improved generalization performance with respect to different camera parameters.

2.5.1 Experimental Setup

The major part of our experiments are done on the 2D-3D Semantics Dataset (Armeni et al., 2017), that contains RGB-D equi-rectangular images. This dataset allows us to generate images with different camera intrinsics *but* the same content. We have observed that depth estimation networks overfit to the camera parameters and the image content distribution (the latter being different in indoors and outdoors datasets, for example). In this manner we eliminate the content distribution factor and isolate the effect of the camera parameters.

All the experiments were done using the 3-fold cross-validation suggested by Armeni et al. (2017). In this section we present median values for the most relevant experiments. To see the complete results, more details on the dataset and image generation process and additional experiments we refer the reader to the Appendix A.

Name	s_1	s_2	s_3
Sensor	256×192	192×256	224×224

Name	s_4	s_5	s_S	s_K
Sensor	128×96	320×320	256×192	384×128

Name	f_{72}	f_{128}	f_{64}	f_n
Focal	72	128	64	100

Table 2.3 Notation for different sensor sizes and focal lengths.

The notation for sensor sizes and focal lengths used during the evaluation is in Table 2.3. As an example, if a network has been trained with sensor sizes 192×256 and 224×224, and focal length 72, we will denote this model as $s_2 s_3 f_{72}$. In some experiments we use a random distribution for the focal length. As an example, if the synthesized focal lengths are uniformly distributed between 72 and 128, the model will be denoted as $\mathcal{U} f_{72} f_{128}$.

We evaluate the performance on both depth and inverse depth. All the error metrics we used in our experiments are standard from the literature. In addition we use relative metrics and the scale-invariant metric presented by Eigen et al. (2014), which are widely used in depth estimation.

2.5.2 Influence of context

Modifying the camera parameters affects the field of view, and hence the amount of context the image is capturing. We evaluate the influence of the context in the depth prediction of a standard U-Net encoder-decoder architecture (network in Figure 2.4 without CAM-Convs) with two different experiments. First, we compare two networks trained with images with sensor size s_1 and two different focal lengths f_{128} and f_{64} (Table 2.4). Second, we compare two networks with images with the same focal length but different sensor sizes: s_1 and s_4 (Table 2.5).

As expected, context helps. The performance is better for the smallest focal f_{64}, which results in a wider FOV and hence more context. Also the performance is better for the bigger sensor size s_1, which also provides more context. To remove the context dependency in our analysis, for some of the experiments in next subsections we will generate images with uniformly distributed focal lengths.

2.5.3 Overfitting of standard networks

In this experiment we evaluate the performance of a standard U-Net architecture for variations of the camera parameters on the training and test sets. We will focus the study on two

Test	**Train**	abs.rel	rmse	sc.inv	sq.rel
		: 1	m	$lg(m)$	: 1
s_1f_{64}	s_1f_{64}	**0.17**	**0.378**	**0.0347**	**0.048**
s_1f_{128}	s_1f_{128}	0.195	0.51	0.0387	0.0606
		smaller is better			

Table 2.4 Influence of context, different focal lengths.

Test	**Train**	abs.rel	rmse	sc.inv	sq.rel
		: 1	m	$lg(m)$	: 1
s_1f_{64}	s_1f_{64}	**0.17**	**0.378**	**0.0347**	**0.048**
s_4f_{64}	s_4f_{64}	0.204	0.54	0.0384	0.0637
		smaller is better			

Table 2.5 Influence of context, different sensor sizes.

parameters: (a) focal length and (b) sensor size. First we will fix the sensor size to s_1 and we will test on images with focal lengths f_{64}, f_{72} and f_{128} (first three test sets in Table 2.6). Second we will sample random focal lengths from a uniform distribution between f_{72} and f_{128} and we will evaluate on images with sensor sizes s_1 and s_2 (last two test sets in Table 2.6). For every test set there are 4 to 5 different train sets (referred in the 2nd column of the table). For every test set we will refer to the case where the cameras from the training and test set are the same as the same-camera baseline. Training sets where we did not use focal length normalization are denoted with a '*'. Networks trained on train sets with two sensor sizes have been trained either as Siamese networks with weight sharing or with image resizing to size s_1 (denoted with a '†').

It is important to remark that, for all the experiments, the test and training data was generated from the exact same images and the networks have the same architecture and were trained for the same number of iterations. Any performance variation, then, should be attributed to the variations in the camera intrinsics and the naïve solutions we analyze. Notice in Table 2.6 that, in general, the same-camera baseline outperforms the rest, demonstrating the overfit to the camera parameters.

The conclusions of these experiments are as follows.

(a) Single-focal training overfits. The performance of a depth network degrades when trained on images from a particular camera and tested on images from different cameras. See, for example, the drop in performance between the 1st row (test: s_1f_{64}, train: $s_1f_{64}{}^*$) and the 2nd (test: s_1f_{64}, train: s_1f_{72}) and 3rd (test: s_1f_{64}, train: s_1f_{128}) rows in all metrics.

Multi-focal training with normalization helps. The results improve when the training set contains images with different focal lengths and is done with focal normalization. See, for example, that the results on test set s_1f_{64} with training set $s_1f_{72}f_{128}$ is close to the same-camera baseline. Notice, however, that the multi-focal train set does not reach the

Test set	**Train set**	l1.inv	rmse	sc.inv
pixels	*pixels*	$1/m$	m	$lg(m)$
$s_1 f_{64}$	$s_1 f_{64}{}^*$	**0.184**	**0.378**	**0.0347**
	$s_1 f_{72}$	0.193	0.395	0.0354
	$s_1 f_{128}$	0.318	0.572	0.0483
	$s_1 f_{72} f_{128}{}^*$	0.659	0.864	0.0614
	$s_1 f_{72} f_{128}$	0.189	0.387	0.0361
$s_1 f_{72}$	$s_1 f_{72}{}^*$	**0.17**	**0.4**	**0.0354**
	$s_1 f_{128}$	0.272	0.564	0.0459
	$s_1 f_{72} f_{128}{}^*$	0.552	0.888	0.0609
	$s_1 f_{72} f_{128}$	0.175	0.404	0.0364
$s_1 f_{128}$	$s_1 f_{128}{}^*$	0.141	0.51	0.0387
	$s_1 f_{72}$	0.133	0.524	0.0411
	$s_1 f_{72} f_{128}{}^*$	0.208	0.813	0.063
	$s_1 f_{72} f_{128}$	**0.132**	**0.504**	**0.038**
$s_1 \mathscr{U} f_{72} f_{128}$	$s_1 \mathscr{U} f_{72} f_{128}$	**0.15**	**0.46**	**0.037**
	$s_2 \mathscr{U} f_{72} f_{128}$	0.175	0.51	0.0422
	$s_1 s_2 \mathscr{U} f_{72} f_{128}$	0.153	0.484	0.0401
	$s_1\ s_2\ s_3 \mathscr{U} f_{72} f_{128}{}^\dagger$	0.179	0.742	0.064
$s_2 \mathscr{U} f_{72} f_{128}$	$s_1 \mathscr{U} f_{72} f_{128}$	0.151	0.44	0.038
	$s_2 \mathscr{U} f_{72} f_{128}$	**0.133**	**0.412**	**0.0323**
	$s_1 s_2 \mathscr{U} f_{72} f_{128}$	0.139	0.436	0.0352
	$s_1\ s_2\ s_3 \mathscr{U} f_{72} f_{128}{}^\dagger$	0.16	0.622	0.0514
		smaller is better		

* trained without focal length normalization.
† images resized to s_1 during training.

Table 2.6 Overfit to camera parameters of standard encoder-decoder architectures. Networks trained from images with variations in their intrinsics perform worse than the same-camera baseline.

performance of the same-camera baseline. In section 2.5.4 we will show how CAM-Convs are able to outperform the same-camera baseline even when the training data does not contain the test focal length.

The performance degrades without focal normalization. Compare, for example, the error metrics of the train sets $s_1 f_{72} f_{128}{}^*$ and $s_1 f_{72} f_{128}$. Networks trained on $f_{72} f_{128}{}^*$, in fact, did not converge easily.

Limitations of focal normalization. Two things should be noticed regarding focal normalization: First, it does not model the changes on the sensor size and the resolution, and we will see now how changes on them degrade the performance. And second, Equation 2.4 only holds if the pixel size is the same for every camera in the training and test sets, which in general is not the case.

Test	**Train**	abs.rel	rmse.inv	sc.inv	sq.rel
		%	$1/km$	$lg(m)100$	%
	s_K	9.16	**10.54**	**13.3**	**2.33**
s_K	$s_S s_K$	24.58	36.82	26.51	9.28
	$s_S s_K^\dagger$	**9.08**	10.55	13.98	2.56
		abs.rel	l1.inv	rmse.inv	sq.rel
		$:1$	$1/m$	$1/m$	$:1$
	s_S	**0.12**	**0.09**	**0.12**	**0.03**
s_S	$s_S s_K$	0.26	0.13	0.16	0.18
	$s_S s_K^\dagger$	**0.12**	**0.09**	**0.12**	**0.03**
		smaller is better			

† sensor size has been resized to the first one in the list.

Table 2.7 Naïve train and test on KITTI Uhrig et al. (2017) and ScanNet Dai et al. (2017a). See that training FCN in multiple image sizes ($s_K s_S$) does not generalize. Resizing works, but only in this particular case, because of the small overlap of visual features.

(b) Single-sensor size training overfits. Networks trained on a sensor size and tested on other sensor sizes do not perform as well as the same-camera baseline. This can be seen in Table 2.6 in the last two test sets $s_1\mathscr{U}f_{72}f_{128}$ and $s_2\mathscr{U}f_{72}f_{128}$. Single-view depth estimation is a context-dependent task, and the network overfits to the amount of context in the training sensor size.

Multi-sensor size training with weight sharing does not generalize. Training with multiple sensor sizes works better than training with the wrong sensor size but cannot reach the same performance as same-camera baselines. Further, training a stack of weight sharing networks also does not scale to large numbers of different sensor sizes.

Resizing does not work. As a naïve approach, which scales to multiple sensor sizes, we use resizing (denoted with '†' in Table 2.6), which converts all the images to size (s_1) during training. Notice that resizing changes the aspect ratio. It also implies the recalculation of a new average focal length $f_r = f\frac{r_x+r_y}{2}$ for normalization. The performance degradation introduced by resizing is noticeable. Resizing creates inconsistent data in train and testing, which leads to learning and convergence difficulties.

Resizing helps only in a particular case (non-overlapping distributions of visual features). Table 2.7 shows an experiment, similar to the previous one, on two public datasets: KITTI (Uhrig et al., 2017), with sensor size s_K, and ScanNet (Dai et al., 2017a), with sensor size s_S. In this case, training with both sensor sizes (by weight sharing) decreased the performance. However, resizing reduced the error to the level of the same-camera baselines. The reason for this is the completely different distribution of the two datasets, with null intersection of

Test	**Train**	abs.rel	l1.inv	rmse	sc.inv
		: 1	$1/m$	m	$lg(m)$
$s_1\mathscr{U}f_{72}f_{128}$	$s_1\mathscr{U}f_{72}f_{128}$	0.189	0.15	0.46	0.037
	CAM-C‡	**0.175**	**0.144**	**0.433**	**0.0312**
$s_2\mathscr{U}f_{72}f_{128}$	$s_2\mathscr{U}f_{72}f_{128}$	0.166	0.133	0.412	0.0323
	CAM-C‡	**0.158**	**0.131**	**0.39**	**0.0265**
$s_3\mathscr{U}f_{72}f_{128}$	$s_3\mathscr{U}f_{72}f_{128}$	0.174	0.14	0.425	0.0336
	$s_1\mathscr{U}f_{72}f_{128}$	0.184	0.143	0.44	0.0357
	$s_2\mathscr{U}f_{72}f_{128}$	0.177	0.145	0.435	0.0356
	$s_1s_2\mathscr{U}f_{72}f_{128}$	0.178	0.143	0.451	0.0365
	CAM-C‡	**0.164**	**0.134**	**0.402**	**0.0283**
s_5f_{64}	s_5f_{64}	**0.163**	**0.227**	0.309	**0.0356**
	s_1f_{64}	0.245	0.292	0.337	0.0598
	$s_1s_2\mathscr{U}f_{72}f_{128}$	0.369	0.369	0.44	0.0427
	CAM-C‡	0.177	0.236	**0.289**	0.0362
		smaller is better			

‡ Trained with weight sharing in sensor sizes s_1, s_2 and $\mathscr{U}f_{72}f_{128}$.

Table 2.8 Camera parameter generalization with EquiConvs. Results on training and testing on different cameras. 1st column: camera parameters for test set. 2nd column: camera parameters seen during training. This is a continuation of Table 2.6. Notice how the network with EquiConvs is the only model that generalize getting better performance than the same-camera baseline on most test sets.

visual features (*e.g.* there are no chairs on KITTI and no cars on ScanNet). This is, however, a very particular case, resizing degrades significantly the accuracy in general.

2.5.4 Robust Generalization with CAM-Convs

In this experiment we show that CAM-Convs generalize to different camera models. In order to evaluate the influence of CAM-Convs we trained our model with two different sensor sizes (s_1 and s_2) and weight sharing. Focal length during training is sampled randomly from a uniform distribution $\mathscr{U}f_{72}f_{128}$. We evaluated the trained model in four different test sets, see Table 2.8. The first two include the camera model the network was trained with, the third has a sensor size unseen during training, and the last (s_5f_{64}) was generated from a camera completely different from the training ones with bigger sensor size and smaller focal length. This case augments considerably the context–*e.g* field of view–which proved to be the hardest case in previous experiments (see network trained with s_1f_{128} in Table 2.6).

CAM-Convs generalize over camera intrinsics, outperforming the same-camera baseline. Results on the test sets $s_1\mathscr{U}f_{72}f_{128}$ and $s_2\mathscr{U}f_{72}f_{128}$ in Table 2.8 show that the network

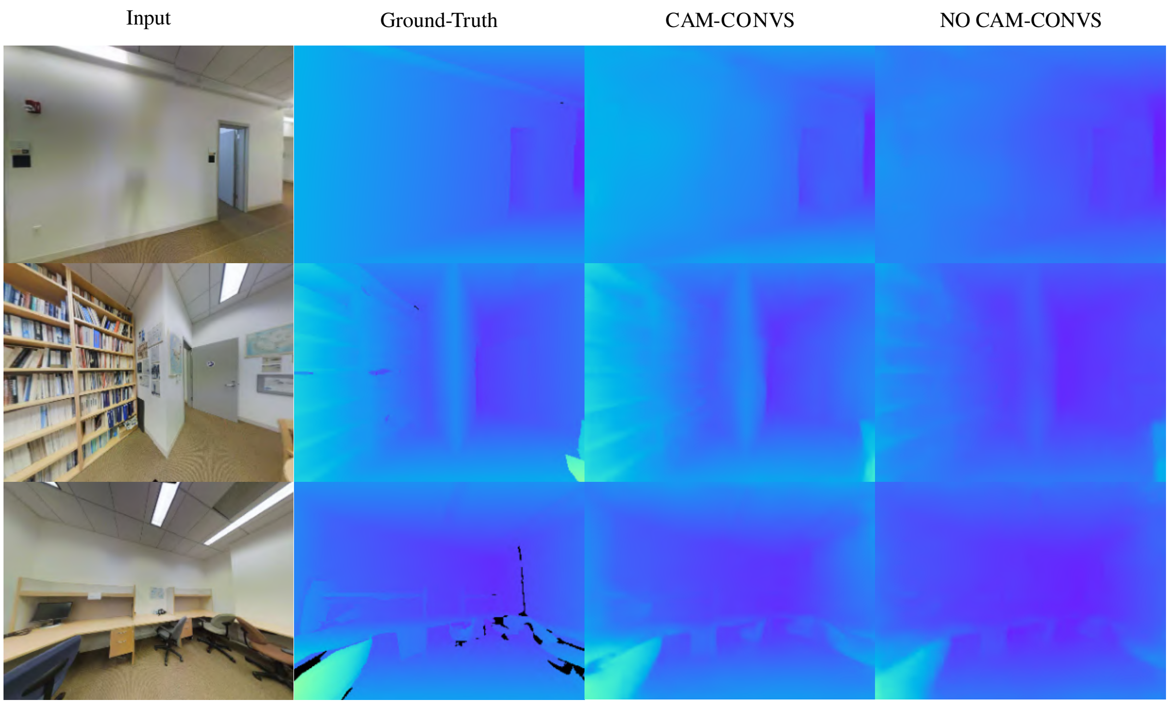


Fig. 2.6 Qualitative results for the test set $s_5 f_{64}$. **1st column**: RGB input. **2nd column:** Ground truth depth. **3rd column:** Prediction with our network using CAM-Convs trained on $s_1 s_2 \mathscr{U} f_{72} f_{128}$. **4th column:** Prediction of a network trained without CAM-Convs. Notice that the test camera parameters are significantly different from the training set and images have a much wider field of view. Despite the large difference in the camera parameters the network with CAM-Convs produces sharp depth maps on which room corners are clearly visible.

with CAM-Convs trained on images of two sizes clearly outperforms the baselines, which was trained on the exact test size. The addition of CAM-Convs allowed the network to learn the dependence of the image features from the calibration parameters.

CAM-Convs generalize to sensor sizes unseen during training. Remarkably, the network with CAM-Convs also outperforms the same-camera baseline on the test set with sensor size s_3 (third test set in Table 2.8), which is not included in the training data. Further, it generalizes better than a network trained on the exact same conditions but without CAM-Convs (see $s_1 s_2 \mathscr{U} f_{72} f_{128}$ in the table).

CAM-Convs generalize to cameras unseen during training. With the last test set ($s_5 f_{64}$) in Table 2.8 we evaluate our network on an extreme case of camera parameters with a very wide field of view and very different sensor size from the training ones. Table 2.8 shows that CAM-Convs improve considerably the generalization to new unseen cameras over the naïve approaches. Figure 2.6 shows a qualitative comparison between our network with CAM-Convs and the network without CAM-Convs ($s_1 s_2 \mathscr{U} f_{72} f_{128}$) in the test set $s_5 f_{64}$.

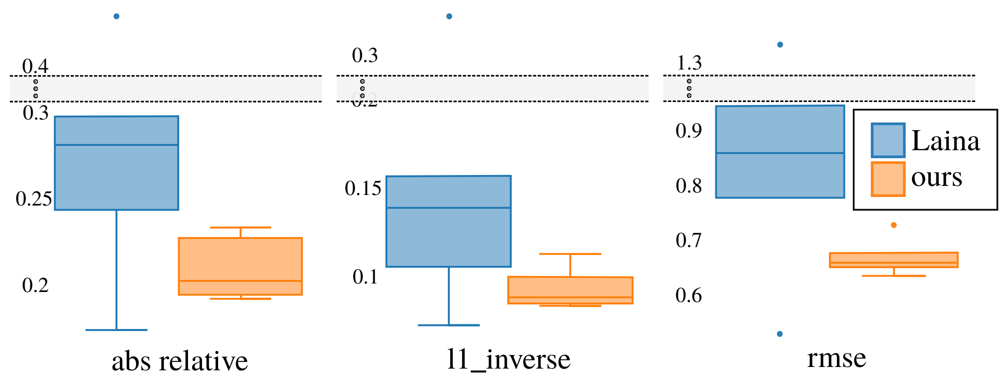


Fig. 2.7 Error distribution on the test set of NYUv2 with 6 different camera parameters. In orange, our network with CAM-Convs, trained on several datasets not including NYUv2. In blue, Laina et al. (2016), trained on NYUv2.

2.5.5 Experiments on Multiple Datasets

In our last experiment we demonstrate how CAM-Convs can generalize across datasests by training on four datasets with different cameras (KITTI (Uhrig et al., 2017), ScanNet(Dai et al., 2017a), MegaDepth (Li and Snavely, 2018) and Sun3D(Xiao et al., 2013) and testing on a different one (NYUv2 (Silberman et al., 2012)).

Training: We trained our network for three different sensor sizes (320×320, 256×256 and 224×224) using weight sharing. We augmented the training data by scaling the images and shifting the principal point to increase the variation of the camera parameters and then crop to image to one of the target sensor sizes. We did not use focal length normalization in this experiment, as we cannot ensure constant pixel size across datasets. As MegaDepth has only up-to-scale ground truth, we applied only scale-invariant losses and added the scale-invariant cost function of Eigen et al. (2014). The same network *without* CAM-Convs, and hence with no camera information, did not converge during training. The lack of calibration information creates inconsistencies (*e.g.* same-size objects may have different depths due to different focal lengths).

Testing: We evaluated our network on the official test set of NYUv2 and compared against the state of the art (Laina et al., 2016) (similar network without CAM-Convs) . Note that the network of Laina et al. (2016) was trained exclusively on NYUv2, while our network was trained on a set of datasets excluding NYUv2 with different cameras and data distributions (some of the datasets are outdoors, see Figure 2.9 and Figure 2.10). This is important since our model cannot benefit from the dataset bias (Torralba and Efros, 2011). We predicted depths for images from 6 different cameras: the original camera of the NYUv2 dataset and 5 simulated ones by cropping (to shift principal point and reduce sensor size) and resizing (to change focal length).

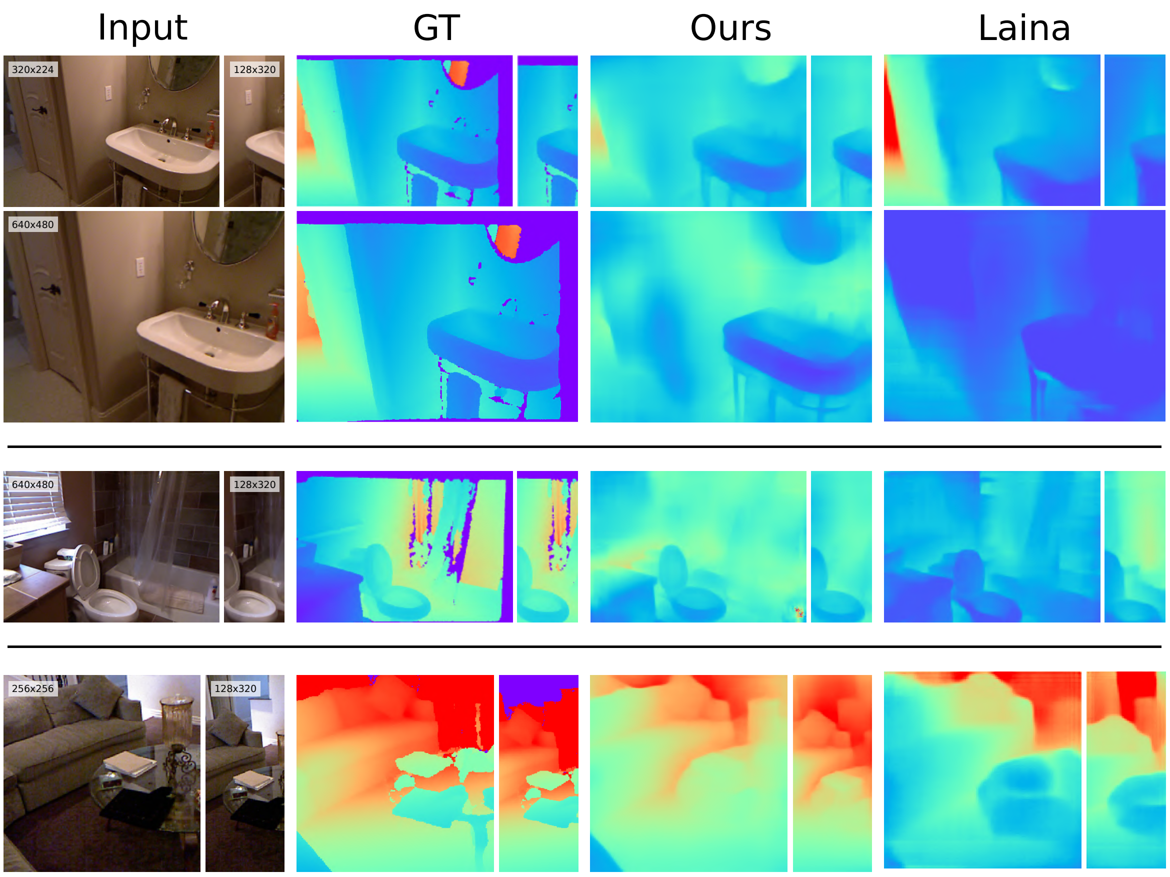


Fig. 2.8 Qualitative results, NYUv2 test set with intrinsics variations. **1st column**: Input RGB images. Each row shows the original one and scaled and cropped versions **2nd column**: Depth groundtruth. **3rd column**: Prediction from our network with CAM-Convs, trained on several datasets *NOT* including NYUv2. Our network produces consistent depth close to the ground truth for all images. **4th column**: Laina et al. (2016), trained exclusively on NYUv2. Its errors are low on the training resolution but does not generalize to new intrinsics.

Figure 2.7 shows the distribution of the mean error of the usual metrics obtained for the 6 different cameras. Since Laina et al. (2016) was trained on the NYUv2 dataset, it works slightly better when it predicts the images from the camera it was trained on (the point with the smallest error). However, performance degrades when the camera changes and CAM-Convs have always smaller error and variance. Figure 2.8 illustrate how CAM-Convs depth predictions are stable for different cameras, while predictions of Laina et al. (2016) vary significantly. Recall that CAM-Convs were not trained on NYUv2, which indicates that they are able to generalize over different camera models and outperform Laina et al. (2016) although they trained on the same dataset.

Figures 2.8, 2.9 and 2.10 show depth predictions for images (and cropped/resized versions) from the NYUv2, KITTI and MegaDepth test sets. Again, note the excellent performance across datasets with different data distributions and camera intrinsics. All predictions

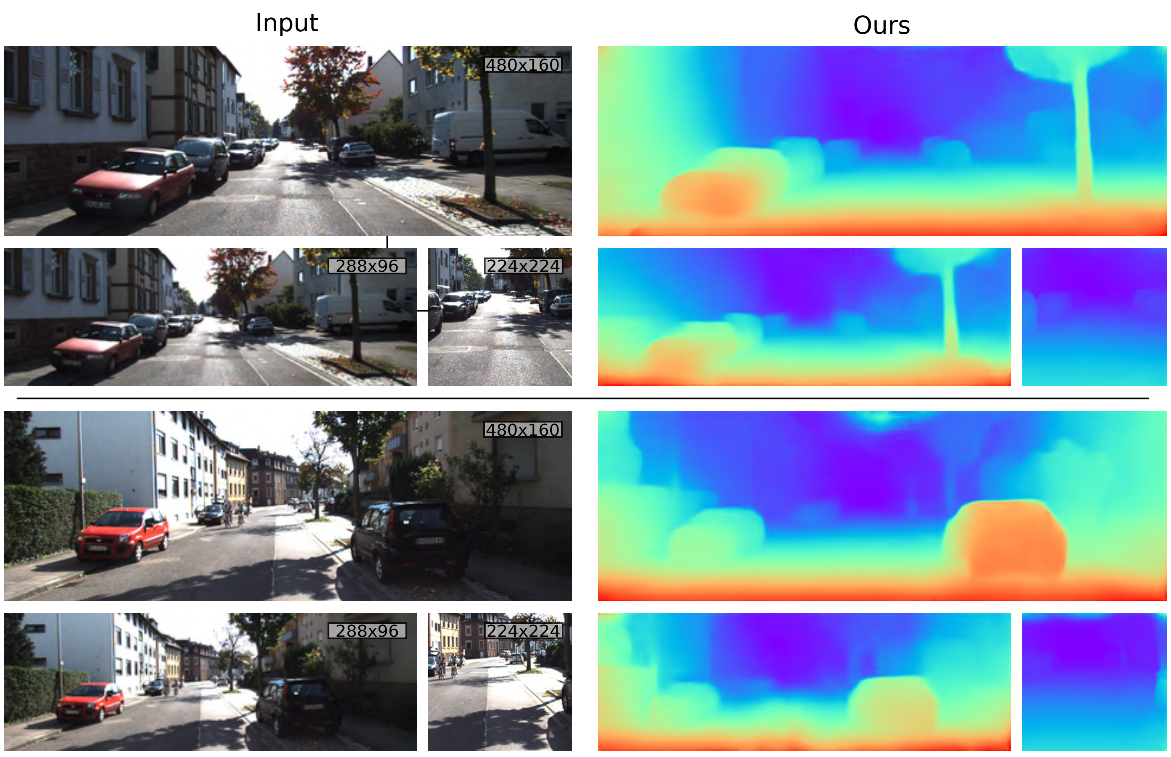


Fig. 2.9 Qualitative results on the KITTI validation set. **1st column:** Input RGB images. Each row shows the original one and scaled and cropped versions. **2nd column:** Prediction from our network.

were done with the exact same network without further fine-tuning to a particular dataset or camera parameters.

2.6 Conclusions

This chapter introduces CAM-Convs, a novel type of convolution that allows depth prediction networks to be camera-independent. Experimental results show that current networks overfit to the training camera model resulting on: 1) a lack of generalization to images from other cameras and 2) degraded performance when trained with images from different cameras. CAM-Convs learn how to use the camera intrinsics jointly with the image features to predict depth; solving both limitations. They maintain prediction accuracy for new cameras and better exploit training data from different cameras. The latter is an interesting direction to scale up systems that depend on camera parameters.

After this work was published, López-Antequera et al. (2020) have published a dataset with over 750 thousand images with scaled depth taken by heterogeneous cameras and successfully tested CAM-Convs and alternatively proposed their own method. Despite being

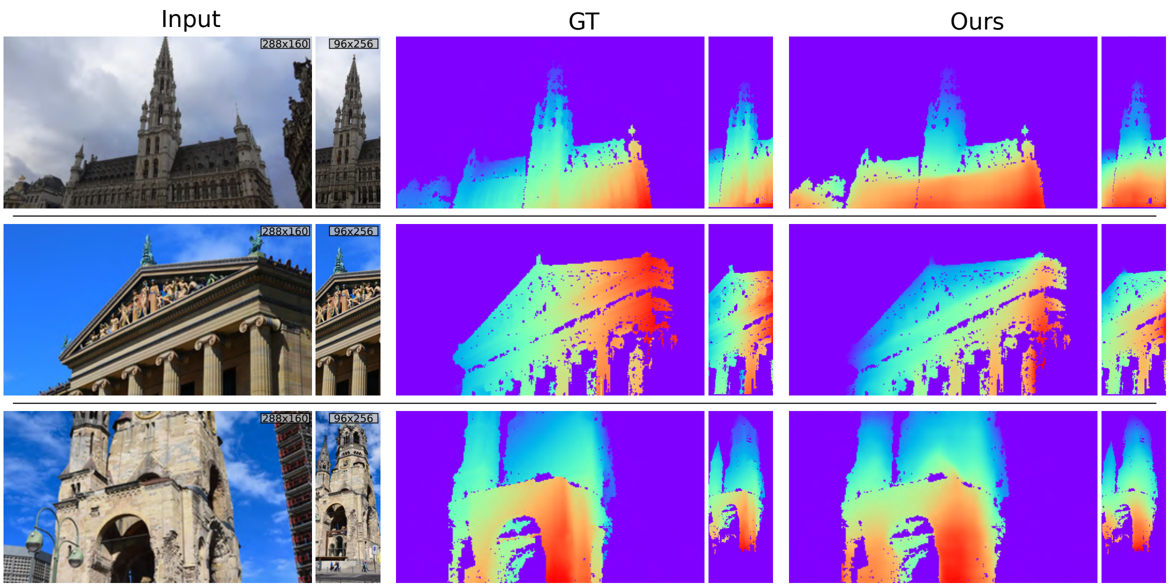


Fig. 2.10 Qualitative results on MegaDepth test set. **1st column:** Input RGB images. Each row shows the original one and scaled and cropped versions. **2nd column:** Depth groundtruth. **3rd column:** Prediction from our network. The predictions are masked as the groundtruth to facilitate visualization.

an stand-alone approach, it seems that their alternative method also benefits of the addition of CAM-Convs. We believe that this dataset it is going to be a very relevant benchmark to push forward progress in this research area.

Chapter 3

Combining Single-View Deep Learning Depth with Multi-View Depth

As introduced in the first chapter of this thesis, traditional methods for Visual SLAM are quite dependent on the motion of the camera, to accurately triangulate points in the scene estimate the camera translation, and also on the texture on the points to successfully estimate is 3D position in the space.

Dense and accurate 3D mapping from a monocular sequence is a key technology for several applications and still an open research area. The work presented in this chapter leverages the results on single-view CNN-based depth estimation by Eigen and Fergus (2015) and fuses them with multi-view depth estimation by Concha and Civera (2015b). Both approaches present complementary strengths. Multi-view depth is highly accurate but only in high-texture areas and high-parallax cases. Single-view depth captures the local structure of mid-level regions, including texture-less areas, but the estimated depth lacks global coherence. The single and multi-view fusion we propose is challenging in several aspects. First, both depths are related by a deformation that depends on the image content. Second, the selection of multi-view points of high accuracy might be difficult for low-parallax configurations. We present contributions for both problems. Our results in the public datasets of NYUv2 and TUM shows that our algorithm outperforms the individual single and multi-view approaches. A video showing the key aspects of mapping in our Single and Multi-view depth proposal is available at `https://youtu.be/ipc5HukTb4k`.

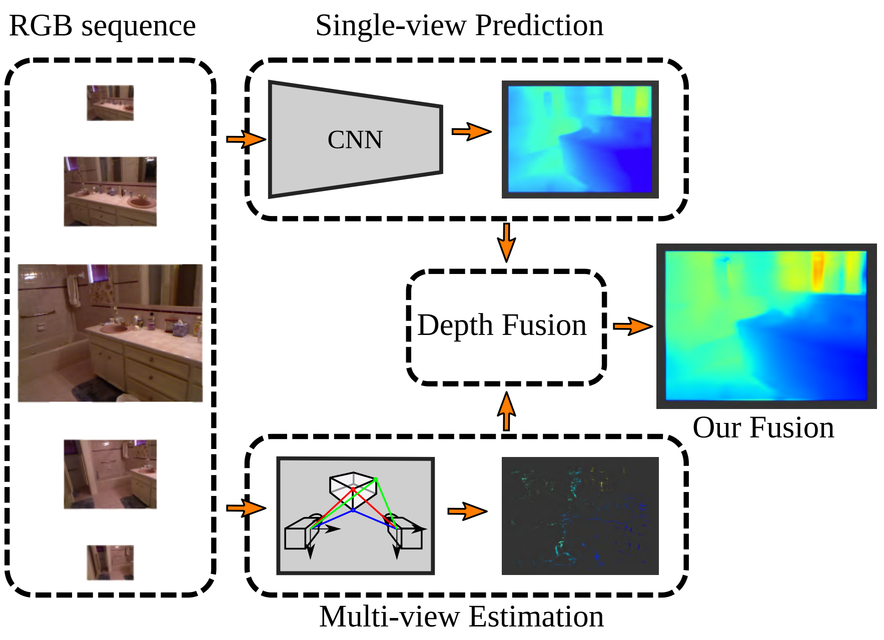


Fig. 3.1 Overview of our proposal. The input is a set of overlapping monocular views. The learning-based single-view and geometry-based multi-view depth are fused, outperforming both of them. All the depth images are color-normalized for better comparison. This figure is best viewed in color.

3.1 Introduction

Estimating an online, accurate and dense 3D scene reconstruction from a general monocular sequence is one of the fundamental research problems in computer vision. The problem has nowadays a high relevance, as it is a key technology in several emerging application markets (augmented and virtual reality, autonomous cars and robotics in general). The state of the art are the so-called direct mapping methods (Newcombe et al., 2011), that estimate an image depth by minimizing a regularized cost function based on the photometric error between corresponding pixels in several views. The accuracy of the multi-view depth estimation depends mainly on three factors: 1) The geometric configuration, with lower accuracies for low-parallax configurations; 2) the quality of the correspondences among views, that can only be reliably estimated for high-gradient pixels; and 3) the regularization function, typically the Total Variation norm, that is inaccurate for large texture-less areas. Due to this poor performance on large low-gradient areas, semi-dense maps are sometimes estimated only in high-gradient image pixels for visual direct SLAM (Engel et al., 2014). Such semi-dense maps are accurate in high-parallax configurations but not a complete model of the viewed scene. Low-parallax configurations are mostly ignored in the visual SLAM literature.

An alternative method is single-view depth estimation, which has recently experienced a qualitative improvement in its accuracy thanks to the use of deep convolutional networks (Eigen and Fergus, 2015). Their accuracy is still lower than that of multi-view methods for high-texture and high-parallax points. But, as we will argue in this chaper, they improve the accuracy of multi-view methods in low-texture areas due to the high-level feature extraction done by the deep networks –opposed to the low-level high-gradient pixels used by the multi-view methods. Interestingly, the errors in the estimated depth seem to be locally and not globally correlated since they come from the deep learning features.

The main idea of this work is to exploit the information of single and multi-view depth maps to obtain an improved depth even in low-parallax sequences and in low-gradient areas. Our contribution is an algorithm that fuses these complementary depth estimations. There are two main challenges in this task. First, the error distribution of the single-view estimation has several local modes, as it depends on the image content and not on the geometric configuration. Single and multi-view depth are hence related by a content-dependent deformation. Secondly, modeling the multi-view accuracy is not trivial when addressing general cases, including high and low-parallax configurations.

We propose a method based on a weighted interpolation of the single-view local structure based on the quality and influence area of the multi-view semi-dense depth and evaluate its performance in two public datasets –NYU and TUM. The results show that our fusion algorithm improves over both individual single and multi-view approaches.

The rest of the chapter is organized as follows. Section 3.2 describes the most relevant related work. Section 3.3 motivates and details the proposed algorithm for single and multi-view fusion. Section 3.4 presents our experimental results and, finally, Section 3.5 contains the conclusions of this work.

3.2 Related Work

We classify the related work for dense depth estimation into two categories: methods based in multiple views of the scene and those which predict depth from one single image.

3.2.1 Multi-View Depth

In the multi-view depth estimation, Newcombe et al. (2011), Graber et al. (2011) and Stühmer et al. (2010) are the first works that achieved dense and real-time reconstructions from monocular sequences. Some of the most relevant aspects are the direct minimization of the photo-metric error –instead of the traditional geometric error of sparse reconstructions–

and the regularization of the multi-view estimation by adding the total variation (TV) norm to the cost function.

TV regularization has low accuracy for large textureless areas, as shown by Concha et al. (2014), Pinies et al. (2015), Piniés et al. (2015) among others. In order to overcome this Concha et al. (2014) propose a piecewise-planar regularization; the plane parameters coming from multi-view superpixel triangulation (Concha and Civera, 2014) or layout estimation (Hedau et al., 2009b). Pinies et al. (2015) propose higher-order regularization terms that enforce piecewise affine constraints even in separated pixels. Piniés et al. (2015) selects the best regularization function among a set using sparse laser data. Building upon Concha et al. (2014), Concha et al. (2015) adds the sparse data-driven 3D primitives of Fouhey et al. (2013) as a regularization prior. Compared to these works, our fusion is the first one where the information added to the multi-view depth is fully dense, data-driven and single-view; and hence it does not rely on additional sensors, parallax or Manhattan and piecewise-planar assumptions. It only relies on the network capabilities for the current domain, assuming that the test data follows the same distribution that the data used for training.

Due to the difficulty of estimating an accurate and fully dense map from monocular views there are several approaches that estimate only the depth for the highest-gradient pixels (Engel et al., 2014). While this approach produces maps of higher density than the more traditional feature-based ones (Mur-Artal et al., 2015), they are still incomplete models of the scene and hence their applicability might be more limited.

3.2.2 Single-View Depth

For a more detailed and updated revision of the state of the art on single-view depth estimation we refer the reader to the Chapter 2 of this thesis that specifically addresses this problem. In this Chapter we discuss the original literature consider for this research.

Depth can be estimated from a single view using different image cues, for example focus (Ens and Lawrence, 1993) or perspective (Sturm and Maybank, 1999). Learning-based approaches, as the one we use, basically discover RGB patterns that are relevant for accurate depth regression.

The pioneering work of Saxena et al. (2009) trained a MRF to model depth from a set of global and local image features. Before that, Saxena et al. (2007) presented an early approach to depth prediction from monocular and stereo cues. Eigen et al. (2014) presented a two deep convolutional neural network (CNN) stacked, one to predict global depth an the second one that refines it locally. Build upon this method, Eigen and Fergus (2015) recently presented a three scale convolutional network to estimate depth, surface normals and semantic labeling.

	High-Gradient	Low-Gradient
Multi-View	0.18	1.02
Single-View	0.36	0.42

Table 3.1 Median depth error [m] for single and multi-view depth estimation, and high and low-gradient pixels. This evaluation has been done in the sequence *living_room_0030a* from the NYUv2 dataset (one of the sequences with higher parallax). The normalized threshold between high and low-gradient pixels is 0.35 (gray scale).

Liu et al. (2015b) use a unified continuous CRF-and-CNN framework to estimate depth. The CNN is used to learn the unary and pairwise potentials that the CRF uses for depth prediction.

Based on Eigen and Fergus (2015), Li et al. (2016) incorporates mid-level features in its prediction using *skip-layers*. It shows competitive results and a small batch-size training strategy that makes their network faster to train. Chakrabarti et al. (2016) introduces a different method to predict depth from single-view using deep neural networks, showing that training the network with a much richer output improves the accuracy. Cao et al. (2016) formulates the depth prediction as a classification problem and the net output is a pixel-wise distribution over a discrete depth range. Finally, Godard et al. (2017) presents an unsupervised network for depth prediction using stereo images.

3.3 Single and Multi-View Depth Fusion

State-of-the-art multi-view techniques have a strong dependency on high-parallax motion and heterogeneous-texture scenes. Only a reduced set of salient pixels that hold both constraints has a small error, and the error for the majority of the points is large and uncorrelated. In contrast, single-view methods based on CNN networks achieve reasonable errors in all the image but they are locally correlated. Our proposal exploits the best properties of these two methods. Specifically, it uses a deep convolutional network (CNN) to produce rough depth maps and fuses their structure with the results of a semi-dense multi-view depth method (Fig. 3.1).

Before delving into the technical aspects, we will motivate our proposal with some illustrative results. Table 3.1 shows the median depth error of the high-gradient and low-gradient pixels for a multi-view and single view reconstruction using a medium/high-parallax sequence of the NYUv2 dataset. For the multi-view reconstruction, the error for the low-gradient pixels increases by a factor of 2. Notice that the opposite happens for the single-view reconstruction: the error of high-gradient pixels is the one increasing by a factor of 2. For

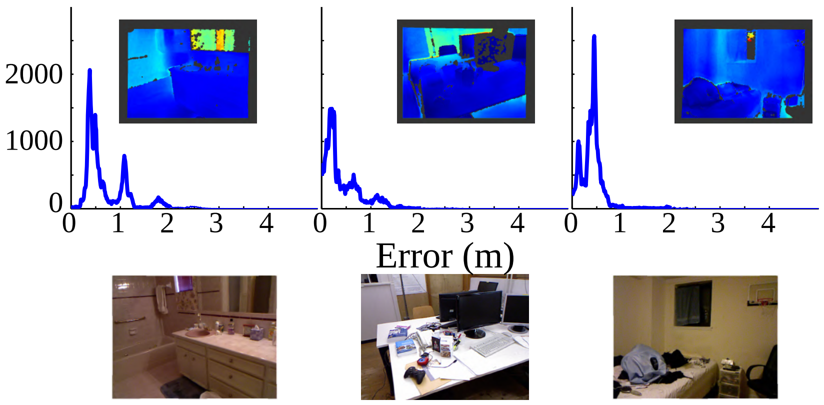


Fig. 3.2 Histogram of single-view depth error [m] for three sample sequences. Notice the multiple modes, each one corresponding to a local image structure, this can be seen in the error images in the top row of the figure.

this experiment, the threshold used to distinguish between high and low-gradient pixels is 0.35 in gray scale (where the maximum gradient would be 1).

Furthermore, the single-view depth error usually has a structure that indicates the presence of local correlations. For instance, Fig. 3.2 shows the histogram of the single-view depth estimation error for three different sequences (two of the NYUv2 dataset and one of the TUM dataset). Notice that the error distribution is grouped in different modes, each one corresponding to an image segment.

This effect is caused by the use of the high-level image features of the latest layers of the CNN network, that extend over dozens of pixels in the original image and hence over homogeneous texture areas. The different nature of the errors can be exploited to outperform both individual estimations. This fusion, however, cannot be naïvely implemented with a simple global model as it requires content-based deformations.

In the next subsections we detail the specific multi and single-view methods that we use in this work and our fusion algorithm.

3.3.1 Multi-view Depth

For the estimation of the multi-view depth we adopt a direct approach (Engel et al., 2014), that allows us to estimate a dense or semi-dense map in contrast to the more sparse maps of the feature-based approaches. In order to estimate the depth of a keyframe $\mathscr{I}_k$ we first select a set of n overlapping frames $\{\mathscr{I}_1, \ldots, \mathscr{I}_o, \ldots, \mathscr{I}_n\}$ from the monocular sequence. After that, every pixel x_l^k of the reference image $\mathscr{I}_k$ is first backprojected at an inverse depth ρ and projected again in every overlapping image $\mathscr{I}_o$.

$$x_l^o = T_{ko}(x_l^k, \rho_l) = KR_{ko}^\top \left(\left(\begin{array}{c} \frac{K^{-1}x_l^k}{||K^{-1}x_l^k||} \\ \rho_l \end{array} \right) - t_{ko} \right), \tag{3.1}$$

where T_{ko},R_{ko} and t_{ko} are respectively the relative transformation, rotation and translation between the keyframe $\mathscr{I}_k$ and every overlapping frame $\mathscr{I}_o$. K is the camera internal calibration matrix.

We define the total photo-metric error $C(\rho)$ as the summation of every photo-metric error ε_l between every pixel (or every high-gradient pixel if we want a semi-dense map) x_l^k in the reference image $\mathscr{I}_k$ and its corresponding one x_l^o in every other overlapping image $\mathscr{I}_o$ at an hypothesized inverse depth ρ_l,

$$C(\rho) \ = \ \frac{1}{n} \sum_{o=1, o \neq k}^{n} \sum_{l=1}^{t} \varepsilon_l(\mathscr{I}_k, \mathscr{I}_o, x_l^k, \rho_l). \tag{3.2}$$

The error $\varepsilon_l(\mathscr{I}_k, \mathscr{I}_o, x_l^k, \rho_l)$ for each individual pixel x_l^k is the difference between the photometric values of the pixel and its corresponding one

$$\varepsilon_l(\mathscr{I}_k, \mathscr{I}_o, x_l^k, \rho_l) \ = \ \mathscr{I}_k(x_l^k) - \mathscr{I}_o(x_l^o). \tag{3.3}$$

The estimated depth for every pixel $\hat{\rho} = (\hat{\rho}_1 \ \dots \ \hat{\rho}_l \ \dots \ \hat{\rho}_t \ )^\top$ is obtained by the minimization of the total photometric error $C(\rho)$:

$$\hat{\rho} \ = \ \arg\min_{\rho} C(\rho) \tag{3.4}$$

3.3.2 Single-view Depth

For single-view depth estimation we use the Deep Convolutional Neural Network presented by Eigen and Fergus (2015). This network uses three stacked CNN to process the images in three different scales. The input to the network is the RGB keyframe $\mathscr{I}_k$. As we use the network structure and parameters released by the authors without further training, our input image size is 320×240. The output of the network is the predicted depth, that we will denote as s. The size of the output is 147×109, that we upsample in our pipeline in order to fuse it with the multi-view depth.

The first scale CNN extract high-level features tuned for depth estimation. This CNN produces 64 feature maps of size 19×14 that are the input, along with the RGB image, of the second scale CNN. This second stacked CNN refines the output of the first one with mid-level features to produce a first coarse depth map of size 74×55. This depth map is upsampled and feeds a third stacked CNN that does a local refinement of the depth. This final step is necessary, as the convolution and pooling steps of the previous layers filter out the high-frequency details.

The first scale was initialized with two different pre-trained networks: the AlexNet (Krizhevsky et al., 2012) and the Oxford VGG (Simonyan and Zisserman, 2014). We use the VGG version, the most accurate one as reported by the authors. This network has been trained in indoor scenes with the NYUDepth v2 dataset (Nathan Silberman and Fergus, 2012). As they used the official train/test splits of the dataset, so do we. We decided to use this neural network because it was the best-performing dense single-view method at the moment we started this work and still it is the one that keeps better trade off between quality and efficiency. We refer the reader to the original work by Eigen and Fergus (2015) for more details on this part of our pipeline.

3.3.3 Depth Fusion

As we mentioned before, the objective is to fuse the output of each previous method while keeping the best properties of each of them: the single-view reliable local structure and the accurate, but semi-dense multi-view depth estimation. Let denote s and m to the single-view depth and the multi-view semi-dense depth estimation, respectively. s is predicted as detailed in section 3.3.2 and $m = \frac{1}{\rho}$ is the inverse of the inverse depth estimated in section 3.3.1.

The fused depth estimation f_{ij} for each pixel (i, j) of a keyframe $\mathscr{I}_k$ is computed as a weighted interpolation of depths over the set of pixels in the multi-view depth image

$$f_{ij} = \sum_{(u,v)\in\Omega} W_{s_{ij}}^{m_{uv}} (m_{uv} + (s_{ij} - s_{uv})), \tag{3.5}$$

where Ω is the semi-dense set of pixels estimated by the multi-view algorithm (e.g. in a high-parallax sequence, they usually correspond with the high-gradient pixels). The interpolation weights $W_{s_{ij}}^{m_{uv}}$ model the likelihood for each pixel $(u, v) \in \Omega$ belonging to the same local structure as pixel (i, j). The interpolation can be interpreted in two ways. First, the depth gradient $(s_{ij} - s_{uv})$ is added to each multi-view depth m_{uv}, i.e. we create depth map for each m_{uv} with the structure of s and then weigh them with pixel based weights. Second, for each depth s_{ij} we modify it according to the weighted discrepancy between $(m_{uv} - s_{uv})$.

The key ingredient of this interpolation are the weights $W_{s_{ij}}^{m_{uv}}$ that model a deformation based on the local image structures. Each weight is computed as the product of four different factors. The first factor

$$\tilde{W}1_{s_{ij}}^{m_{uv}} = e^{\frac{-\sqrt{(i-u)^2+(j-v)^2))}}{\sigma_1}}, \tag{3.6}$$

simply measures proximity based on the distance of the pixels (i, j) and (u, v). The parameter σ_1 controls the radius of proximity for each point. The remainder three factors depend on the structure of the single-view prediction s. The second factor

$$\begin{aligned}\tilde{W}2_{s_{ij}}^{m_{uv}} =& \frac{1}{|\nabla_x s_{uv} - \nabla_x s_{ij}| + \sigma_2} \\ &\cdot \frac{1}{|\nabla_y s_{uv} - \nabla_y s_{ij}| + \sigma_2}\end{aligned} \tag{3.7}$$

measures the similarity of depth gradients and assigns larger weights to similar ones. $\nabla_x s_{ij}$ and $\nabla_y s_{ij}$ represent the depth gradient in the x and y direction respectively at the pixel (i, j). σ_2 limits the influence of a point to avoid extremely high weights for very similar or identical gradients. We set it to 0.1 in the experiments.

Finally, the factors $\tilde{W}3_{s_{ij}}^{m_{uv}}$ and $\tilde{W}4_{s_{ij}}^{m_{uv}}$ strengthen the influence between the points lying in the same plane and are defined as

$$\tilde{W}3_{s_{ij}}^{m_{uv}} = e^{-|(s_{ij}+\nabla_x s_{ij}\cdot(u-i))))-s_{uv}|} + \sigma_3 \tag{3.8}$$

and

$$\tilde{W}4_{s_{ij}}^{m_{uv}} = e^{-|(s_{ij}+\nabla_y s_{ij}\cdot(v-j))))-s_{uv}|} + \sigma_3, \tag{3.9}$$

where σ_3 sets a minimum weight to any point in Ω. This is required to avoid vanishing weights when they are combined with $\tilde{W}1_{s_{ij}}^{m_{uv}}$ and $\tilde{W}2_{s_{ij}}^{m_{uv}}$.

The product of this four factor makes a non-normalized weight for each pixel in Ω

$$\tilde{W}_{m_{uv}}^{s_{ij}} = \prod_{n=1}^{4} \tilde{W}_{n m_{uv}}^{s_{ij}} \tag{3.10}$$

and represents its area of influence. The parameters σ_1, σ_2 and σ_3 shape the area of influence and have to be selected to balance proximity, gradient and planarity and to avoid discontinuities in the result of the fusion. This was done empirically on a small set of three images. The values of the parameters are 15, 0.1 and $1e-3$, respectively, and we kept them fixed for all our experiments.

Fig. 3.3 shows this area for a point on an image and how it is computed. Notice how the influence expands around the point but is kept inside the same local structure (the table). Once all the factors has been computed, since all the pixels (i, j) are influenced by all the

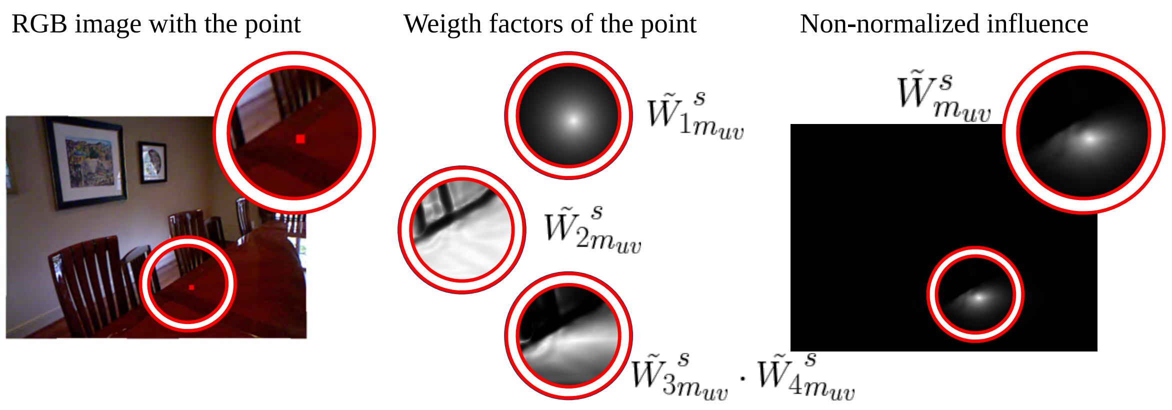

RGB image with the point
Weigth factors of the point
Non-normalized influence
$\tilde{W}_{1m_{uv}}^{s}$
$\tilde{W}_{2m_{uv}}^{s}$
$\tilde{W}_{3m_{uv}}^{s} \cdot \tilde{W}_{4m_{uv}}^{s}$
$\tilde{W}_{m_{uv}}^{s}$

Fig. 3.3 Non-normalized influence of the highlighted red point in the image. *First column:* RGB input image with a red point over the table, this point represent one pixel estimated by the multi-view algorithm. *Second column:* each one of the weights calculated separately, the third and fourth weights are shown as a product for a more intuitive view. *Third column:* Non-normalized influence of the highlighted point in the RGB image. Notice how its influence is cut on the edge of the table. Figure best viewed in electronic format.

pixels in Ω (see Eq. 3.5), we normalize the weights for each single-view pixel so all the weights over a pixel (i,j) sum 1.

$$W_{s_{ij}}^{m_{uv}} = \frac{\tilde{W}_{s_{ij}}^{m_{uv}} - \min_{(g,h)\in\Omega} \tilde{W}_{s_{ij}}^{m_{gh}}}{\sum_{(p,k)\in\Omega} \tilde{W}_{s_{ij}}^{m_{pk}} - \min_{(g,h)\in\Omega} \tilde{W}_{s_{ij}}^{m_{gh}}} \tag{3.11}$$

The normalized weights expand the local influence to the whole image (see Fig. 3.4 and Fig. 3.5 for a more detailed view). Notice how the influence expands along planes even if the points in Ω do not reach the end of the plane; and is sharply reduced when the local structure changes. Once these influence weights have been calculated and normalized, the fusion depth estimation, f, for each point (i,j) is a combination of all the selected points in Ω, as presented in Eq. 3.5.

3.3.4 Multi-view Low-Error Point Selection

Up to now we have assumed that all the points in the multi-view semi-dense depth map Ω have low error. This is easily achievable in high-parallax sequences by using robust estimators –robust cost functions or RANSAC. However, it is problematic for the degenerate or quasi-degenerate low-parallax geometries that we also target in this work. In this case, multi-view depths may contain large errors that will propagate to the fused depth map and it is necessary to filter them out. Unexpectedly, selecting high gradient pixels was not robust enough to remove points with large depth errors and we have developed a two step

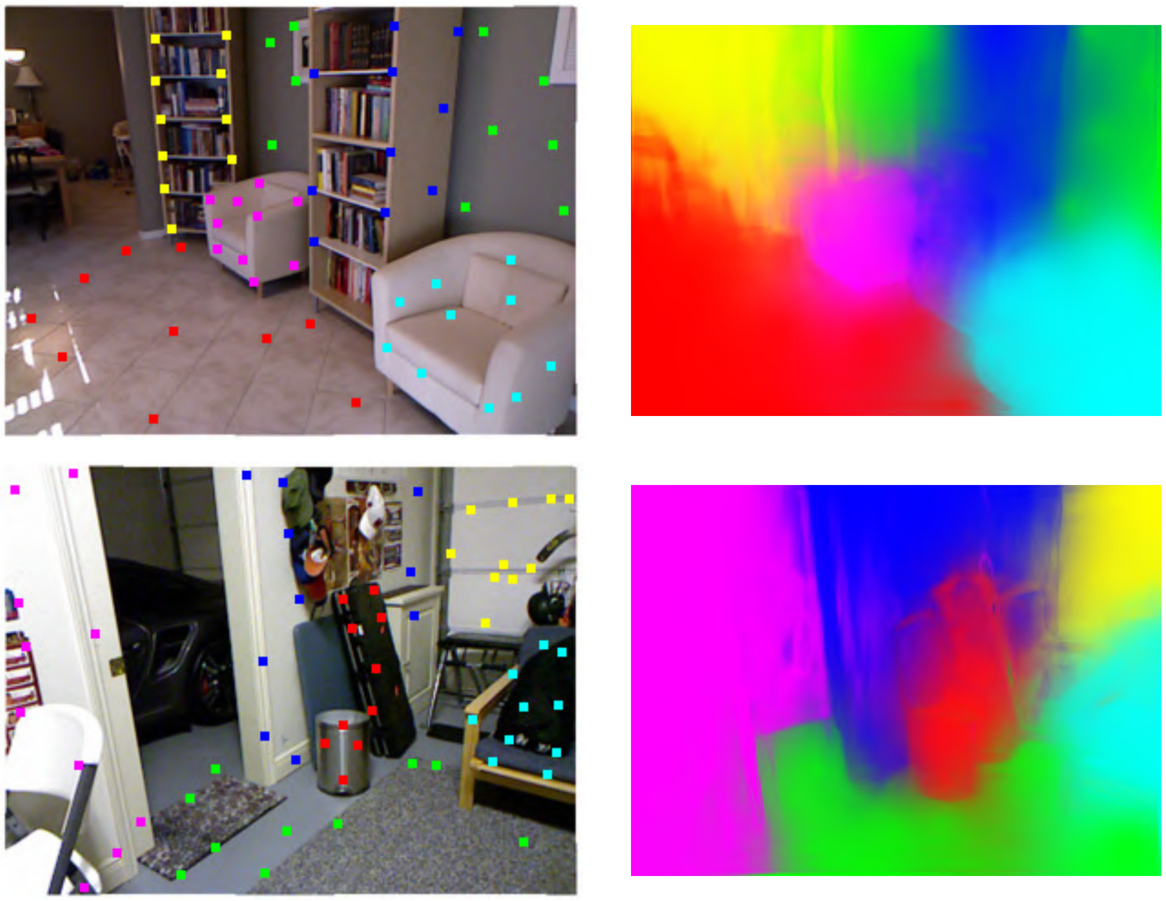


Fig. 3.4 Normalized influence area of the points. Notice how it expands around local structure areas given a set of points in Ω. *First column:* RGB image with the points of Ω labeled with different colors. *Second column:* influence areas computed by our method. Notice how this influence expands in areas with the same local structure but can be misled in areas where there is a lack of points or where the estimation from the neural net is not accurate enough. Figure best viewed in color.

algorithm that takes into account photometric and geometric information in the first step and the single-view depth map in the second one.

The first step selects a fixed percentage of the best correspondence candidates –the best 25% in our experiments– based on the product of a photometric and a geometric scores. On one hand, the photometric criterion focuses on the quality of the correspondences using image information. We apply a modified version of the *second best ratio*.We first extract the two closest matches for a pixel (smallest photometric errors according to Eq. 3.3). We then compute the score as a function of the ratio between the distance of the two descriptors (a high ratio suggesting a good match) and the gradient of the distance function along the epipolar line (i.e., the error function presenting a distinct *V-shape* around this match and

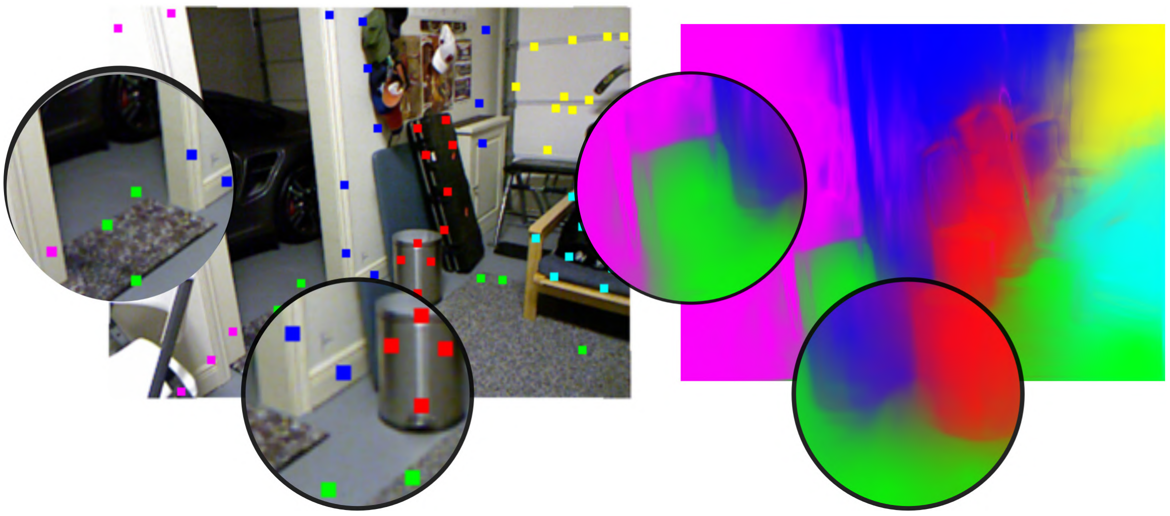


Fig. 3.5 Detail of the influence area. Notice how it expands mainly in the areas with same local structure. Figure best viewed in color.

suggesting spatial accuracy). On the other hand, the geometric score simply backpropagates the image correspondence error to the depth estimation, resulting in low scores for low-parallax correspondences.

In a second stage we also use the structure of the single-view reconstruction and apply RANSAC to estimate a spurious-free linear transformation between the multi and single-view points using only the points pre-filtered in the first stage. We apply this linear model along the entire image, consensus with outliers is found if small patches are used. This reduces further the number of spurious depth values from the multi-view algorithm. The result is a small set of low-error points that we use for the interpolation of the previous section. As mentioned before, in our experiments this algorithm behaves better than a geometric-only compatibility test, especially in the low-parallax sequences of the NYUv2 dataset.

3.4 Experimental Results

In this section we evaluate the algorithm and compare its performance against two state-of-the-art methods: multi-view direct mapping using TV regularization (implemented following Newcombe et al. (2011), Handa et al. (2011)) and the single-view depth estimation using the network of Eigen and Fergus (2015). We have selected two datasets with different properties. The first one is the NYUv2 Depth Dataset (Nathan Silberman and Fergus, 2012), a general dataset aimed at image segmentation evaluation and hence likely to contain low-parallax and low-texture sequences. We analyze results in six sequences from the test set (i.e. the single-view net had not been trained on these sequences) selected just to include different

		RMSE			SCALE INVARIANT			MEAN ERROR (m)			MEAN ERROR (m)
	Sequence	TV	Eigen	Ours(a)	TV	Eigen	Ours(a)	TV	Eigen	Ours(a)	Ours(m)
NYUDepth v2	bath_0018	1.458	0.852	**0.793**	0.405	0.150	**0.145**	1.174	0.692	**0.612**	0.263
	bed_0013	1.004	0.550	**0.482**	0.212	0.139	**0.136**	0.690	0.441	**0.344**	0.163
	dr_0032	2.212	0.710	**0.694**	0.416	0.209	**0.204**	1.797	0.581	**0.554**	0.318
	kit_0032	3.599	1.621	**1.572**	0.812	0.592	**0.583**	2.920	1.222	**1.183**	0.805
	lr_0025	1.073	0.620	**0.597**	0.289	0.236	**0.219**	0.798	0.471	**0.435**	0.289
	lr_0030a	1.031	0.818	**0.792**	0.411	0.228	**0.219**	0.849	0.532	**0.440**	0.329
TUM	fr1_desk	1.581	0.433	**0.410**	0.255	0.121	**0.103**	1.211	0.317	**0.294**	0.154
	fr1_room	1.467	0.323	**0.301**	0.167	0.092	**0.081**	1.163	0.231	**0.207**	0.102

Table 3.2 *Left table:* Error metrics for the NYUv2 and TUM datasets. For each sequence and metric we compare the TV-regularized multi-view depth, the single-view depth Eigen and Fergus (2015) and our fused depth. Ours(a) represent our proposal with the automaic selection of points. *Right table:* Mean error for the fused depth with manual multi-view point selection (Ours(m)); selected points under certain threshold. (The evaluation has been performed in the first 100 frames of each sequence)

types of rooms. The second one is the TUM RGB-D SLAM Dataset (Sturm et al., 2012a), a dataset oriented to visual SLAM and then likely to present a bias benefiting multi-view depth. In this case, we evaluated two sequences selected randomly.

We run our algorithm in a 320×240 subsampled version of the images, as this is the size of the single-view neural network given by the authors. We also run our multi-view depth estimation at this image size, and upsample the fused depth to 640×480 in order to compare it against the ground truth D channel from the kinect camera.

As our aim is to evaluate the accuracy of the depth estimation, we will assume that camera poses are known for the multi-view estimation. In the TUM RGB-D SLAM Dataset (Sturm et al., 2012a) we use the ground truth camera poses. In the NYUv2 Depth Dataset sequences we estimate them using the RGB-D Dense Visual Odometry by Gutiérrez-Gómez et al. (2015). These camera poses will remain fixed and used to create the multi-view depth maps. As mentioned before, the parameters of the fusion algorithm were experimentally set prior to the evaluation on a small separate set of images.

To evaluate the methods, we computed three different metrics, the RMSE, the Mean Absolute Error in meters and the scale invariant error proposed in Eigen et al. (2014) $\frac{1}{n}\sum_i d_i^2 - \frac{1}{n^2}(\sum_i d_i)^2$ where d is $(\log(y) - \log(y^*))$, y and y^* are the ground truth depth and the estimated depth respectively. The results are summarized in Table 3.2. Our method outperforms the TV regularization in both datasets obtaining an average improvement over 50% with respect to the mean of the error in meters. As expected, the TV regularization performs better in the TUM sequences and achieves lower errors, but in terms of improvement there seems not to be big differences between both datasets. Our fusion of depths also outperforms the single-view depth reconstruction, the improvement being 10% on average.

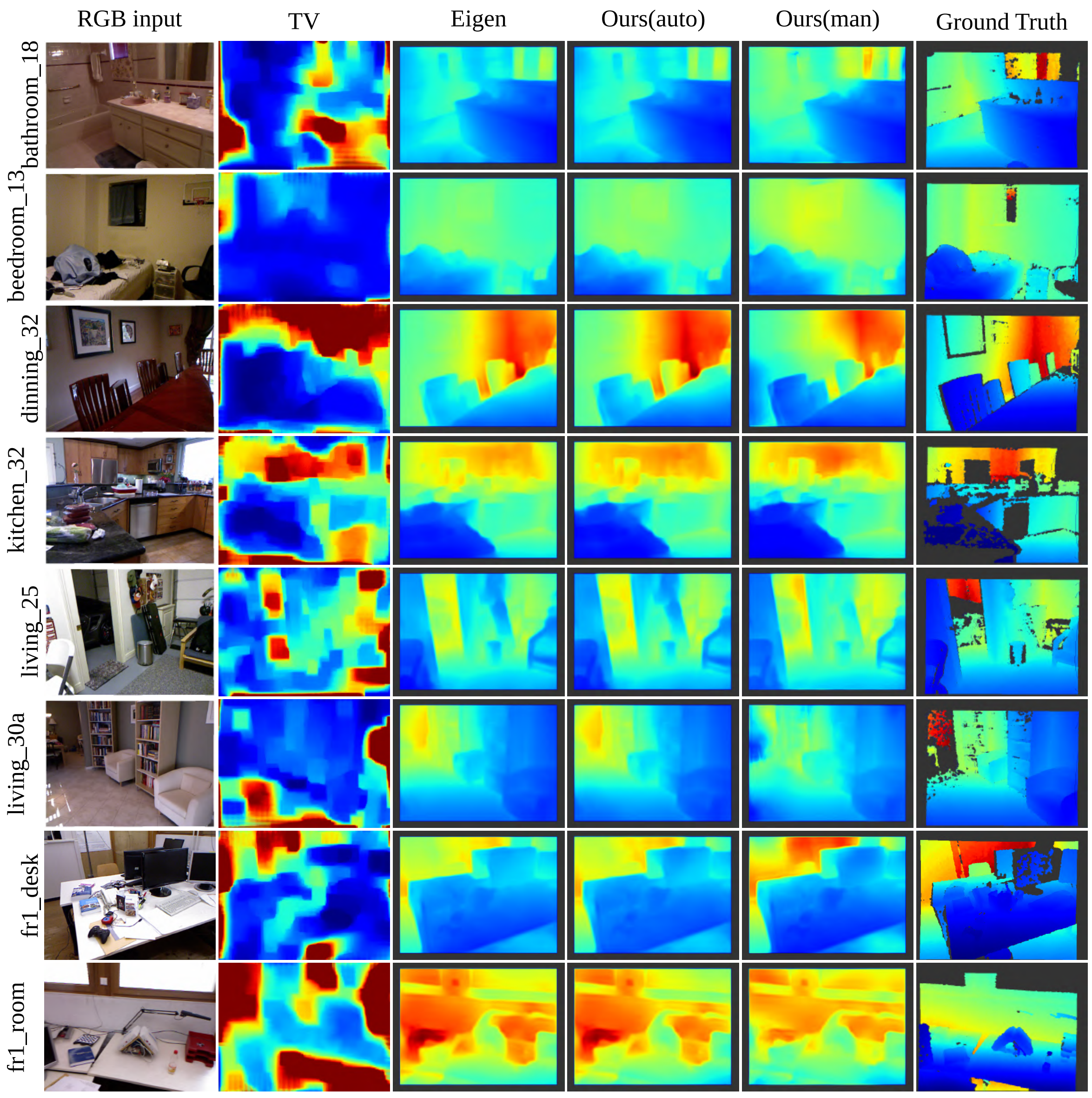


Fig. 3.6 The first six rows are depth images for the NYUDepth v2 dataset Nathan Silberman and Fergus (2012) and the last two rows are for the TUM Dataset Sturm et al. (2012a). Color ranges are row-normalized to facilitate the comparison between different methods. *First column* RGB keyframe, *second column* TV-regularized multi-view depth, *third column* single-view depth, *fourth column* our depth fusion with automatic multi-view point selection, *fifth column* our depth fusion with manual multi-view point selection, and *sixth column* ground truth. Figure best viewed in electronic format.

Both methods perform similarly in both datasets, but except in one sequence, our method is always better or as good as the deep single-view reconstruction. Notice that the improvement does not come exclusively from scale correction; the scale invariant error shows that our method improves the structure estimation in both the single and multi-view cases.

	SCALE INVARIANT			MEAN ERROR (m)		
	W_1	$W_1 \cdot W_2$	$\prod_{i=1}^4 W_i$	W_1	$W_1 \cdot W_2$	$\prod_{i=1}^4 W_i$
NYUv2	0.224	0.216	**0.208**	0.390	0.376	**0.353**
TUM	0.098	0.088	**0.064**	0.145	0.142	**0.128**

Table 3.3 Mean of error metrics for the NYUv2 and TUM datasets. For each sequence and metric we compare the fusion with the only use of the weight W_1, the use of $W_1 \cdot W_2$. and all the weights together.

The right-most colum of Table 3.2 shows the depth errors when the set of multi-view points does not contain outliers. We selected them using the ground-truth data from the D channel, and keeping only those points whose depth error was lower than 10cm. The results are for all sequences better than any method attaining improvements around 70% and 38% with respect to TV and Eigen and Fergus (2015), respectively. Although expected, this result highlights the impact of multi-view outliers and the need for good point selection. It also provides an upper bound and shows that there is still room for improvement in this latest part of our algorithm. In Table 3.3 we show an experiment to better understand the contribution of each weight of our algorithm. For this evaluation we have considered the spurious-free set of multi-view points in order to avoid the influence of noise. It can be seen that using all the weights has an average of 9.8% improvement in mean absolute error with respect to using just W_1 and a 6.5% of improvement with respect to using W_1 and W_2.

Finally, we present the results of some randomly picked images for each sequence of each dataset. Figure 3.6 shows the obtained depth images for the NYUDepth v2 and the TUM datasets. The improvement with respect to the regularized multi-view approach is clear visually since the depth structure is much more consistent. Improvements with respect to single-view images are more subtle and are best viewed by looking at the corresponding depth error images of Figure 3.7. Usually, the improvement comes from a better relative placement of some local structure. For instance, the walls are darker in the error images (see the bathroom_18, bedroom_13 or fr1_desk in Figure 3.7). The effect is more evident when the multi-view points were selected based on the ground truth. This better alignment of local structures reduces the error, as can be seen in the per-sequence error boxplots of Figure 3.8.

3.5 Conclusions

In this chapter we have presented an algorithm for dense depth estimation by fusing 1) the multi-view depth estimation from a direct mapping method, and 2) the single-view depth that comes from a deep convolutional network trained on RGB-D images. Our approach selects a set of the most accurate points from the multi-view reconstruction and fuses them with the dense single-view estimation. It is worth remarking that the single-view depth errors do not depend on the geometric configuration but on the image content and hence the transformation is not geometrically rigid and varies locally. The estimation of this alignment is our main contribution and the most challenging aspect of this research.

Our experiments show that our proposal improves over the state of the art (Eigen and Fergus, 2015) for single-view depth and direct mapping plus TV regularization for multi-view depth). Contrary to other approaches, the single-view depth we use is entirely data-driven and hence does not rely on any scene assumption. As mentioned, we take the network of Eigen and Fergus (2015) as our single-view baseline, because of its availability and its excellent accuracy-cost ratio. However, our fusion algorithm is independent of the specific network and could be used with any of the single-view approaches mentioned in Section 3.2.

Experiments carried out in this research were done in two datasets that were recorded with the same Kinect camera model (Silberman et al., 2012, Sturm et al., 2012a). To extend this work to be reliable with multiple cameras the single-view model would need to use learn how to use the camera model into its advantage. We refer the reader to CAM-Convs presented in Chapter 2 or López-Antequera et al. (2020), since this is actually a previous work it did not count with them.

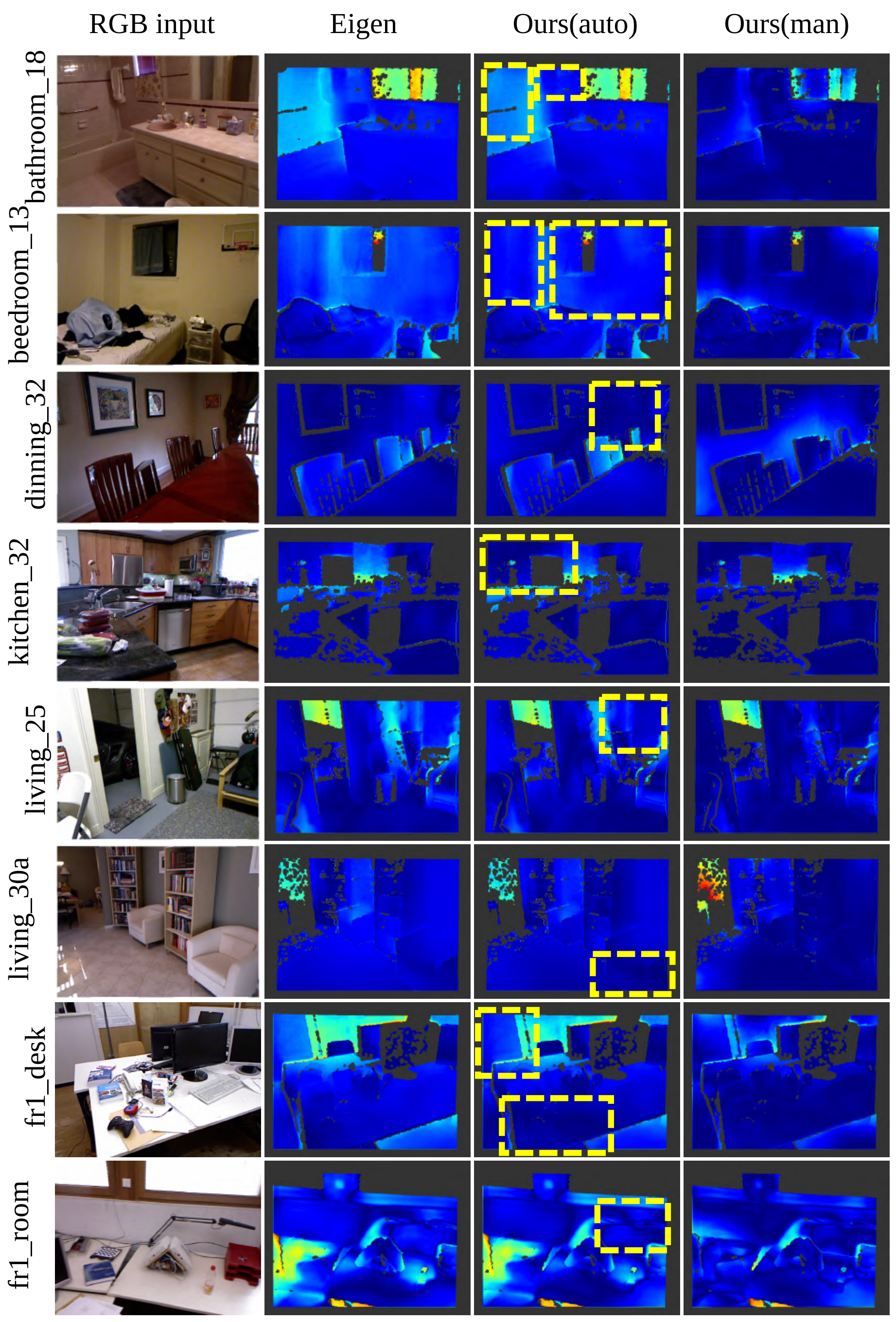


Fig. 3.7 The first six rows are error images (predicted depth - ground truth) for the NYUDepth v2 dataset (Nathan Silberman and Fergus, 2012) and the last two rows are for the TUM Dataset (Sturm et al., 2012a). Color ranges are row-normalized to facilitate the comparison between different methods. Darker blue is better. *First column* RGB keyframe, *second column* single-view depth, *third column* our depth fusion with automatic multi-view point selection, *fourth column* our depth fusion with manual multi-view point selection. In the third column, in yellow, are highlighted the areas where the improvement of our method can be easily appreciated with respect to single-view's error. Figure best viewed in electronic format.

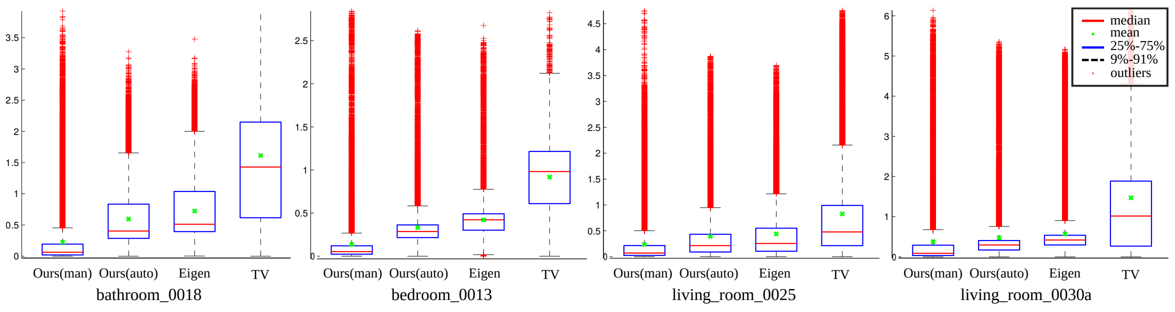


Fig. 3.8 Box-and-Whiskers plots of the pixel error distribution for four of our test scenes. From left to right: Our method with manual point selection, our method with automatic point selection, single-view depth from Eigen and Fergus (2015) and TV-regularized multi-view depth.

Chapter 4

Condition-Invariant Place Recognition

Visual place recognition is particularly challenging when places suffer changes that modify drastically their appearance. Such changes are indeed common, e.g., due to weather, night/day, seasonal features or dynamic content. In this chapter we leverage on place recognition research using deep networks; and explore how it can be improved by exploiting the information from multiple views. Specifically, we propose 3 different alternatives (Descriptor Grouping, Fusion and Recurrent Descriptors) for deep networks to combine visual features of several frames in a sequence; together we present also a single-view network. We show that our approaches produce more compact and better-performing descriptors than state-of-the-art single- and multi-view baselines in two public databases.

4.1 Introduction

Given a dataset of images taken at different places, visual place recognition (Lowry et al., 2016, Arandjelović and Zisserman, 2014, Torii et al., 2013) aims to identify the place of a new query image by associating it to one or several images of the dataset taken in the same location. Recent advances in computer vision have improved the performance of these algorithms, which are currently applied in several different applications such as image retrieval (Noh et al., 2017), mapping and navigation in robotics (Pronobis et al., 2006, Gálvez-López and Tardos, 2012, Lee and Civera, 2019), autonomous driving (McManus et al., 2014) and augmented reality (AR) (Middelberg et al., 2014).

One of the main challenges of visual place recognition is dealing with changes in the appearance of places (Garg et al., 2019). Indeed, place recognition is reasonably robust under small changes in viewpoint and illumination, due to the invariance of local features and rigidity checks (Gálvez-López and Tardos, 2012). But, in contrast, non-rigid scene changes, wide baseline matching and extreme illumination variations are considerably more

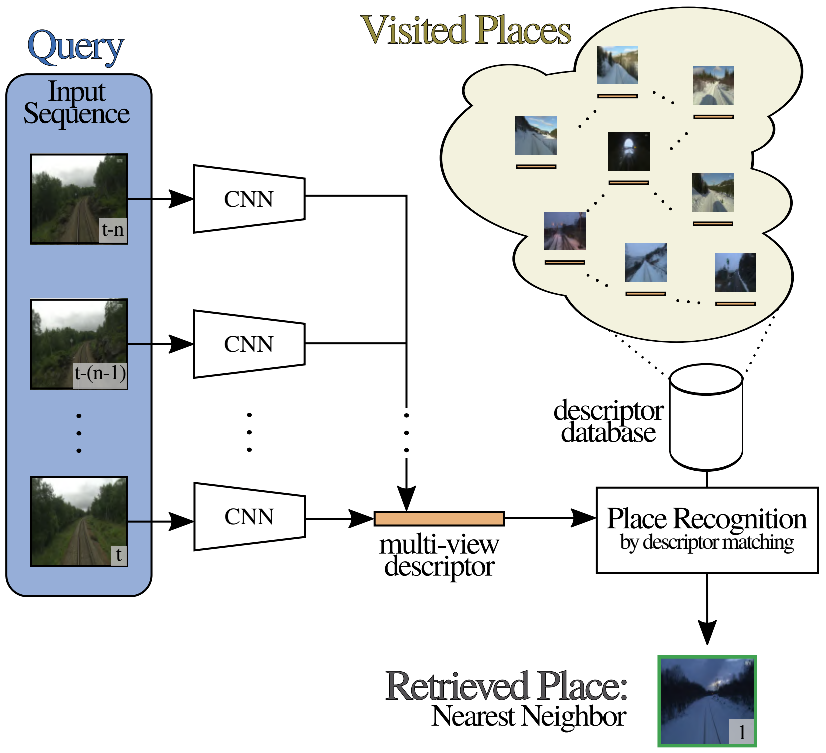


Fig. 4.1 Overview of our proposal. We extract descriptors (using deep networks) for small sequences of *n* frames. We use such descriptors to find the closest match in a database of already visited places.

challenging and result in lower performance. Using multiple frames in a sequence can improve the robustness of place recognition against such changes. But the sequence models proposed by the state of the art (Milford and Wyeth, 2012, Naseer et al., 2018) are handcrafted for a certain set of assumptions (*e.g.* overlapping trajectories, similar velocity patterns), and their performance suffers if these are not hold. Also, typically, they require a high number of frames.

Descriptors directly extracted from CNNs have shown good generalization properties (Gomez-Ojeda et al., 2015), but they usually do not exploit multi-view information. Improvements usually come at the cost of large descriptors, with dimensionality in the order of thousands or hundreds of thousands. The complexity of all place recognition algorithms depends on the size of the descriptor and the number of images in the database, the latest being typically high. This limits the applicability of these techniques in robotics and AR/VR scenarios, in which the computational budget is limited due to real-time constrained loops and limited on-board computational power.

In this thesis we target place recognition in the presence of challenging changes in the condition of an environment, that eventually happen in most of the scenes as time passes.

For example day/night illumination, seasonal and weather changes, or objects that are moved (cars, persons or furniture). We address this problem by generating a global descriptor of the visual input, *i.e.*, every input (one or several images) is encoded as a descriptor and matched versus the rest of the descriptors retrieving the closest place.

Our contribution is the proposal and evaluation of three different deep network architectures that exploit multi-view and temporal information for place recognition: 1) naïve descriptor grouping, 2) learning the fusion of single-view descriptors, and 3) recurrent networks using LSTM (Long Short Term Memory) layers Hochreiter and Schmidhuber (1997). Fig. 4.1 shows an overview of our proposal. Up to our knowledge, ours are the first models that use deep learning to combine multiple views to generate descriptors for place recognition. Encoding temporal information allows us to model short-time relations between images (e.g. short smartphone videos or live photos) without the need of keeping a global map or long image sequences to achieve a high accuracy.

We compare our models to state-of-the-art single-view deep baselines and a non-deep sequential one using two standard datasets: the Partitioned Nordland (Olid et al., 2018) and Alderley (Milford and Wyeth, 2012). The experimental results show that the performance of our three multi-view descriptors outperforms single-view ones. We also outperform SeqSLAM, a state-of-the-art baseline for place recognition from image sequences. Furthermore, our learned descriptors are at least one order of magnitude smaller than those of the state of the art showing that multi-view learning is able to extract relevant information for place recognition.

The rest of the chapter is organized as follows. Section 4.2 presents a partition of the Norland dataset that we have used in this chapter that was published in it is available in our project website [1]. Section 4.3 refers the related work. Section 4.4 details our network architectures, and section 4.5 details how they are trained. Finally, section 4.6 presents the experimental results and section 4.7 the conclusions and lines for future work. Our **code** and a **video** showing our results can be found in our project website [2].

4.2 The Partitioned Nordland Dataset

For this work, we have used the Nordland railroad videos. In 2012, the Norway broadcasting company (NRK) made a documentary about the Nordland Railway, a railway line between the cities of Trondheim and Bodø. They filmed the 729km journey with a camera in the front

[1]https://webdiis.unizar.es/ jmfacil/pr-nordland/

[2]http://webdiis.unizar.es/~jmfacil/cimvpr/

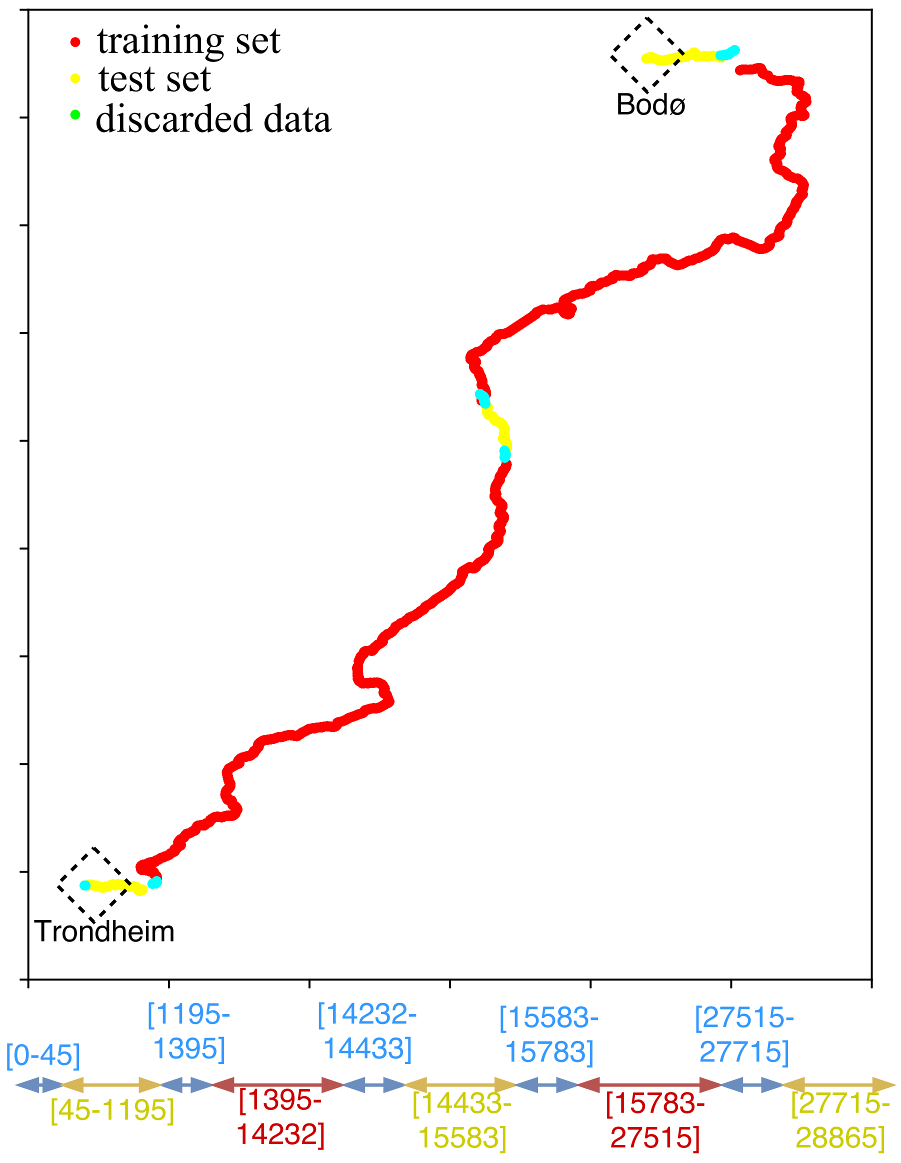


Fig. 4.2 Proposed dataset partition for the Nordland dataset. **Top:** Geographical representation of the training (red) and test (yellow) sets. **Bottom:** Index representation of the distribution, w.r.t. frame index in the videos.

part of the train in winter, spring, fall and summer. The length of each video is about 10 hours and each frame is timestamped with the GPS coordinates.

This dataset has been used by other research groups in place recognition, for example Gomez-Ojeda et al. (2015) and Lowry and Milford (2016). Each group uses different partitions for training and test, making difficult to reproduce the results. In this work we propose a specific partition of the dataset and a baseline, to guarantee a fair comparison between algorithms. We released this dataset partition with the publication of Olid et al. (2018).

4.2.1 Data Pre-processing

The first step, creating the dataset, was to extract the maximum number of images from each video. Moreover, GPS data corruption was fixed and we also eliminated tunnels and stations. After these steps, grabbing one frame per second, we obtained $28,865$ images per video. We used speed information from the GPS data to filter stations and a darkness threshold to filter tunnels.

4.2.2 Dataset Partitions

Fig. 4.2 illustrates the partition of the whole image set in the Nordland dataset. We decided to create the test set with three different sequences of $1,150$ images (a total of $3,450$, yellow in the figure). The rest of the images were used for training ($24,569$, red in the figure). By using multiple sections, the variety of places and appearance changes contained in the test set increases. We also left a separation of a few kilometers between each test and train section by discarding some images in order to guarantee the difference between test and train data.

4.2.3 Place labels

Given the similarity between consecutive images, in this work we propose to consider that two images are of the same place if temporally they are separated by 3 images or less. We applied a sliding window of 5 images over the whole dataset in order to group images taken from five consecutive seconds. This process can be seen in Fig. 4.3.

4.3 Related Work

There have been many works addressing visual place recognition and related problems. For a general overview, we refer the reader to two surveys, Garcia-Fidalgo and Ortiz (2015) on topological mapping and Lowry et al. (2016) exclusively for visual place recognition. In this section we will focus on the works that are most relevant to our proposal. We will review first the literature on place recognition descriptors, and later refer to full place recognition pipelines. Notice that our contribution lies mainly on the former, that is, the proposal of novel multi-view descriptors.

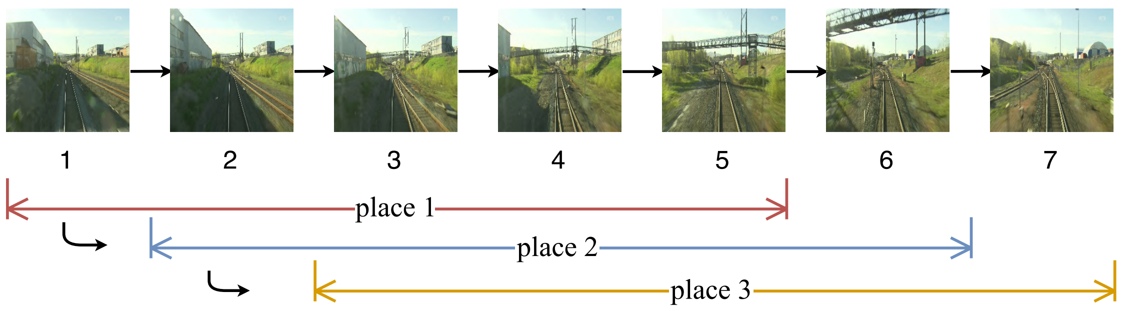


Fig. 4.3 A sliding window of five images is considered in this work as the same place. Notice the similarity of consecutive images. The figure is best viewed in electronic format.

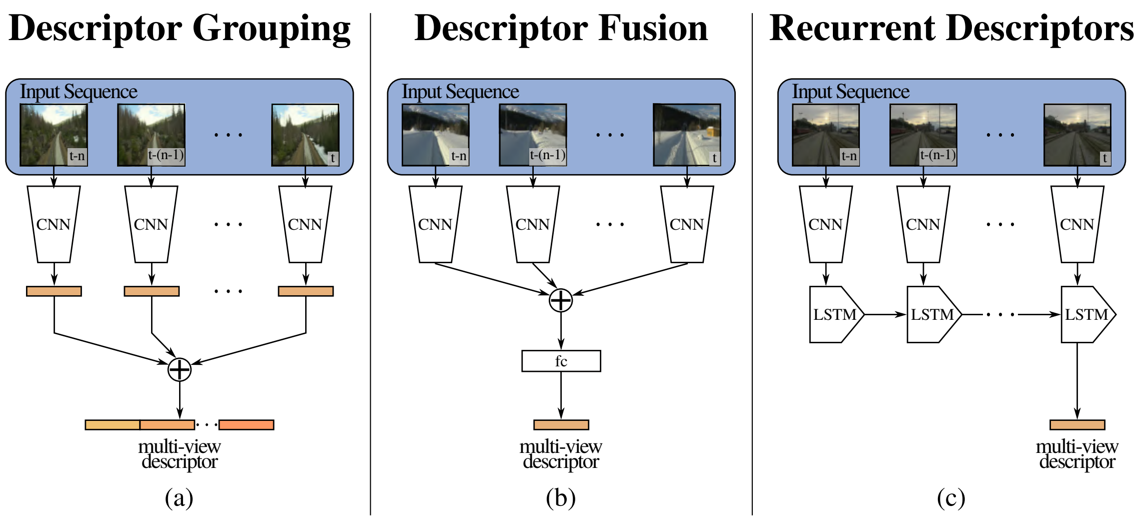


Fig. 4.4 **Multi-view descriptors** proposed in this thesis. From left to right: (a) **Descriptor Grouping**, where the descriptor of a sequence is the concatenation of all the single image descriptors. (b) **Descriptor Fusion**, the output of the CNNs serves as input to a fully-connected layer that combines the information into a single descriptor. (c) **Recurrent Descriptors**, the output of the CNNs serves as input to an LSTM network that integrates over time the single-image features to create a multi-image descriptor.

4.3.1 Descriptors

Local Descriptors

Techniques based on local descriptors address place recognition by detecting a set of salient keypoints in an image and generating descriptors for each one of them. These descriptors are used to find correspondences in other images, that would potentially allow us to perform visual place recognition or even 6DoF camera pose recovery. Detection and description are usually decoupled applying a method (Harris et al., 1988, Lowe, 2004, Mikolajczyk et al.,

2005) that detects salient points and then generate every a descriptor for every point using (Bay et al., 2006, Lowe, 2004, Rublee et al., 2011). Recently, some deep-learning approaches have addressed descriptor generation (Luo et al., 2018, 2019), keypoint detection (Savinov et al., 2017, Ono et al., 2018) or both in an end-to-end manner (Revaud et al., 2019, Dusmanu et al., 2019).

Global Descriptors

Despite the advantages of local descriptors, global descriptors come handy when the tasks requires larger context. One example of this is in the presence of large appearance changes like weather or illumination conditions. On those occasions, when local patterns might change substantially, a global view of the scene captured by high level features (*e.g.* the skyline of the city) may be more helpful.

Traditional global codes include handcrafted holistic image descriptors, like low-resolution thumbnails (Milford and Wyeth, 2012) or GIST (Murillo et al., 2013). None of these are robust to appearance changes due to scene dynamics, seasonal and weather changes, or extreme viewpoint or lighting variations. To address such cases, Lowry and Milford (2016) used PCA to reduce the dimensionality of descriptors eliminating the dimensions that are influenced by condition changes. Chen et al. (2018) incorporates attention in order to focus on the most relevant image features for place recognition. Descriptors based on CNNs have shown a high degree of robustness against appearance changes. Sünderhauf et al. (2015a) and Sünderhauf et al. (2015b) showed that CNNs outperform other models, especially for drastic appearance changes. They used AlexNet (Krizhevsky et al., 2012), pretrained on ImageNet (Russakovsky et al., 2015). The features of AlexNet contain semantic information about the whole scene, which improves the invariance to certain appearance changes. Thereafter, may other works have studied CNNs as condition-invariant feature extractors (Gomez-Ojeda et al., 2015, Arandjelovic et al., 2016, Arroyo et al., 2016, Chen et al., 2017, Lopez-Antequera et al., 2017, Olid et al., 2018). Gomez-Ojeda et al. (2015) were the first that trained a network as single-image feature extractor for visual place recognition under appearance changes. In NetVLAD (Arandjelovic et al., 2016), they proposed a new type of layer inspired in VLAD, an image representation commonly used in image retrieval. Chen et al. (2017) proposed a network trained to classify the place the image was taken. Olid et al. (2018) proposed a model based on pre-trained VGG-16 Simonyan and Zisserman (2014) and fine-tuned it for the place recognition task in a Triplet-Siamese architecture.

4.3.2 Visual Place Retrieval

This groups the approaches used to find retrieve the right place for every image. That comes down to the matching algorithms uses to retrieve the right image from the database of visited places.

Single-View Place Recognition

These are all those approaches which goal is to retrieve a single image from a single image; ignoring the time or pose relation between different frames in the query or in the database. The literature addresses the nearest neighbor problem in place recognition in many different ways: brute force (Olid et al., 2018, Murillo et al., 2013), KD-tree (Murillo et al., 2013) and bag of words (Gálvez-López and Tardos, 2012). Lowry and Andreasson (2018) presented a model using SURF detector and HOG features and studies the use of Bag of Words and Vectors of Locally Aggregated Descriptors (VLAD) for place matching.

Multi-View Place Recognition using Single-View Descriptors

For place recognition, it can be assumed that images in the database are independent and all that can be done is 1-to-1 matching, or else that you know the links between the images (in the query sequence and/or in the database) an use that as a prior knowledge. Although there are only a few works that consider temporal and multi-view information for place recognition, they all have shown that sequences provide useful information for place recognition. For instance, DBoW (Gálvez-López and Tardos, 2012) and Bampis et al. (2016) incorporate a temporal consistency constraint. SeqSLAM (Milford and Wyeth, 2012) and following works (Pepperell et al., 2014) use sequence matching, similarly to Newman et al. (2006). Differently to our approach, they assume linear temporal correlation for sequence matching. Also, we use data-driven high-level features, while they use downsampled images.

More recently, some works have extended SeqSLAM in several aspects. On the one hand, Chen et al. (2014) refines the temporal filtering. On the other hand, Naseer et al. (2018) proposes a graph of single-view descriptors (based on HOG and AlexNet) to model and match image sequences. Their approach is similar to SeqSLAM, with two main differences. The *most* straightforward one is that they use different descriptors. The second one, more subtle, is that their search of the best-matching sequence does not assume a constant speed variation between the sequences. SeqSLAM looks for straight lines in the similarity matrix, while Naseer et al. (2018) uses a more sophisticated model. In any case, none of them model changes in the sequence direction. Also, they typically rely on long-term sequence matching (*i.e.*, query and database sequences having many consecutive matching frames),

Method	**Descriptor Size**	**Accuracy avg**
		%
ours (single-view)	**128**	80.19%
ours (single-view)	256	80.62%
ours (single-view)	512	80.65%
ours (single-view)	1024	**81.46%**
		the bigger the better

Table 4.1 Evaluation of different descriptor sizes on the **Partitioned Nordland Dataset** (Olid et al., 2018) introduced previously in this chapter. **1st column:** Method. **2nd column:** Descriptor size in 32-bits floating point numbers. **3rd column:** Average accuracy.

which limits their applicability to such case. Assuming that consecutive frames have similar appearance, Neubert et al. (2015) combines CNN single-view descriptors with a directed search. Continuing their previous work Vysotska and Stachniss (2016) and Vysotska and Stachniss (2017) propose a combination of their lazy data association with a hash reduction of single-view CNN features.

4.3.3 Multi-View Place Recognition using Multi-View Descriptor

All the multi-view models described so far are handcrafted. Up to our knowledge, ours are the first ones that learn to combine multiple single-view features maps into a multi-view descriptor. We compare three different approaches to generate a descriptor based on multiple images, simple concatenation as baseline, learning to fuse descriptors and using a recurrent neural network to accumulate the knowledge over time. The traditional approach to using multiple views in place recognition is adding extra constraints when looking for the nearest neighbor. In our case, we do not add any constraints but generate descriptors that already include temporal and/or spatial information.

4.4 Network Architectures

In this section, we discuss four different models for place recognition: A single-view one, based on ResNet-50, and the three multi-view ones proposed in this chapter.

4.4.1 Single-View ResNet-50

Our first network is based on the model presented in Olid et al. (2018). The main difference is that we start from ResNet-50 (He et al., 2016) pretrained on ImageNet (Russakovsky et al., 2015) as our backbone, instead of VGG-16 (Simonyan and Zisserman, 2014). Although it is common to directly use the descriptors of different layers (see Section 4.6.1 for results on this), in our case we added and trained a fully connected layer after ResNet-50 to learn a 128-dimensional descriptor especially designated to the task of visual place recognition. We chose a size of 128 experimentally (see Table 4.1) , as a reasonable compromise between performance and compacity.

4.4.2 Descriptor Grouping

In order to include temporal information into the descriptors, our first approach is the naïve concatenation of the descriptors of consecutive frames, see Fig. 4.4a. Thus, starting from our previous single-view model, we first choose a temporal window of frames (n), we then generate a 128-dimensional descriptor per frame, and we finally concatenate them. The descriptor size is then $128 \times n$. Notice that this model is trained only from single-view samples. Hence, the relation between consecutive frames is not learned and this model only provides a filtering effect.

4.4.3 Descriptor Fusion

Descriptor Grouping, as the simplest strategy to consider several frames, is limited in its capability to weight differently certain features (*i.e.* features of some of the frames may be more representative of the place than others). It is also limited to cases where the sequences (map/query) are aligned – meaning that both sequences follow the same trajectory. For that reason, we designed a model that learns how to fuse the information of our n-frames window into a more discriminant –as well as smaller– 128-dimensional descriptor. With this Descriptor Fusion strategy, we add an extra fully connected layer that learns how to combine the outputs of n ResNet-50 into a single compact descriptor. See Fig. 4.4b for an illustration of this approach. As this network is able to learn how to weight the features from different frames, it can model more complex cases. For example, when sequences are recorded in reverse order Descriptor Grouping is limited, while Descriptor Fusion has the capability of learning a suitable fusion.

4.4.4 Recurrent Descriptors

Descriptor Fusion does not explicitly exploits the sequential nature of the data. With Recurrent Descriptors, we update in an online manner the sequence coding as new frames come, keeping the most relevant previous information. With that intention, we propose a Recurrent Neural Network (see Recurrent Descriptors in Fig. 4.4c). In this model, every

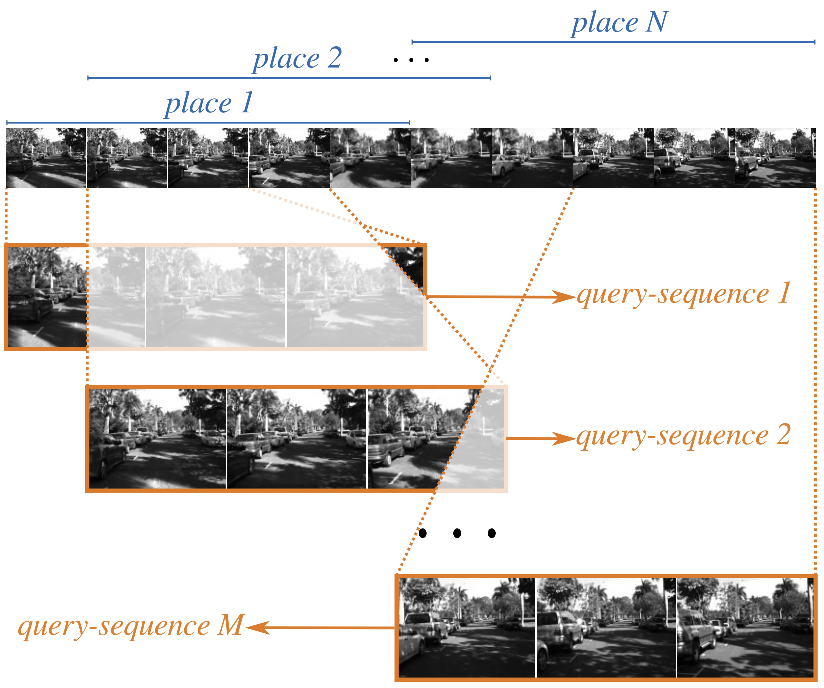


Fig. 4.5 Same place convention, illustrated with an example where the *query-sequence* has a length of 3 frames. A *place* represents a set of frames that are considered to be on the same place. Notice that a frame can be in more that one *place*. A *query-sequence* is an input sequence for our model. We want to recover the corresponding place of a *query-sequence*.

frame is the input to a ResNet-50, and the top layers serve as the input of a LSTM network (Hochreiter and Schmidhuber, 1997), that generates a 128-dimensional descriptor. LSTMs keep an inner state, that is updated with each input frame, and the output depends on the state and the input. Differently to previous models, keeping a recurrent inner state allows this network to produce a descriptor from the first frame, and update it sequentially as more frames arrive.

4.5 Training

4.5.1 Convention for Same Place

Since our descriptor is generated from a sequence of images (*query-sequence*) instead of a single image, we must define when two *query-sequence* of n frames are considered to be at the same *place* (the definition of a *place* being dataset-dependent). To illustrate this definition we will make use of Fig. 4.5. The figure shows a sequence of frames and several examples of *query-sequence*, and also shows the set of frames that we consider as the same *place*. Therefore, during training, we consider two *query-sequence* to be on the same *place* if they

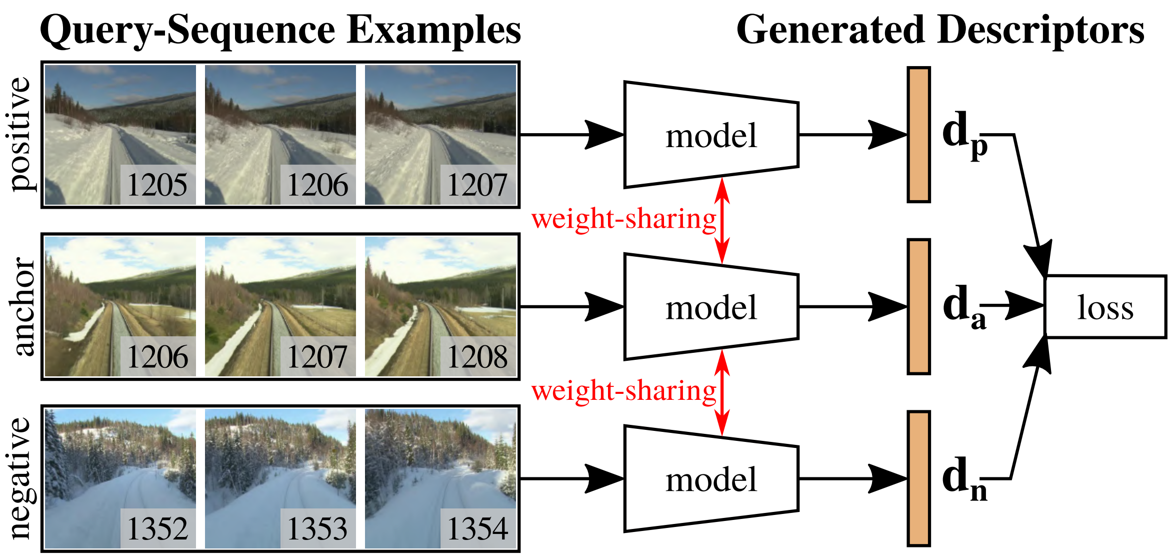


Fig. 4.6 Triplet architecture. We used this scheme for training all our models.

contain two frames (one per *query-sequence*) that belong to the same *place*. For instance, in Fig. 4.5, *query-sequence 1* and *query-sequence 2* belong to the same place, as the first frame of *query-sequence 1* belongs to *place 1*, the same as the first frame of *query-sequence 2*).

4.5.2 Model training

We start from ResNet-50 pre-trained on ImageNet in a standard classification task. We add the extra layers, and train them on our place-recognition task in our datasets. We trained all the models proposed in this work using a triplet architecture (see Fig. 4.6 for a scheme and more details). In a few words, triplet architectures are given 3 training samples: An anchor, a positive example and a negative one. During training, the objective is to reduce the distance between the anchor and positive descriptors, and to increase the distance between the anchor and the negative one. The loss we use to achieve that is the *Wohlhart-Lepetit* loss (Wohlhart and Lepetit, 2015),

$$\mathscr{L} = \max\left\{0, 1 - \frac{||\mathbf{d_a} - \mathbf{d_n}||}{m + ||\mathbf{d_a} - \mathbf{d_p}||}\right\}, \tag{4.1}$$

where m (margin) is a parameter that limits the difference between the distances, $\mathbf{d_a}$ is the descriptor generated for the *anchor* image, $\mathbf{d_p}$ is the descriptor for the *positive* sample and $\mathbf{d_a}$ is the descriptor for the *negative* sample (see Fig. 4.6). The specific training details for each model are as follows.

Descriptor Grouping

This model is trained as a single-view place recognition model. Hence, the data triplets consist of single images (an anchor image, a positive and a negative examples). During training, every single image generates a 128-dimensional descriptor, which means a $128 \times n$ elements descriptor during test.

Descriptor Fusion

Our second model learns a fusion of features for an image sequence (*query-sequence*). We concatenate the output of the ResNet-50 for each image, and add a fully-connected extra layer to generate a single descriptor of 128 elements. We train this model generating triples samples of n-frames *query-sequence*.

Recurrent Descriptors

In our last model we make a sequential update of the image descriptors using Recurrent Neural Networks, concretely an LSTM layer. In order to force the network to learn from the three images instead of only the last one, we add some random sampling in one of the n images of the *query-sequence* plus a Dropout on the LSTM layer.

4.6 Experimental Results

In this section we evaluate our three descriptors on two datasets: Partitioned Norland Olid et al. (2018) and Alderley Milford and Wyeth (2012); comparing them against state-of-the-art single-view and sequence-based methods. For every method, we retrieve the nearest neighbor as the matched place for a query image or sequence. We consider it as a correct match if it fits with the same-place convention for the dataset (*i.e.* each dataset has its own ground-truth frame correspondence). In the experiments, we set the ***query-sequence*** **length to 3 frames** for all our multi-view models. We observed in our experiments that, for more than 3 frames, the performance did not improved significantly. It slightly improves for our Descriptor Grouping, but degrades its performance when query and reference sequences do not have many consecutive matching frames. To compare different models we report the accuracy, *i.e.*, we retrieve a single place for every query (its nearest neighbor) and compute the fraction of correct matches over the total number of queries.

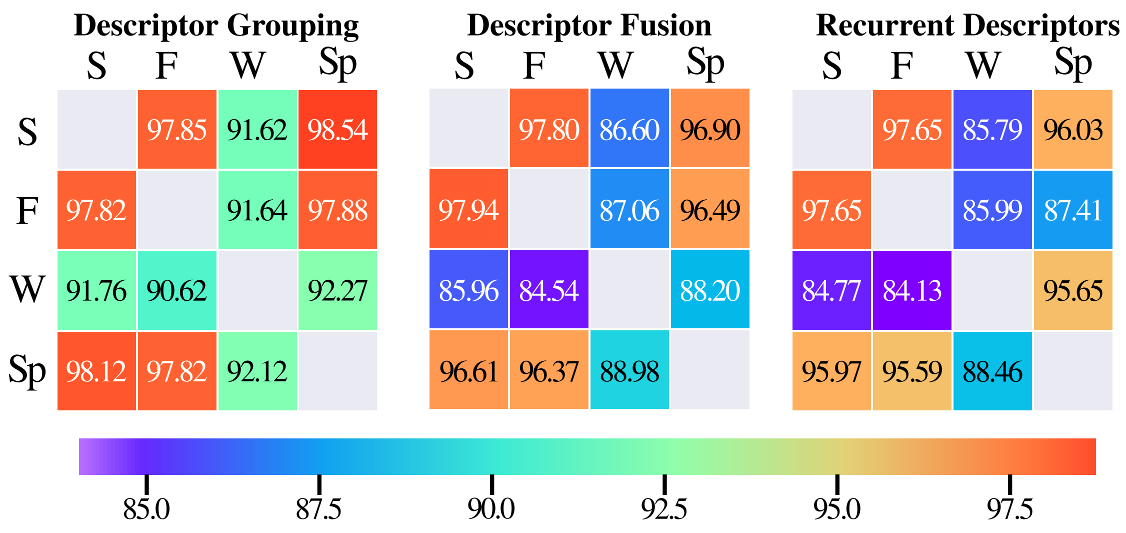
Method	**Number of frames**	**Descriptor Size**	**Accuracy W vs S**	**Accuracy S vs W**
	#		%	%
VGG16(pool4)	1	100352	51%	21%
VGG16(pool5)	1	25088	13%	7%
VGG16(fc6)	1	4096	6%	3%
VGG16(fc7)	1	4096	4%	3%
ResNet-50(3a-2a)	1	100352	42%	32%
ResNet-50(3d-2b)	1	100352	73%	42%
ResNet-50(4a-2a)	1	50176	62%	41%
ResNet-50(4b-2a)	1	50176	62%	31%
ResNet-50(4c-2a)	1	50176	50%	40%
ResNet-50(4f-2b)	1	50176	12%	8%
ResNet-50(5a-2a)	1	100352	43%	24%
Hybridnet (Chen et al., 2017)	1	4096	77%	41%
Amosnet (Chen et al., 2017)	1	4096	69%	48%
Lowry and Milford (2016)	1	1860	67%	66%
Olid et al. (2018)	1	128	75%	79%
ours (single-view)	1	128	77%	75%
ours (grouping)	3	384	92%	92%
ours (fusion)	3	**128**	87%	86%
ours (recurrent)	3	**128**	85%	86%
			the bigger the better	

Table 4.2 Results on the **Partitioned Nordland Dataset** Olid et al. (2018). **1st column:** Method. **2nd column:** Number of frames used for recognition (*e.g.*, 1 stands for single-view). **3rd column:** Descriptor size in 32-bits floating point numbers. **4th column:** Winter vs Summer, query taken from winter and matched to summer database. **5th column:** Summer vs Winter, query taken from summer and matched to winter database.

4.6.1 Partitioned Nordland Dataset

Our experiments use the train-test split proposed previously in this chapter. We trained our model with 24.5K images and evaluated its performance in a 3.45K-images set. We evaluated the performance of the descriptors of different layers of ResNet-50 and the best performing features are those of the layer *bn3d-branch2b* (3d-2b in the table), so we use these in our experiments. We train for 5 full epochs, where each epoch corresponds to 840K triplet examples.

Quantitative Results: Results on Table 4.2 show the performance of different models for visual place recognition on the Partitioned Nordland. In this table we report the hardest recognition cases, which are representative for the rest, specifically using the seasons Winter



Descriptor Grouping

	S	F	W	Sp
S		97.85	91.62	98.54
F	97.82		91.64	97.88
W	91.76	90.62		92.27
Sp	98.12	97.82	92.12	

Descriptor Fusion

	S	F	W	Sp
S		97.80	86.60	96.90
F	97.94		87.06	96.49
W	85.96	84.54		88.20
Sp	96.61	96.37	88.98	

Recurrent Descriptors

	S	F	W	Sp
S		97.65	85.79	96.03
F	97.65		85.99	87.41
W	84.77	84.13		95.65
Sp	95.97	95.59	88.46	

Fig. 4.7 **Accuracy** on Partitioned Nordland Dataset Olid et al. (2018). We evaluate the fraction of correct matches between all the seasons. **S** stands for summer, **F** for Fall, **W** for Winter and **Sp** for Spring. Our best performing model, Descriptor Grouping, never drops under 90% of correct matches. Models in this figure are using 3-frames sequences

and Summer both as query and database respectively. The upper part of the table shows single-view models and the lower part shows the multi-view models.

Our three proposals **ours (grouping)**, **ours (fusion)** and **ours (recurrent)** outperform very clearly our single-view approach **ours (single-view)**. Notice that they also outperform the state-of-the-art baselines. Among our multi-view proposals, **ours (grouping)** is the one achieving the best performance (92% of accuracy using 3 frames). Notice that its descriptor size, 384, is smaller than most of the single-view and mult-view baselines.

Increasing the number of frames in the Fusion approach increases the number of parameters and slows down the efficiency and training significantly. On the other side the Recurrent approach would allow to vary the number of frames without modifying the model, but it requires a more complex and dedicated study that will be addressed as future work.

Fig. 4.7 shows the results of our descriptors for all query-reference combinations. As mentioned, winter is always the hardest case. Notice, however, that none of our models drops under 80% performance.

Qualitative Results: Fig. 4.8 shows some examples of matched places with the grouping model and illustrates when multi-view methods achieve better performance. Notice that, although our single-view method fails in these examples, some of the places are indeed very similar and would be hard to match even by humans.

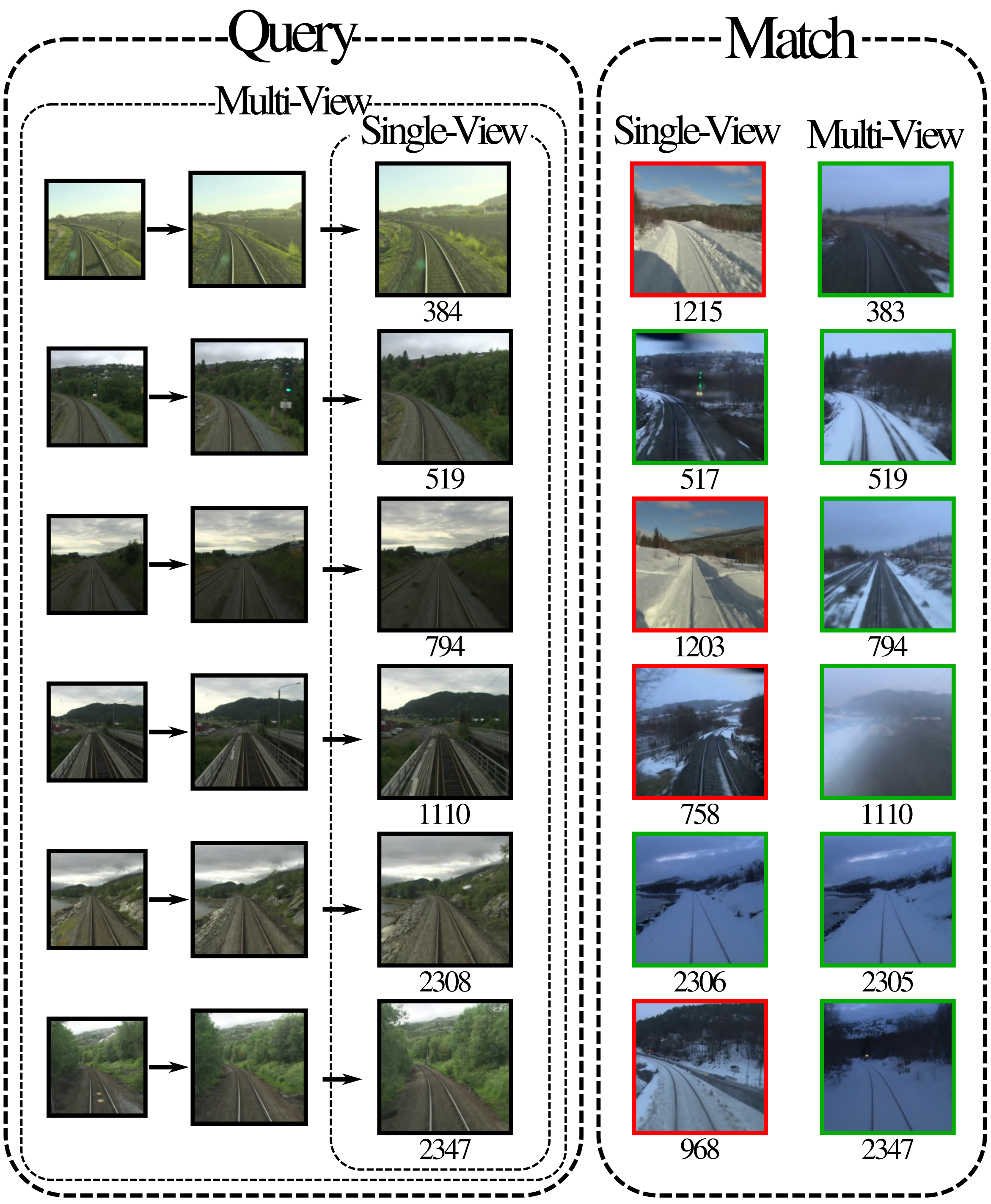


Fig. 4.8 **Example of Matched Places** for Single and Multi-View grouping model in the Partitioned Nordland Dataset (Olid et al., 2018). The retrieved image is framed on green if it is a correct match or red if it is incorrect. Mismatched frames are very similar and could even fool humans if they are not carefully inspected.

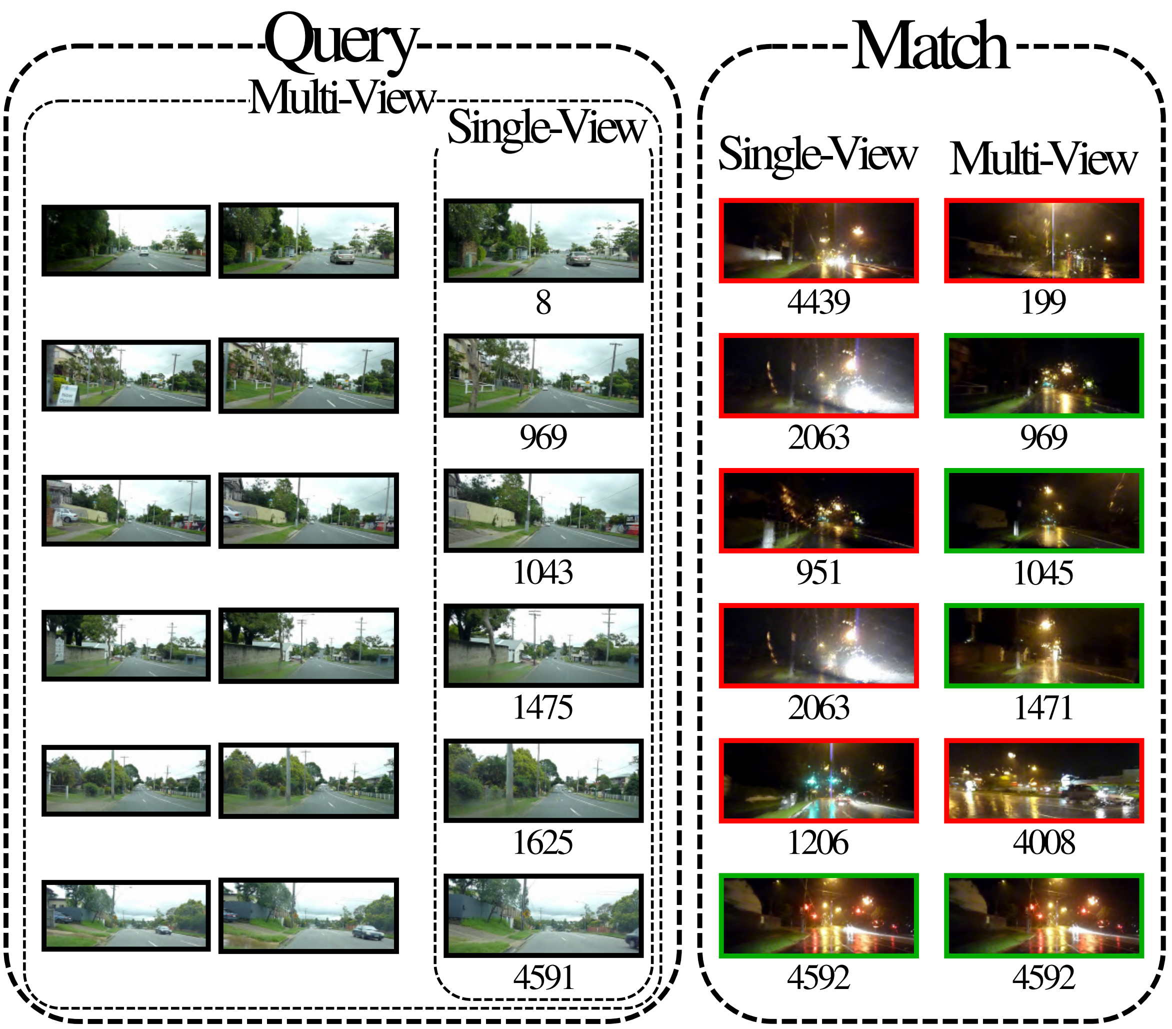


Fig. 4.9 **Examples of Matched Places** for Single and the grouping Multi-View model in the Alderley Dataset (Milford and Wyeth, 2012). The returned image is framed on green if it is a correct match or on red if it is incorrect.

4.6.2 Alderley

We also evaluated our approach on the Alderley dataset (Milford and Wyeth, 2012), that contains 15K images of a car trip in the day, and the same trip at night. It is a very challenging dataset due to the extreme illumination changes. We used the last 4.6K images as test samples.

Table 4.3 shows that our multi-view approach is the best performing descriptor, outperforming the rest both in accuracy and descriptor compacity. We also compare the difference when training on Partitioned Norland and Alderley (only shown for single-view approaches, as results are similar for multi-view ones). Fine-tuning on Alderley clearly helps on the task, as the condition variations (*seasons* vs *day/night*) impact differently in the visual appearance.

Method	**Trained on**	**Number of frames**	**Descriptor Size**	D vs N
		#		%
Olid et al. (2018)	Norland	1	128	0.15%
Olid et al. (2018)	Alderley	1	128	6.84%
ours (single-view)	Norland	1	128	1.65%
ours (single-view)	Alderley	1	128	6.8%
ours (grouping)	Alderley	3	384	9.05%
ours (grouping)	Alderley	6	768	**11.48%**
ours (fusion)	Alderley	3	128	**10.18%**
ours (recurrent)	Alderley	3	128	5.73%
				biggest the best

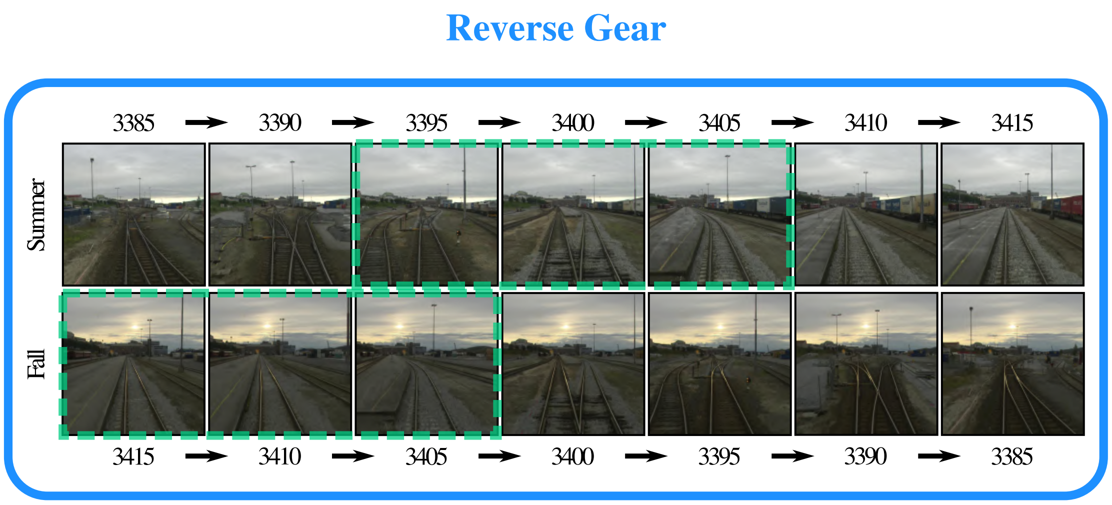
Table 4.3 Results on the **Alderley Dataset** Milford and Wyeth (2012), **1st column:** Method. **2nd column:** Dataset in which the model it has been trained with. **3rd column:** Number of frames used for recognition (*e.g.* 1 would imply to be single-view). **4th column:** Descriptor size in 32b floating point numbers.**5th column:** Day vs Night (D vs N), query with daylight image while reference database composed by nighttime images.

Qualitative results. Fig. 4.9 shows several test samples. Notice the increased challenge with respect to the Nordland dataset, with the presence of severe illumination changes plus inclusion of artificial illumination and dynamic objects.

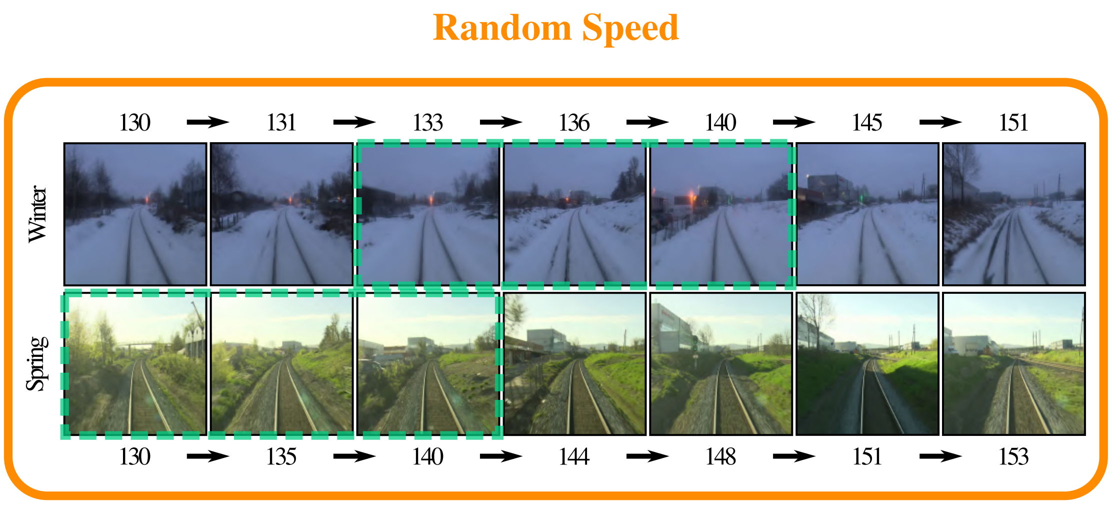
4.6.3 Multi-View Evaluation

Sequence Speed Changes: Inspecting the previous results (Table 4.2), Descriptor Grouping (**ours (grouping)**) trained only on single-view and then applied on multi-view by concatenation is the best performing. This is surprising at first sight, as the other two models were trained on multi-view data. We designed two extra experiments (**Reverse Gear** and **Random Speed**) to illustrate why this is happening. Both experiments are performed during test time, which means none of the networks has been retrained.

The **Reverse Gear** experiment consist on changing the direction of the train motion on one of the sequences at test time (*e.g.* when testing Winter vs Fall, the sequence of Fall is played in reverse order, see Fig 4.10a). This experiment will help to discern how much the model exploits the multi-view information rather than just the sequence consistency. Table 4.4 shows that, as we expected, models trained with multi-view examples (**ours (fusion)** and **ours (recurrent)**) have learned to exploit multiple views: Its performance only degrades by 6% and 4% respectively. On the other side, **ours (grouping)** drops severely its performance, by 18%.



(a)



(b)

Fig. 4.10 **Experiment setup details.** (a) **Reverse Gear**, in which the sequence is played in reverse order for one of the seasons (*Fall* in the figure). (b) **Random Speed**, in which the vehicle speed is modified for both seasons, reference (*Winter*) and query (*Spring*). In both cases, we mark with a **dashed green box** the same-place three-frames sequences.

In the **Random Speed** experiment we synthetically modified the speed of the train motion on one of the sequences at test time. Specifically, we modified the frame rate along the sequence simulating changes on the train velocity, see Fig. 4.10b (in our experiments the velocity was randomly multiplied by $\times 1$, $\times 2$ or $\times 3$ at every moment of the sequence). The "speed" is modified for the whole sequence, implying that the one-to-one correspondence in plain Nordland does not hold. Table 4.4 proves that **Random Speed** is the most challenging setup for the **ours (grouping)** approach, dropping its accuracy to 36%. **ours (fusion)** and

Method	Number of frames	Normal Test	Reverse Gear	Random Speed	Mean± Std
	#	%	%	%	%
SeqSLAM	3	33%	0.08%	9%	14.0±13.9
ours (grouping)	3	**92%**	74%	36%	67.3±23.3
ours (fusion)	3	86%	80%	78%	81.33±3.4
ours (recurrent)	3	86%	**82%**	**84%**	**84.0±1.6**
		biggest the best			

Table 4.4 Experimental results for **Speed Changes** in the **Partitioned Norland Dataset** Olid et al. (2018). Comparison of all our multi-view methods and SeqSLAM presented by Milford and Wyeth (2012). **1st column:** Method. **2nd column:** Number of frames used for recognition. **3rd-5th column:** Summer vs Winter experiments: *3rd column:* **Normal Test** corresponds to the one showed on Table 4.2. *4th column:* **Reverse Gear** experiment where the query frames are all in reversed order, *i.e.* simulating the train has used a *reverse gear*. *5th column:* **Random Speed** experiment where the speed of the train is simulated to be random, which means some of the frames are lost). The speed variations are independent for the query and the reference databases, and this implies no more 1 to 1 correspondence. These *speed changes* are perform only on the test sequences, this implies that for a network, *e.g.* **ours (grouping)**, results on the experiments are achieved using the same set parameters than in Table 4.2 with no retraining.

ours (recurrent) keep its performance at a very similar level than the standard Nordland setup (78% and 84% respectively). Notice that the goal of this experiment is not creating a very realistic setup, but misaligning the query sequence and the database ones. The aim is to evaluate the generalization over a plain frame-to-frame filtering effect.

We run the state-of-the-art multi-view baseline SeqSLAM (Milford and Wyeth, 2012) in both experiments, **Reverse Gear** and **Random Speed**, observing that its performance drops in both. This should be expected, as SeqSLAM assumes a linear relation between the velocities of the query and the reference sequences (sequence consistency).

The last column of Table 4.4 (**Mean/Std**) summarizes the conclusions of both experiments, reporting the mean (biggest the best) and standard deviation (smallest the best) for all experiments (**Normal Test**, **Reverse Gear** and **Random Speed**). Observe that **ours (recurrent)** is the best performing, presenting both the highest average accuracy and smallest variations. This confirms our hypothesis: The sequence descriptors that use learning (**ours (fusion)** and (**ours (recurrent)**) are more resilient than those based on plain concatenation (**ours (grouping)**) or handcrafted relations (SeqSLAM).

Method	**Num of frames**	**Descriptor Size**	**Accuracy W vs S Norland**	**Accuracy S vs W Norland**	**Accuracy D vs N Alderley**
	#		%	%	
SeqSLAM	3	6144	31%	33%	3.91%
SeqSLAM	10	20480	71%	70%	9.90%
SeqSLAM	100	204800	95%	94%	-
ours (grouping)	3	384	92%	92%	9.05%
ours (grouping)	6	768	97%	97%	11.48%
ours (fusion)	3	**128**	87%	86%	10.18%
ours (recurrent)	3	**128**	85%	86%	5.73%
ours (recurrent)	6	128	87%	88%	-
			the bigger the better		

Table 4.5 Results on the **Partitioned Nordland Dataset** (Olid et al., 2018) and **Alderley** adding more frames in the for the search. Comparison of all our multi-view methods and SeqSLAM presented by Milford and Wyeth (2012). **1st column:** Method. **2nd column:** Number of frames used for recognition (*e.g.*, 1 stands for single-view). **3rd column:** Descriptor size in 32-bits floating point numbers. **4th column:** Winter vs Summer, query taken from winter and matched to summer database. **5th column:** Summer vs Winter, query taken from summer and matched to winter database. **6th column:** Day vs Night, query taken from night and matched to day database.

Last, in Table 4.5 we show a comparison between our methods and SeqSLAM (Milford and Wyeth, 2012) on both Norland and Alderley. We also evaluate the effect of considering more frames in the different methods.

Our methods outperform SeqSLAM (Milford and Wyeth, 2012), a state-of-the-art baseline able to model information from several frames, when both use the same number of frames (specifically, 3). As all multi-view approaches improve their performance when increasing the number of frames, we increased the number of frames used by SeqSLAM. Notice that, in order to outperform our approach in Norland, the number of frames has to be increased up to 100 and 10 in Alderley. Similarly occurs when increasing the number of frames used by ours.

4.6.4 Execution time

We compared the execution time of our models in the upper part of Table 4.6. The fourth column (**Descriptor Extraction**) shows the time needed to extract the descriptor of a query 3-frames sequence on a NVIDIA TITAN Xp. In this part of our pipeline Descriptor Grouping is the fastest method, as is uses the simplest network.

Method	**Descriptor Size**	**Descriptor Extraction**	**Search** *1 vs 10K*
		ms	*ms*
ours (fusion)	128	17	**3.86**
ours (recurrent)	128	22	**3.86**
ours (grouping)	384	**15**	10.70
-	1860	-	49.21
-	4096	-	111.44
-	6144	-	166.13
-	20480	-	688.24
-	204800	-	9279.62
		smallest the best	

Table 4.6 **Execution Time** of all our models. 1st **column:** Method. 2nd **column:** Descriptor size. 3rd **column:** Time in milliseconds needed to extract 1 descriptor. 4th **column:** Given a descriptor and a reference data base of $10K$ descriptors, time in milliseconds needed to find the best match.

Last column (**Search**) shows the time needed to find the best match (Nearest Neighbor (NN)) given a query and a database of $10K$ descriptors. Notice that our methods Descriptor Fusion and Recurrent Descriptors are faster. This was expected, as their descriptor sizes are n (*query-sequence* size) times smaller (3 times in our experiments). Our NN algorithm consists on an exhaustive search through the database. We iterate over all the visited places, compute the distance between their descriptors and the query and keep the minimum-distance one. For the distance function we use the *Squared Euclidean Distance* ($d^2(\mathbf{d}_q, \mathbf{d}_i)$). The computational complexity of this search is $\mathcal{O}(N)$ where N is the number of elements in the database and for the distance function $\mathcal{O}(k)$ where k is the descriptor size.

Additionally, we computed the search time corresponding to the sizes of some of the other descriptors used in Tables 4.2 and 4.3 (bottom part of the table). As expected, the time increases with the descriptor size. Notice that high dimensional descriptors rule out the use of more efficient data structures, such as *KD-trees*, to speed up the search techniques, since it is not possible to reject candidates by using the difference of a single coordinate (Marimont and Shapiro, 1979). A directed search using the sequentiality of data (Neubert et al., 2015, Vysotska and Stachniss, 2016) would reduce the number of comparisons. Hashing (Vysotska and Stachniss, 2017) can also reduce the computational cost. In any case, reducing the dimensionality of the descriptor has a direct influence in the cost of all the approaches mentioned.

4.7 Conclusions

In this chapter we have introduced three deep learning-based multi-view global descriptor models, that outperform existing baselines both in accuracy and compactness. We analyzed different approaches to combine the information of the features from multiple views (Grouping, Fusion and Recurrent), and we evaluated them on different experimental setups in two public datasets: Partitioned Norland and Alderley. Each model we propose has its own strengths and weaknesses. On the one side, Descriptor Grouping ensures the sequential consistency of the frame in a sequence, achieving the best performance in the standard Nordland/Alderley benchmarks, where the inter-frame motion is similar in different runs. On the other side, Descriptor Fusion and Recurrent Descriptors are able to learn more complex relations between frames and hence proved to be better in cases where the velocities differ or the frames ordering is different. We also show the low computational cost of all the approaches, demonstrating its potential for robotic applications.

We believe that recurrent models, in spite of challenges associated to training and generalizing to different vehicle dynamics, are promising. They are robust to speed changes and adapt to different sequence lengths without increasing the descriptor and network sizes. We have observed such challenges in preliminary results, along with a slight performance improvement with an increase of the sequence length (up to 6 frames). Encouraged by this, our future work will study carefully the use of recurrent models with large image sequences. However that presents the challenge of the beginning and the end of the sequence being two place too far apart.

Chapter 5

Corner Prediction for Layout Reconstruction

Another relevant cue for visual 3D reconstruction is the extraction of high-level semantic information corresponding to the structure of the scene. For example, detecting and estimating depth and orientation of planar structures (Concha and Civera, 2015a). In this chapter, we focus on detecting the main structure for indoor scenes. Also refereed as layout recovery, that is detecting the walls, ceiling and floor. We have not applied our advances directly to visual localization or mapping for sequences but we refer the reader to Salas et al. (2015) as an example of how this advances could be use for Visual SLAM.

The problem of 3D layout recovery in indoor scenes has been a core research topic for over a decade. However, there are still several major challenges that remain unsolved. Among the most relevant ones, a major part of the state-of-the-art methods make implicit or explicit assumptions on the scenes –*e.g.* box-shaped or Manhattan layouts. Also, current methods are computationally expensive and not suitable for real-time applications like robot navigation and AR/VR. In this work we present CFL (Corners for Layout), the first end-to-end model that predicts layout corners for 3D layout recovery on $\mathbf{360}^\circ$ images. Our experimental results show that we outperform the state of the art, making less assumptions on the scene than other works, and with lower cost. We also show that our model generalizes better to camera position variations than conventional approaches by using EquiConvs, a convolution applied directly on the spherical projection and hence invariant to the equirectangular distortions.

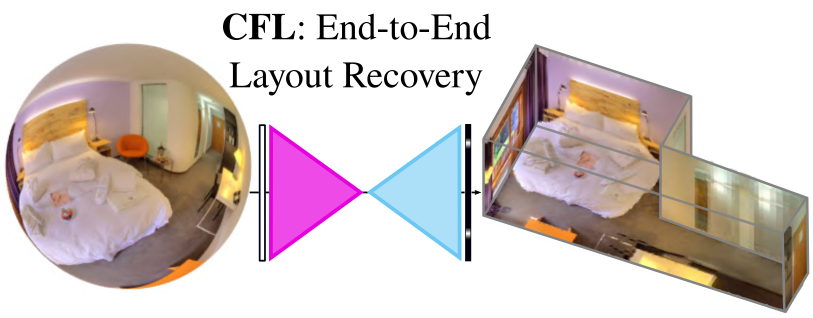


Fig. 5.1 **Corners for Layout**: The model end-to-end predicts the layout corners from the spherical image. Connecting the corners and assuming ceiling-floor parallelism, we can directly obtain the 3D layout in a very short time.

5.1 Introduction

Recovering the 3D layout of an indoor scene from a single view has attracted the attention of computer vision and graphics researchers in the last decade. The idea is going beyond pure geometrical reconstructions and provide higher-level contextual information about the scene, even in the presence of clutter. Layout estimation is a key technology in several emerging application markets, such as augmented and virtual reality and robot navigation (Salas et al., 2015). But also for more traditional ones, like real estate (Liu et al., 2015a).

Layout estimation, however, is not a trivial task and there are several major problems that still remain unsolved. For example, most existing methods are based on strong assumptions on the geometry (*e.g.* Manhattan scenes) or the over-simplification of the room types (*e.g.* box-shaped layouts), often underfitting the richness of real indoor spaces. The limited field of view of conventional cameras leads to ambiguities, which could be solved by considering a wider context. For this reason it is advantageous to use wide fields of view, like 360° panoramas. In these cases, however, the methods for conventional cameras are not suitable due to the image distortions and new ones have to be developed (Pais et al., 2019).

In the last years, the main improvements in layout recovery from panoramas have come from the application of deep learning. The high-level features learned by deep networks have proven to be as useful for this problem as for many others. Nevertheless, these techniques entail other problems such as the lack of data or overfitting. State-of-the-art methods require additional pre- and/or post-processing. As a consequence they are very slow, and this is a major drawback considering the aforementioned applications for real-time layout recovery.

In this work, we present Corners for Layout (CFL), the first end-to-end neural network that predicts a map of the corners of the room to directly obtain the 3D layout from a single 360° image (Figure 5.1). This makes **CFL more than 100 times faster** than the state of the art, while still **outperforming the accuracy of current approaches**. Furthermore, our proposal is not limited by typical scene assumptions, meaning that it can predict complex geometries, such as rooms with more than four walls or non strict Manhattan structures. Additionally, we propose a novel implementation of the convolution for 360° images (Tateno et al., 2018, Cohen et al., 2018) in the equirectangular projection. We deform the kernel,using the advances presented by Dai et al. (2017b), to compensate the distortion and make CFL more **robust to camera rotation and pose variations**, generalizing to unseen configurations. Hence, it is equivalent to applying directly a convolution operation to the spherical image, which is geometrically more coherent than applying a standard convolution on the equirectangular panorama. We have extensively evaluated our network in two public datasets with several training configurations, including data augmentation techniques to address occlusions by enforcing the network to learn from the context. We also propose a **robustness analysis** to see the effect of extrinsic variations in panoramas and dataset bias. Our **code** and labeled **dataset** can be found here: CFL webpage.

5.2 Related Work

The layout of a room provides a strong prior for other visual tasks like single-view (Eigen and Fergus, 2015) and multi-view depth recovery (Concha et al., 2014), realistic insertions of virtual objects into indoor images (Karsch et al., 2011), indoor object recognition (Bao et al., 2011, Song and Xiao, 2016), indoor place recognition (Hussain et al., 2016) or human pose estimation (Fouhey et al., 2014). A large variety of methods have been developed for this purpose using multiple input images (Tsai et al., 2011, Flint et al., 2011) or depth sensors (Zhang et al., 2013), which deliver high-quality reconstruction results. For the common case when a single RGB image is available, the problem becomes considerably more challenging and researchers need very often to rely on strong assumptions.

The seminal approaches to layout prediction from a single view were (Delage et al., 2006, Lee et al., 2009), followed by (Hedau et al., 2009a, Schwing et al., 2013). They basically model the layout of the room with a vanishing-point-aligned 3D box, being hence constrained to this particular room geometry and unable to generalize to others appearing frequently in real applications. Most recent approaches exploit CNNs and their excellent performance in a wide range of applications such as image classification, segmentation and detection. (Mallya and Lazebnik, 2015, Ren et al., 2016, Zhang et al., 2017, Zhao et al., 2017), for example, focus on predicting the informative edges separating the geometric classes (walls, floor and ceiling). Alternatively, Dasgupta et al. (2016) proposed a FCN to predict labels for each of the surfaces of the room. All these methods require extra computation added to the forward propagation of the network to retrieve the actual layout. In Lee et al. (2017), for example, an end-to-end network predicts the layout corners in a perspective image, but after that it has to infer the room type within a limited set of manually chosen configurations.

While layout recovery from conventional images has progressed rapidly with both geometry and deep learning, the works that address these challenges using omnidirectional images are still very few. Panoramic cameras have the potential to improve the performance of the task: their 360° field of view captures the entire viewing sphere surrounding its optical center, allowing to acquire the whole room at once and hence predicting layouts with more visual information. PanoContext (Zhang et al., 2014) was the first work that extended the frameworks designed for perspective images to panoramas. It recovers both the layout, which is also assumed as a simple 3D box, and bounding boxes for the most salient objects inside the room. Pano2CAD (Xu et al., 2017) extends the method to non-cuboid rooms, but it is limited by its dependence on the output of object detectors. Motivated by the need of addressing complex room geometries, Fernandez-Labrador et al. (2018b) generates layout hypotheses by geometric reasoning from a small set of structural corners obtained from the combination of geometry and deep learning. The most recent works along this line are LayoutNet (Zou et al., 2018), that trains a FCN from panoramas and vanishing lines, generating the layout models from edge and corner maps, and DuLa-Net (Yang et al., 2018), that predicts Manhattan-world layouts leveraging a perspective ceiling-view of the room. All of these approaches require pre- or post-processing steps like line and vanishing point extraction or room model fitting, that increase their cost.

In addition to all the challenges mentioned above, we also notice that there is an incrongruence between panoramic images and conventional CNNs. The space-varying distortions caused by the equirectangular representation makes the translational weight sharing ineffective. Very recently, Cohen et al. (2018) did a relevant theoretical contribution by studying convolutions on the sphere using spectral analysis. However, it is not clearly demonstrated whether Spherical CNNs can reach the same accuracy and efficiency on equirectangular images. EquiConvs are inspired by the work of Tateno et al. (2018). They propose distortion-aware convolutional filters to solve the problem of dense prediction by leveraging commonly used datasets with annotations for perspective images during training. In this work, instead, we exploit this idea to tackle the problem of layout recovery from panoramas and intensively study their robustness to camera pose variation. In practice, we propose a novel parameterization and implementation of the deformable convolutions (Dai et al., 2017b) by following the idea of adapting the receptive field of the convolutional kernels by deforming their shape according to the distortion of the equirectangular projection.

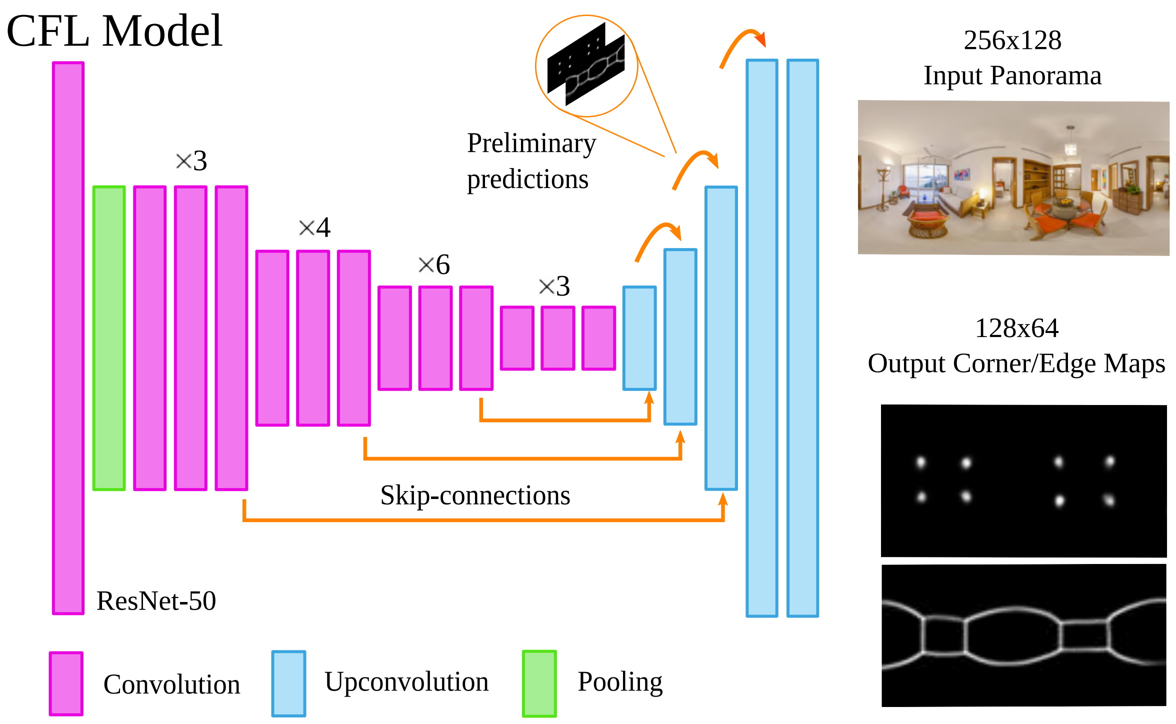


Fig. 5.2 **CFL architecture**. Our network is built upon ResNet-50, adding a single decoder that jointly predicts edge and corner maps. There are two network variations: one applies Standard Convolutions and Upconvolutions on the equirectangular panorama, whereas the other one applies Equirectangular Convolutions and Equirectangular Convolutions + unpooling directly on the sphere.

5.3 Corners for Layout

Here we describe our end-to-end approach for recovering the room corners that allow us to estimate the layout, *i.e.* the main structure of the room, from a single 360° image. First, we describe the proposed network architecture and training and finally we describe how we directly transform the output into the 3D layout. The network architecture is adapted for Standard Convolutions and for our proposed Equirectangular Convolutions implementation, the latest being explained in Section 5.4.

5.3.1 Network architecture

The proposed FCN follows the encoder-decoder structure and builds upon ResNet-50 (He et al., 2016). We replace the final fully-connected layer with a decoder that jointly predicts layout edges and corners locations already refined. We illustrate the proposed architecture in Figure 5.2.

Encoder. Most of deep-learning approaches facing layout recovery problem have made use of the VGG16 (Simonyan and Zisserman, 2014) as encoder (Mallya and Lazebnik, 2015, Dasgupta et al., 2016, Lee et al., 2017). Instead, Zhao et al. (2017) builds their model over ResNet-101 (He et al., 2016) outperforming the state of the art. Here, we use ResNet-50 (He et al., 2016), pre-trained on the ImageNet dataset (Russakovsky et al., 2015), which leads to a faster convergence due to the general low-level features learned from ImageNet. Residual networks allow us to increase the depth without increasing the number of parameters with respect to their plain counterparts. This leads, in ResNet-50, to capture a receptive field of 483×483 pixels, enough for our input resolution of 256×128 pixels.

Decoder. Most of the recent work (Mallya and Lazebnik, 2015, Zou et al., 2018, Ren et al., 2016) builds two output branches for multi-task learning, which increases the computation time and the network parameters. We instead propose a unique branch with two output channels, corners and edge maps, which helps to reinforce the quality of both map types. In the decoder, we combine two different ideas. First, skip-connections (Ronneberger et al., 2015) from the encoder to the decoder. Specifically, we concatenate "up-convolved" features with their corresponding features from the contracting part. Second, we do preliminary predictions at lower resolutions which are also concatenated and fed back to the network following the spirit of (Dosovitskiy et al., 2015), ensuring early stages of internal features aim for the task. We use ReLU as non-linear function except for the prediction layers, where we use Sigmoid.

We propose two variations of the network architecture for two different convolution operations (Figure 5.2). The first one, *CFL StdConvs*, convolves the feature maps with Standard Convolutions and use up-convolutions to decode the output. The second one, *CFL EquiConvs*, uses Equirectangular Convolutions both in the encoder and the decoder, using unpooling to upsample the output. Equirectangular Convolutions are deformable convolutions that adapt their size and shape depending on the position in the equirectangular image, for which we propose a new implementation explained in Section 5.4 to make our results reproducible.

Implementation

Tables 5.1 and 5.2 show more details about the network architecture and the differences between the two variants, the first one with standard convolutions, StdConvs, and the second one with Equirectangular Convolutions, EquiConvs. Notice that the decoder uses ReLU as standard activation function except for the prediction layers that use Sigmoid to obtain the edge and corner maps. Prediction layers include those that generate intermediate predictions, **IP-** in Tables 5.1 and 5.2, and the final **output** layer.

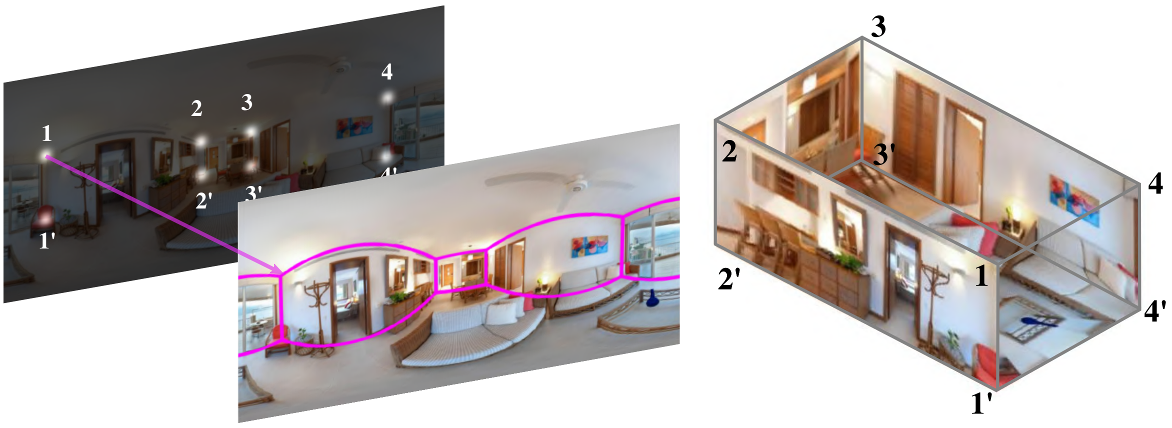


Fig. 5.3 **Layout from corner predictions**. From the corner probability map, the coordinates with maximum values are directly selected to generate the layout.

The blocks for the two models we present are:
CFL StdConvs: Uses the *encoder* and *decoder convs*. In this case *encoder* uses the *std-conv-block* and *std-id-block*, see Tables 5.1 and 5.2.
CFL EquiConvs: Uses the *encoder* and *decoder EquiConvs*. In this case *encoder* uses the *equi-conv-block* and *equi-id-block*, see Tables 5.1 and 5.2.

5.3.2 Training

Objective output

The ground truth (GT) for every panorama consists of a set of corner coordinates. With this coordinates we generate two maps, m, one represents the room edges ($m = e$), *i.e.* intersections between walls, ceiling and floor, and the other encodes the corner locations ($m = c$). Both maps are defined as $\mathscr{Y}^m = \{y_1^m, \ldots, y_i^m, \ldots\}$, with pixel values $y_i^m \in \{0, 1\}$. y_i^m has a value of 1 if it belongs to an edge or a corner, and 0 otherwise. Dealing with the image at pixel level is very noise-sensitive so we do line thickening and Gaussian blur for easier convergence during training since it makes the loss progression continuous instead of binary. The loss is gradually reduced as the prediction approaches the target.

Notice here that our target is considerably simpler than others that usually divide the ground truth into different classes. This contributes to the small computational footprint of our proposal. For example, (Mallya and Lazebnik, 2015, Zhao et al., 2017) use independent feature maps for background, wall-floor, wall-wall and wall-ceiling edges. A full image segmentation into left, front and right wall, ceiling and floor categories is performed in Dasgupta et al. (2016). In Lee et al. (2017), they represent a total of 48 different corner types by a 2D Gaussian heatmap centered at the true keypoint location. Here, instead, we only

INPUT	LAYER	K	S	FUN	CH	OUTPUT
			encoder			
image	conv+BN	7	2	ReLU	64	conv1
conv1	maxpool	3	2	-	64	pool
pool	conv-block	3	1	ReLU	256	res2a
res2a	id-block	3	-	ReLU	256	res2b
res2b	id-block	3	-	ReLU	256	res2c
res2c	conv-block	3	2	ReLU	512	res3a
res3a	id-block	3	-	ReLU	512	res3b
res3b	id-block	3	-	ReLU	512	res3c
res3c	id-block	3	-	ReLU	512	res3d
res3d	conv-block	3	2	ReLU	1024	res4a
res4a	id-block	3	-	ReLU	1024	res4b
res4b	id-block	3	-	ReLU	1024	res4c
res4c	id-block	3	-	ReLU	1024	res4d
res4d	id-block	3	-	ReLU	1024	res4e
res4e	id-block	3	-	ReLU	1024	res4f
res4f	conv-block	3	2	ReLU	2048	res5a
res4a	id-block	3	-	ReLU	2048	res5b
res4b	id-block	3	-	ReLU	2048	res5c
			decoder convs			
res5c	upconv	5	2	ReLU	512	upconv4
res4f,upconv4	concat	-	-	-	-	in3
in3	upconv	5	2	ReLU	256	upconv3
upconv3	upconv	3	1	Sigmoid	2	**IP-1**
res3d,upconv3,IP-1	concat	-	-	-	-	in2
in2	upconv	5	2	ReLU	128	upconv2
upconv2	upconv	3	1	Sigmoid	2	**IP-2**
res2c,upconv2,IP-2	concat	-	-	-	-	in1
in1	upconv	5	2	ReLU	64	upconv1
upconv1	upconv	3	1	Sigmoid	2	**IP-3**
conv1,upconv1,IP-3	concat	-	-	-	-	in
in	upconv	3	1	ReLU	64	upconv
upconv	upconv	3	1	Sigmoid	2	**output**
			decoder Equiconvs			
res5c	EquiConv	3	1	ReLU	512	equiconv4
equiconv4	unpool	2	2	-	-	unpool4
res4f,unpool4	concat	-	-	-	-	in3
in3	EquiConv	3	1	ReLU	256	equiconv3
equiconv3	unpool	2	2	-	-	unpool3
unpool3	EquiConv	3	1	Sigmoid	2	**IP-1**
res3d,unpool3,IP-1	concat	-	-	-	-	in2
in2	EquiConv	3	1	ReLU	128	equiconv2
equiconv2	unpool	2	2	-	-	unpool2
unpool2	EquiConv	3	1	Sigmoid	2	**IP-2**
res2c,unpool2,IP-2	concat	-	-	-	-	in1
in1	EquiConv	5	1	ReLU	64	equiconv1
equiconv1	unpool	2	2	-	-	unpool1
unpool1	EquiConv	3	1	Sigmoid	2	**IP-3**
conv1,unpool1,IP-3	concat	-	-	-	-	in
in	EquiConv	5	1	ReLU	64	equiconv
equiconv	EquiConv	3	2	Sigmoid	2	**output**

Table 5.1 Network Architecture Details. Note that *id-block* and *conv-block* can be either *equi-* or *std-* depending on the version of the network (see Table 5.2).

INPUT	LAYER	K	S	BN	FUN	CH	OUTPUT
	std-conv-block						
I	**std-conv-block**	**k**	**s**	**-**	**f**	**N**	**O**
I	StdConv	1	s	✓	f	N/4	O2a
O2a	StdConv	k	1	✓	f	N/4	O2b
O2b	StdConv	1	1	✓	-	N	O2c
I	conv	1	s	✓	-	N	O-1
O-1,O2c	add	1	1	-	f	N	O
	std-id-block						
I	**std-id-block**	**k**	**-**	**-**	**f**	**N**	**O**
I	StdConv	1	1	✓	f	N/4	O2a
O2a	StdConv	k	1	✓	f	N/4	O2b
O2b	StdConv	1	1	✓	-	N	O2c
I,O2c	add	1	1	-	f	N	O
	equi-conv-block						
I	**equi-conv-block**	**k**	**s**	**-**	**f**	**N**	**O**
I	EquiConvs	1	s	✓	f	N/4	O2a
O2a	EquiConvs	k	1	✓	f	N/4	O2b
O2b	EquiConvs	1	1	✓	-	N	O2c
I	EquiConvs	1	s	✓	-	N	O-1
O-1,O2c	add	1	1	-	f	N	O
	equi–id-block						
I	**equi-id-block**	**k**	**-**	**-**	**f**	**N**	**O**
I	EquiConvs	1	1	✓	f	N/4	O2a
O2a	EquiConvs	k	1	✓	f	N/4	O2b
O2b	EquiConvs	1	1	✓	-	N	O2c
I,O2c	add	1	1	-	f	N	O

Table 5.2 Blocks of the Network

use two probability maps, one for edges and another one for corners – see *outputs* in the Figure 5.2.

Loss function

Edge and corner maps are learned through a pixel-wise sigmoid cross-entropy loss function. Since we know a priori that the natural distribution of pixels in these maps is extremely unbalanced ($\sim$ 95% have a value of 0), we introduce weighting factors to make the training stable. Defining as 1 and 0 the positive and negative labels, the weighting factors are defined as $w_t = \frac{N}{N_t}$, being N the total number of pixels and N_t the amount of pixels of class t per sample. The per-pixel per-map loss $\mathscr{L}_i^m$ is as follows:

$$\begin{aligned}\mathscr{L}_i^m &= w_1(y_i^m(-\log(\hat{y}_i^m)))+ \\ &+ w_0((1-y_i^m)(-\log(1-\hat{y}_i^m))),\end{aligned} \tag{5.1}$$

where y_i^m is the objective value for pixel i in the map m and $\hat{y}_i^m$ is the network output for pixel i and map m. We minimize this loss at 4 different resolutions $k = \{1,\ldots,4\}$, specifically in the network output ($k = 4$) and 3 intermediate layers ($k = \{1,\ldots,3\}$). The total loss is then the sum over all pixels, the 4 resolutions and both the edge and corner maps

$$\mathscr{L} = \sum_{k=\{1,\ldots,4\}} \sum_{m=\{e,c\}} \sum_{i} \mathscr{L}_i^m [k]. \tag{5.2}$$

5.3.3 From Corner Maps to 3D Layout

Current methods (Zou et al., 2018, Fernandez-Labrador et al., 2018b, Zhang et al., 2014) use pre-computed vanishing points and posterior optimizations, being constrained to produce strict Manhattan 3D layouts. Aiming to a fast end-to-end simple model, CFL avoids extra computation and adopt a representation usually referred as Soft/Weak Manhattan (Furlan *et al*, 2013) or Atlanta World (Joo *et al*, 2018). Following this, horizontal directions are not necessarily orthogonal to each other, thus relaxing the model assumptions. To this end, we simply follow a natural transformation from corners coordinates to 2D and 3D layout. The 2D corners coordinates are the maximum activations in the probability map. Assuming that the corner set is consistent, they are directly joined, from left to right, in the unit sphere space and re-projected to the equirectangular image plane. The 3D layout is inferred by only assuming ceiling-floor parallelism, leaving the wall structure unconstrained –*i.e.*, we assume that the floor corners are on the same plane and the top corners are directly above the floor ones, but we do not force the usual Manhattan perpendicularity between walls. Corners are projected to floor and ceiling planes given a unitary camera height (trivial as results are up to scale). See Figure 5.3.

Here we provide a further explanation of how the process to go from 2D to 3D works. From the predicted 2D corner positions, we can directly recover the 3D layout by doing the following assumptions:

1. **Soft Manhattan or Atlanta world.** This is a relaxation of the Manhattan World assumption whereby horizontal directions are not necessarily orthogonal to each other. That is, walls can intersect with each other in any direction.

2. **Ceiling-floor parallelism.** Corners can be classified depending on their position along the vertical direction (above or below the horizon line, which in central panoramas is at the middle row) between ceiling and floor corners respectively. Floor corners are on the same floor plane and ceiling corners are directly above the floor ones. The vertical direction is the normal direction of both floor and ceiling planes.

3. **Unitary camera height.** This is trivial as results are up to scale but needed to predict the total height of the room.

Taking all of this into account, we can define a plane as the set of all points $P = (x, y, z)$ such that $P \cdot N + d = 0$, where the normal $N = (n_x, n_y, n_z)$ is a normalized vector perpendicular to its surface and d is the distance that separates it from the origin of coordinates in the direction of the normal. Due to assumptions b) and c), N of both the floor and ceiling planes is equal and corresponds to the vertical direction, and the distance d from the floor to the camera is known. The distance to the ceiling is yet unknown.

Additionally, thanks to the nature of spherical images, we can easily obtain the 3D ray $R(t) = O + \vec{V} \cdot t$ (parametric representation) going from the center of the sphere $O = (o_x, o_y, o_z)$ through the corner position, with normalized direction vector $\vec{V} = (v_x, v_y, v_z)$. To obtain the normalized direction vector $\vec{V}$, we need the corner position in the sphere, thus we transform the image coordinates of the corners (u, v) into spherical coordinates and then to the Euclidean 3D space. Equations for this can be found in Section 4.1. of this chapter. In the first place, Eq (5.3) give us the angles that define the point (u, v) in the sphere.

$$\phi = (u - \frac{W}{2})\frac{2\pi}{W} \quad ; \quad \theta = -(v - \frac{H}{2})\frac{\pi}{H} \tag{5.3}$$

Where W and H are the width and height of the equirectangular image. Second, once these rotations are known we can compute the direction of the ray. Therefore, using Eq (5.4) we can calculate $\vec{V}$.

$$\vec{V} = \begin{bmatrix} -\cos(\theta)\sin(\phi) \\ \sin(\theta) \\ \cos(\theta)\cos(\phi) \end{bmatrix} \tag{5.4}$$

The intersection between the corner ray and the corresponding floor or ceiling plane will give us the actual 3D corner point $P = (x, y, z)$ (up to scale), *ie.* the intersection represents that point P on the surface of the plane that verifies the ray equation: $(o_x + v_x \cdot t)n_x + (o_y + v_y \cdot t)n_y + (o_z + v_z \cdot t)n_z + d = 0$. The point P of intersection would simply be the result of evaluating the calculated t, Eq (5.5), in the ray equation $R(t)$.

$$t = -\frac{o_x n_x + o_y n_y + o_z n_z + d}{v_x n_x + v_y n_y + v_z n_z} \tag{5.5}$$

Let's consider we have performed the operations to compute one corner point on the floor plane, $P^F = (x^F, y^F, z^F)$. The corresponding point on the ceiling plane (P^C) will be on top of it (*ie.* $x^F = x^C$ and $y^F = y^C$). Therefore, we can use this to compute t^C, Eq (5.6), and thus the ceiling point:

$$t^C = \frac{(x^F - o_x)}{v_x^C} \tag{5.6}$$

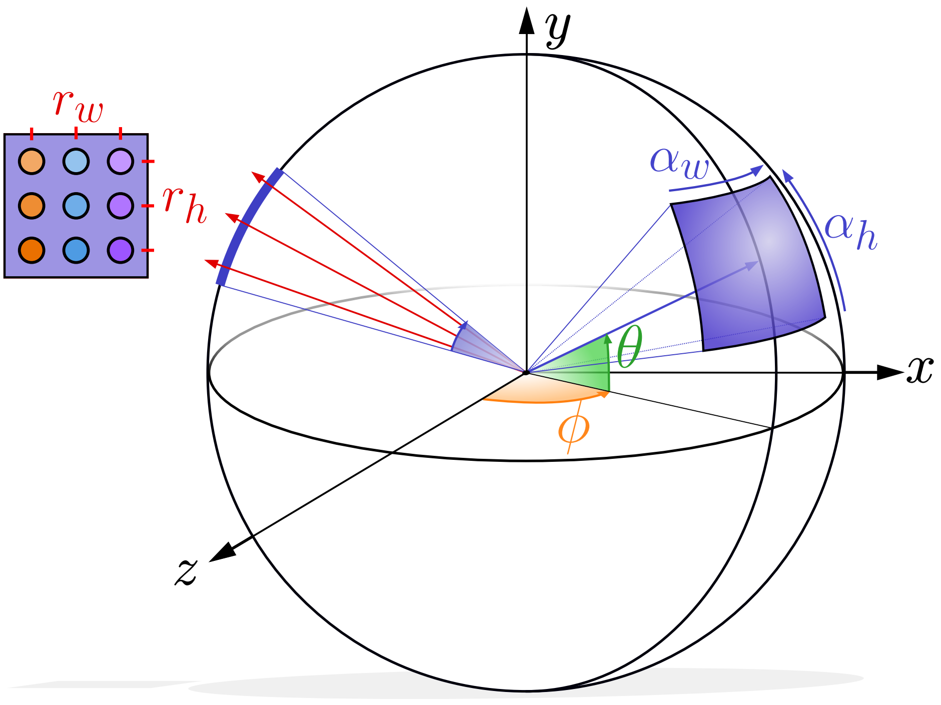


Fig. 5.4 **Spherical parametrization of EquiConvs**. The spherical kernel, defined by its angular size ($\alpha_w \times \alpha_h$) and resolution ($r_w \times r_h$), is convolved around the sphere with angles ϕ and θ.

where $\vec{V}^C = (v_x^C, v_y^C, v_z^C)$ is computed as in (5.4) with the corresponding ceiling point in the image. Notice that with P^C we have the information we were missing to recover the ceiling plane.

Limitations of CFL: We directly join corners from left to right, meaning that our model would not work if any wall is occluded because of the convexity of the scene. In those particular cases, the joining process should follow a different order. Fernandez-Labrador et al. (2018b) proposes a geometry-based post-processing that could alleviate this problem, but its cost is high and it needs the Manhattan World assumption. The addition of this post-processing into our work, in any case, could be done similarly to Fernandez-Labrador et al. (2018a).

5.4 Equirectangular Convolutions

Spherical images are receiving an increasing attention due to the growing number of omnidirectional sensors in drones, robots and autonomous cars. A naïve application of convolutional networks to a equirectangular projection, is not, in principle, a good choice due to the space-varying distortions introduced by such projection.

In this section we present a convolution that we name EquiConv, which is defined in the spherical domain instead of the image domain and it is implicitly invariant to equirectangular representation distortions. The kernel in EquiConvs is defined as a spherical surface patch –see Figure 5.4. We parametrize its receptive field by the angles α_w and α_h. Thus, we directly define a convolution over the field of view. The kernel is rotated and applied along the sphere and its position is defined by the spherical coordinates (ϕ and θ in the figure) of its center. Unlike standard kernels, that are parameterized by their size $k_w \times k_h$, with EquiConvs we define the angular size ($\alpha_w \times \alpha_h$) and resolution ($r_w \times r_h$). In practice, we keep the aspect ratio, $\frac{\alpha_w}{r_w} = \frac{\alpha_h}{r_h}$, and we use square kernels, so we will refer the field of view as α ($\alpha_w = \alpha_h$) and the resolution as r ($r_w = r_h$) respectively from now on. In this work, we choose values of resolution and field of view to be the same as the image.

5.4.1 EquiConvs Details

In Dai et al. (2017b), they introduce deformable convolutions by learning additional offsets from the preceding feature maps. Offsets are added to the regular kernel locations in the Standard Convolution enabling free form deformation of the kernel.

Inspired by this work, we deform the shape of the kernels according to the geometrical priors of the equirectangular image projection. To do that, we generate offsets that are not learned but fixed given the spherical distortion model and constant over the same horizontal locations. Here, we describe how to obtain the distorted pixel locations from the original ones.

Let us define $(u_{0,0}, v_{0,0})$ as the pixel location on the equirectangular image where we apply the convolution operation (*i.e.* the image coordinate where the center of the kernel is located). First, we define the coordinates for every element in the kernel and afterwards we rotate them to the point of the sphere where the kernel is being applied. We define each point of the kernel as

$$\hat{p}_{ij} = \begin{bmatrix} \hat{x}_{ij} \\ \hat{y}_{ij} \\ \hat{z}_{ij} \end{bmatrix} = \begin{bmatrix} i \\ j \\ d \end{bmatrix}, \tag{5.7}$$

where i and j are integers in the range $[-\frac{r-1}{2}, \frac{r-1}{2}]$ and d is the distance from the center of the sphere to the kernel grid. In order to cover the field of view α,

$$d = \frac{r}{2\tan(\frac{\alpha}{2})}. \tag{5.8}$$

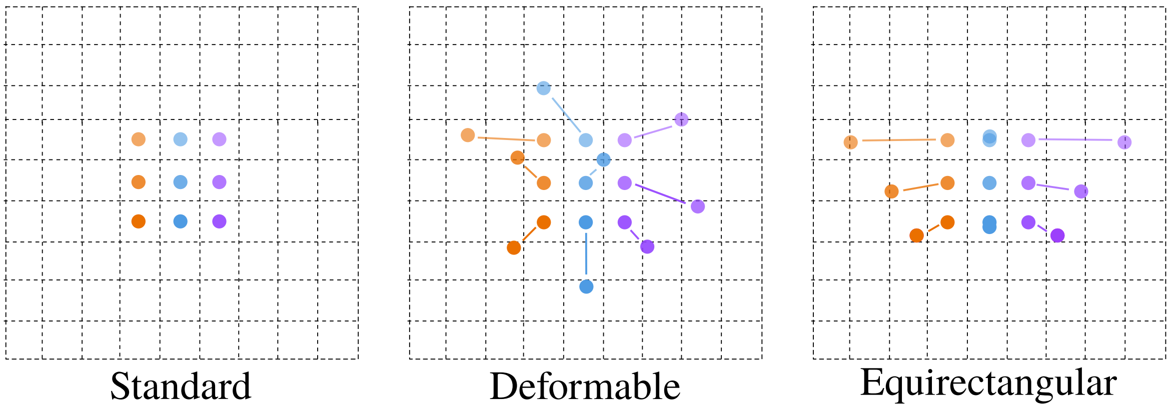


Fig. 5.5 **Effect of offsets on a** 3×3 **kernel**. Left: Regular kernel in Standard Convolution. Center: Deformable kernel in Dai et al. (2017b). Right: Spherical surface patch in EquiConvs.

We project each point into the sphere surface by normalizing the vectors, and rotate them to align the kernel center to the point where the kernel is applied.

$$p_{ij} = \begin{bmatrix} x_{ij} \\ y_{ij} \\ z_{ij} \end{bmatrix} = R_y(\phi_{0,0}) R_x(\theta_{0,0}) \frac{\hat{p}_{ij}}{|\hat{p}_{ij}|}, \tag{5.9}$$

where $R_a(\beta)$ stands for a rotation matrix of an angle β around the a axis. $\phi_{0,0}$ and $\theta_{0,0}$ are the spherical angles of the center of the kernel –see Figure 5.4, and are defined as

$$\phi_{0,0} = (u_{0,0} - \frac{W}{2}) \frac{2\pi}{W} \quad ; \quad \theta_{0,0} = -(v_{0,0} - \frac{H}{2}) \frac{\pi}{H}, \tag{5.10}$$

where W and H are, respectively, the width and height of the equirectangular image in pixels. Finally, the rest of elements are back-projected to the equirectangular image domain. First, we convert the unit sphere coordinates to latitude and longitude angles:

$$\phi_{ij} = \arctan(\frac{x_{ij}}{z_{ij}}) \quad ; \quad \theta_{ij} = \arcsin(y_{ij}). \tag{5.11}$$

And then, to the original 2D equirectangular image domain:

$$u_{ij} = (\frac{\phi_{ij}}{2\pi} + \frac{1}{2}) W \quad ; \quad v_{ij} = (-\frac{\theta_{ij}}{\pi} + \frac{1}{2}) H. \tag{5.12}$$

In Figure 5.5 we show how these offsets are applied to a regular kernel; and in Figure 5.6 three kernel samples on the spherical and on the equirectangular images.

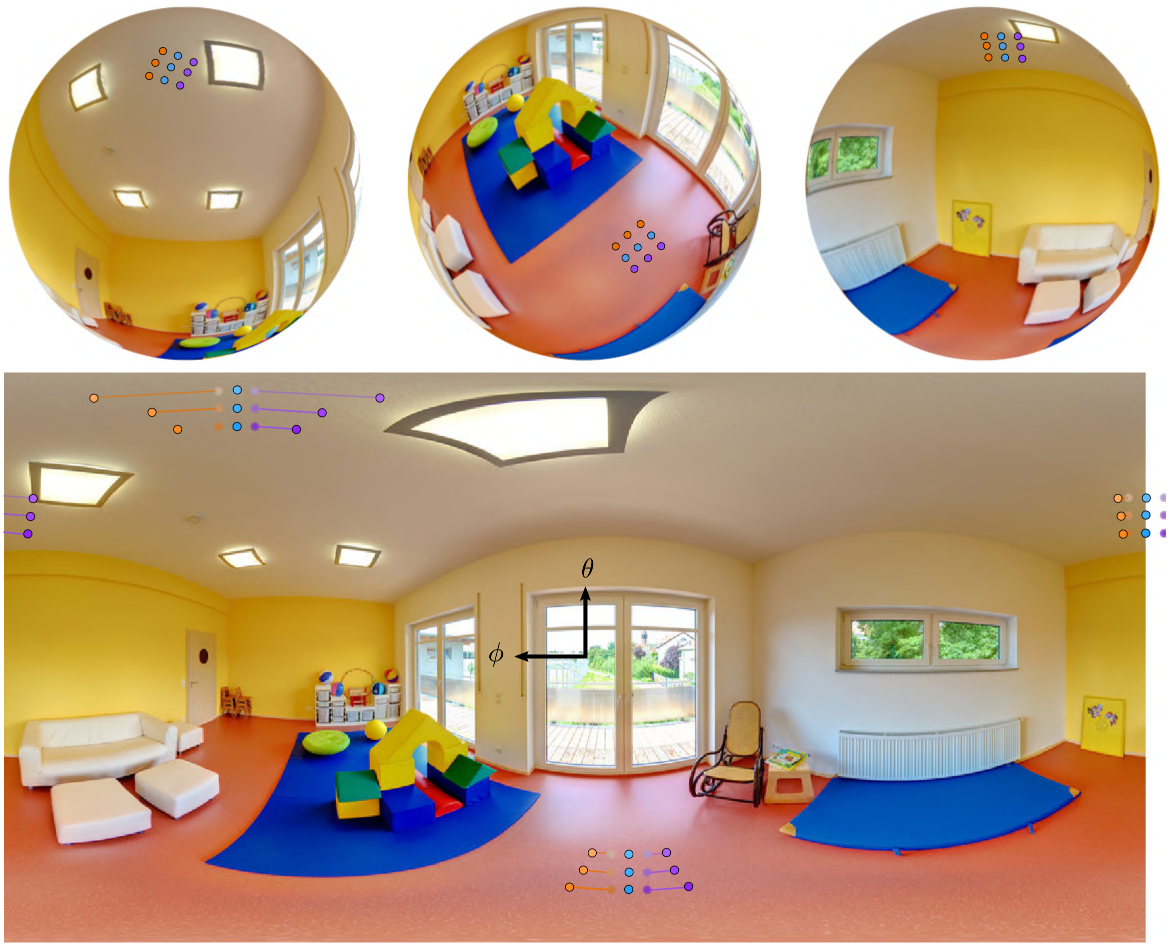


Fig. 5.6 **EquiConvs on spherical images.** We show three kernel positions to highlight the differences between the offsets. As we approach to the poles (larger θ angles) the deformation of the kernel on the equirectangular image is bigger, in order to reproduce a regular kernel on the sphere surface. Additionally, with EquiConvs, we do not use padding when the kernel is on the border of the image since offsets take the points to their correct position on the other side of the 360° image.

5.5 Experiments

We present a set of experiments to evaluate CFL using both Standard Convolutions (StdConvs) and the proposed Equirectangular Convolutions (EquiConvs). We do not only analyze the corner maps predicted by our model, but also the impact of each algorithmic component through ablation studies. We report the performance of our proposal in two different datasets, and show qualitative 2D and 3D models of different indoor scenes.

5.5.1 Datasets

We use two public datasets that comprise several indoor scenes, SUN360 (Xiao et al., 2012) and Stanford (2D-3D-S) Armeni et al. (2017) in equirectangular projection (360°). The former is used for ablation studies, and both are used for comparison against several state-of-the-art baselines.
SUN360 (Xiao et al., 2012): We use ∼500 bedroom and livingroom panoramas from this dataset labeled by Zhang et al. (2014). We use these labels but, since all panoramas were labeled as box-type rooms, we hand-label and substitute 35 panoramas representing more faithfully the actual shapes of the rooms. We split the raw dataset in 85% training scenes and 15% test scenes randomly by making sure that there were rooms of more than 4 walls in both partitions.
Stanford 2D-3D-S (Armeni et al., 2017): This dataset contains more challenging scenarios like cluttered laboratories or corridors. In Zou et al. (2018), they use areas 1, 2, 4, 6 for training, and area 5 for testing. For our experiments we use same partitions and the ground truth provided by them.

5.5.2 Implementation details

The input to the network is a single panoramic RGB image of resolution 256×128. The outputs are, on the one hand, the room layout edge map and on the other hand, the corner map, both of them at resolution 128×64. A widely used strategy to improve generalization of neural networks is data augmentation. We apply random erasing, horizontal mirroring as well as horizontal rotation from 0° to 360° of input images during training. The weights are all initialized using ResNet-50 (He et al., 2016) trained on ImageNet (Russakovsky et al., 2015). For *CFL EquiConvs* we use the same kernel resolutions and field of views as in ResNet-50. This means that for a standard 3×3 kernel applied to a W×H feature map, $r = 3$ and $\alpha = r\frac{fov}{W}$, where $fov = 360°$ for panoramas. We minimize the cross-entropy loss using Adam (Kingma and Ba, 2014), regularized by penalizing the loss with the sum of the L2

Conv.	IP	EM	Corners *IoU*	*Acc*	*P*	*R*	F_1
			: 1	: 1	: 1	: 1	: 1
StdConvs	-	-	0.519	0.978	0.611	0.763	0.675
StdConvs	-	✓	0.531	0.979	0.639	0.749	0.685
StdConvs	✓	✓	0.569	0.982	0.684	0.761	0.718
EquiConvs	-	-	0.485	0.972	0.551	0.786	0.642
EquiConvs	-	✓	0.536	0.980	0.649	0.744	0.690
EquiConvs	✓	✓	**0.580**	**0.983**	**0.697**	**0.762**	**0.726**
			bigger is better				

Table 5.3 **Ablation study on SUN360 dataset.** We show results for both Standard Convolutions (StdConvs) and our proposed Equirectangular Convolutions (EquiConvs) with some modifications: Using or not intermediate predictions (IP) in the decoder and edge map predictions (EM).

of all weights. The initial learning rate is $2.5e^{-4}$ and is exponentially decayed by a rate of 0.995 every epoch. We apply a dropout rate of 0.3.

The network is implemented using TensorFlow (Abadi et al., 2016) and trained and tested in a NVIDIA Titan X. The training time for StdConvs is around 1 hour and the test time is 0.31 seconds per image. For EquiConvs, training takes 3 hours and test around 3.32 seconds per image.

5.5.3 Network's output evaluation

We measure the quality of our predicted probability corner maps using five standard metrics: intersection over union IoU, precision P, recall R, F1 Score F_1 and accuracy Acc. Table 5.3 summarizes our results and allows us to answer the following questions:

What are the effects of different convolutions? As one would expect, EquiConvs, aware of the distortion model, learn in a non-distorted generic feature space achieving accurate predictions, like StdConvs on conventional images (Lee et al., 2017). Distortion understanding, additionally, gives the network other advantages. While StdConvs learn strong bias correlation between features and distortion patterns (*e.g.* ceiling line on the top of the image or clutter in the mid-bottom), EquiConvs are invariant to that. For this reason, the performance of EquiConvs does not degrade when varying the camera DOF pose – see Section 5.5.4. Additionally, EquiConvs allow to directly leverage networks pre-trained on conventional images. Specifically, this translates into a faster convergence, which is desirable as, to date, 360° datasets contain far less images than datasets with conventional images. In

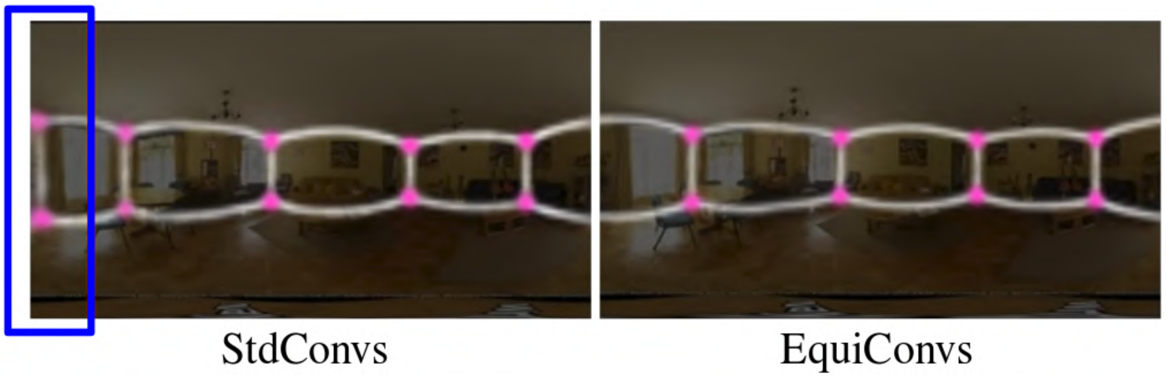


Fig. 5.7 **EquiConvs show more consistent qualitative results** whereas StdConvs simply do not understand that the image wraps around the sphere, losing the continuous context that these images provide.

omnidirectional images, the right and the left edge are the same spot in reality so, another strength of EquiConvs lie in the fact that we can avoid padding when the kernel reaches the border of the image since offsets take the points to their correct position on the other side of the 360° image. This allows the model to understand the continuity of the scene. StdConvs, instead, simply do not understand that the image wraps around the sphere. As a consequence, in most cases when corners approach the borders, StdConvs predict these corners twice, i.e. at both ends, or the edges at one side would not coincide with the edges at the other side. This effect is highlighted in Figure 5.7 and further demonstrated in the supplementary video.
How can we refine predictions? There are some techniques that we can use in order to obtain more accurate and refined predictions. Here, we make pyramid preliminary predictions in the decoder and iteratively refine them, by feeding them back to the network, until the final prediction. Also, although we only use the corner map to recover the layout of the room, we train the network to additionally predict edge maps as an auxiliary task. This is another representation of the same task that ensures that the network learns to exploit the relationship between both outputs, *i.e.*, the network learns how edges intersect between them generating the corners. The improvement is shown in the Table 5.3.
How can we deal with occlusions? We do Random Erasing Data Augmentation. This operation randomly selects rectangles in the training images and removes its content, generating various levels of virtual occlusion. In this manner we simulate real situations where objects in the scene occlude the corners of the room layout, and force the network to learn context-aware features to overcome this challenging situation. Figure 5.8 illustrates this strategy with an example.
Is it possible to relax the scene assumptions while keeping a good performance? By avoiding constrained Manhattan 3D layout predictions we not only achieve better results

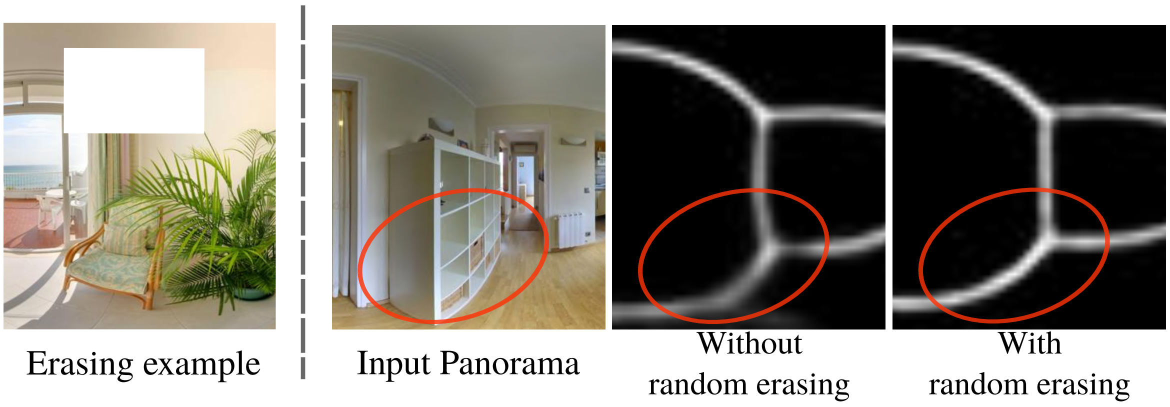


Fig. 5.8 **Augmenting the data with virtual occlusions.** Left: Image with erased pixels. Right: Input panorama and predictions without and with pixel erasing. Notice the improvement by random erasing.

compared with current arts, but also we save in computation. Additionally, our model overcomes the classic box-room simplification (four-walls room setups), even if we still have a largely unbalanced dataset after labeling some panoramas more accurately to their actual shape. We address this problem by choosing a batch size of 16 and forcing it to always include one non-box sample. This favors the learning of more complex rooms despite having few examples.

		F_1	Acc	IoU
Trans	StdConvs	55.32 ± 8.23	95.46 ± 1.3	39.135 ± 7.82
	EquiConvs	$\mathbf{59.55 \pm 8.95}$	$\mathbf{96.21 \pm 1.14}$	$\mathbf{43.47 \pm 8.83}$
Rot x	StdConvs	45.89 ± 14.72	93.44 ± 3.18	31.26 ± 12.83
	EquiConvs	$\mathbf{46.2 \pm 15.1}$	$\mathbf{94.43 \pm 2.18}$	$\mathbf{31.625 \pm 13.41}$
Rot y	StdConvs	72.28 ± 2.7	98.21 ± 0.21	57.54 ± 3.25
	EquiConvs	$\mathbf{72.96 \pm 2.02}$	$\mathbf{98.29 \pm 0.14}$	$\mathbf{58.44 \pm 2.44}$

Table 5.4 **Robustness analysis**. Values represent the mean value (*bigger is better*) $\pm$ standard deviation (*smaller is better*) in %. We apply three types of transformations to the panoramas: **translations** in y dependant on the room height from $-0.3h$ to $0.3h$, **rotations** in x from $-30°$ to $+30°$ and **rotations** in y from $0°$ to $360°$. We do not use these images for training but just for testing in order to show the generalization capabilities of both models.

5.5.4 Robustness analysis

We test our model with previously unseen images where the camera viewpoint is different from that in the training set. The distortion in equirectangular projection is location dependent,

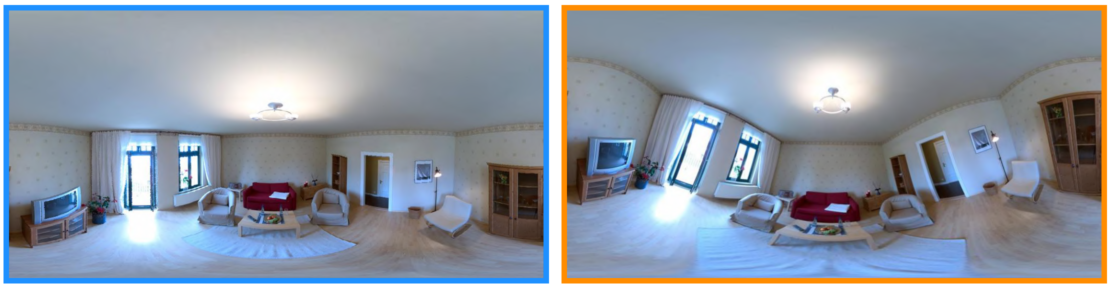


Fig. 5.9 **Synthetic images for robustness analysis.** Here we show two examples of panoramas generated with upward **translation** in y and **rotation** in x respectively.

specifically, it depends on the polar angle θ. Since EquiConvs are invariant to this distortion, it is interesting to see how modifications in the camera extrinsic parameters (**translation** and **rotation**) affect the model performance using EquiConvs against StdConvs. When we generate translations (over vertical axis y) and rotations (over horizontal axis x), the shape of the layout is modified by the distortion, losing its characteristic pattern (which StdConvs use in its favor).

Since standard datasets have a strong bias when referring to camera pose and rotation, we synthetically render these transformations along our test set. The **rotation** is trivial as we work on the spherical domain. As the complete 3D dense model of the rooms is not available, the **translation** simulation is performed by using the existing information, ignoring occlusions produced by viewpoint changes. Nevertheless, as we do not work with wide translations the effect is minimal and images are realistic enough to prove the point we want to highlight (see Figure 5.9). Refer to supplementary material for more details. For both experiments, we uniformly sample from a minimum to a maximum transformation and calculate the mean and standard deviation for all the metrics. What we see in Table 5.4 is that we obtain higher mean values by using EquiConvs. This means that this EquiConvs make the model more robust and generalizable to real life situations, not covered in the datasets, *e.g.* panoramas taken by hand, drones or small robots.

We also quantitatively analyzed the robustness of the model to **rotation** over the vertical axis y. Even though this rotation do not distort the shape of the layout like the previous extrinsic parameters, the incapability of StdConvs to wrap around the sphere and understand the continuity of the scene was a frequent source of failure as we showed in Figure 5.7 and the supplementary video. Table 5.4 compare both convolutions, where the numbers represent the mean of the results obtained from each panorama after doing all possible rotations (from 0° to 360° horizontally) and computing mean and standard deviation per panorama. Results show that EquiConvs not only have better overall performance, but the standard deviation is much smaller since there are no special cases that cause failure due to lack of continuity in the borders.

5.5.5 3D Layout comparison

We evaluate our layout predictions using three standard metrics, 3D intersection over union $3DIoU$, corner error CE and pixel error PE, and compare ourselves against four approaches from the state of the art Zhang et al. (2014), Zou et al. (2018), Fernandez-Labrador et al. (2018b), Yang et al. (2018). Pano2CAD Xu et al. (2017) has no source code available nor evaluation of layouts, making direct comparison difficult. The pixel error metric given by Zou et al. (2018) only distinguishes between ceiling, floor and walls, PE^{SS}. Instead our proposed segmented mask distinguish between ceiling, floor and each wall separately, PE^{CS}, which is more informative since it also has into account errors in wall-wall boundaries. For all experiments, only SUN360 dataset is used for training. Table 5.5 shows the performance of our proposal testing on both datasets, SUN360 and Stanford 2D-3D. Results are averaged across all images. It can be seen that our approach outperforms the state of the art clearly, in all the metrics.

It is worth mentioning that our approach, not only obtains better accuracy but also it recovers shapes more faithful to the real ones, since it can handle non box-type room designs with few training examples. In Table 5.6 we show that, apart from achieving better localization of layout corners, our model is much faster. Our full method with EquiConvs takes 3.47 seconds (0.3 fps) to process one room and with StdConvs just 0.46 seconds (2.2 fps), which is a major advantage considering the aforementioned applications of layout recovery need to be real-time (robot navigation, AR/VR).

5.5.6 Extra Qualitative Results

Here we show additional qualitative results of our recovered layouts in SUN360 and Stanford 2D-3D datasets.

Figures 5.11 and 5.12 collect examples in SUN360 dataset and show indoor scenes with different geometries, not only cuboid shapes. Figure 5.13 shows examples in Stanford 2D-3D dataset. Panoramas in this dataset do not cover full view vertically and the indoor scenes represent more challenging scenarios like cluttered laboratories or corridors.

Test	Method	*3DIoU*	*CE*	PE^{SS}	PE^{CS}
SUN360	PanoContext (Zhang et al., 2014)	67.22	1.60	4.55	10.34
	Fernandez-Labrador et al. (2018b)	-	-	-	7.26
	LayoutNet (Zou et al., 2018)	74.48	1.06	3.34	-
	DuLa-Net (Yang et al., 2018)	77.42	-	-	-
	CFL StdConvs	78.79	0.79	**2.49**	3.33
	CFL EquiConvs	**78.87**	**0.75**	2.6	**3.03**
Std.2D3D	Fernandez-Labrador et al. (2018b)	-	-	-	12.1
	LayoutNet (Zou et al., 2018)	64.56	1.44	5.16	-
	CFL StdConvs	65.13	**1.44**	**4.75**	**6.05**
	CFL EquiConvs	**65.23**	1.64	5.52	7.11
			smaller is better		

Table 5.5 **Layout results** on both datasets (in %), training on SUN360 data. *SS*: Simple Segmentation (3 categories): ceiling, floor and walls (Zou et al., 2018). *CS*: Complete Segmentation: ceiling, floor, wall$_1$,..., wall$_n$ (Fernandez-Labrador et al., 2018b). Observe how our method outperforms all the baselines in all the metrics.

Method	Computation Time (s)
PanoContext (Zhang et al., 2014)	> 300
LayoutNet (Zou et al., 2018)	44.73
DuLa-Net (Yang et al., 2018)	13.43
CFL EquiConvs	3.47
CFL StdConvs	**0.46**

Table 5.6 **Average computing time per image.** Every approach is evaluated using NVIDIA Titan X and Intel Xeon 3.5 GHz (6 cores) except DuLa-Net, evaluated using NVIDIA 1080Ti GPU. Our end-to-end method is more than 100 times faster than other methods.

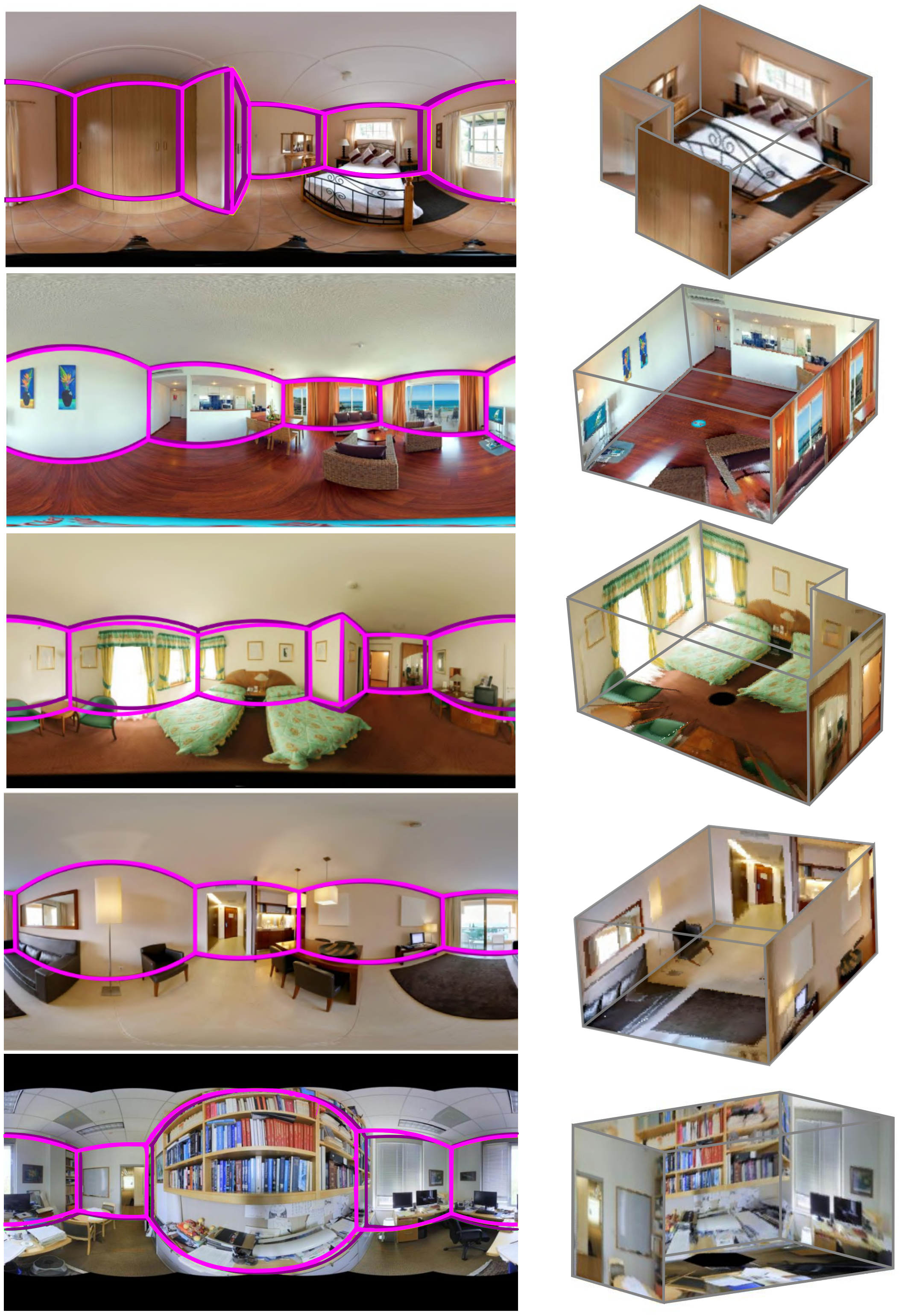


Fig. 5.10 Layout predictions (light magenta) and ground truth (dark magenta) for **complex room geometries**.

5.6 Conclusions

In this chapter we present CFL, the first end-to-end algorithm for layout recovery in 360° images. Our experimental results demonstrate that our predicted layouts are clearly more accurate than the state of the art. Additionally, the removal of extra pre- and post-processing stages makes our method much faster than other works. Finally, being entirely data-driven relaxes the geometric assumptions that are commonly used in the state of the art and limits their usability in complex geometries. We present two different variants of CFL. The first one, implemented using Standard Convolutions, reduces the computation in 100 times and it is very suitable for images taken with a tripod (recommended if the time is a critical issue). The second one uses our proposed implementation of Equirectangular Convolutions that adapt their shape to the equirectangular projection of the spherical image (recommended if looking for robustness and better generalization). This proves to be more robust to translations and rotations of the camera making it ideal for panoramas taken by a hand-held camera.

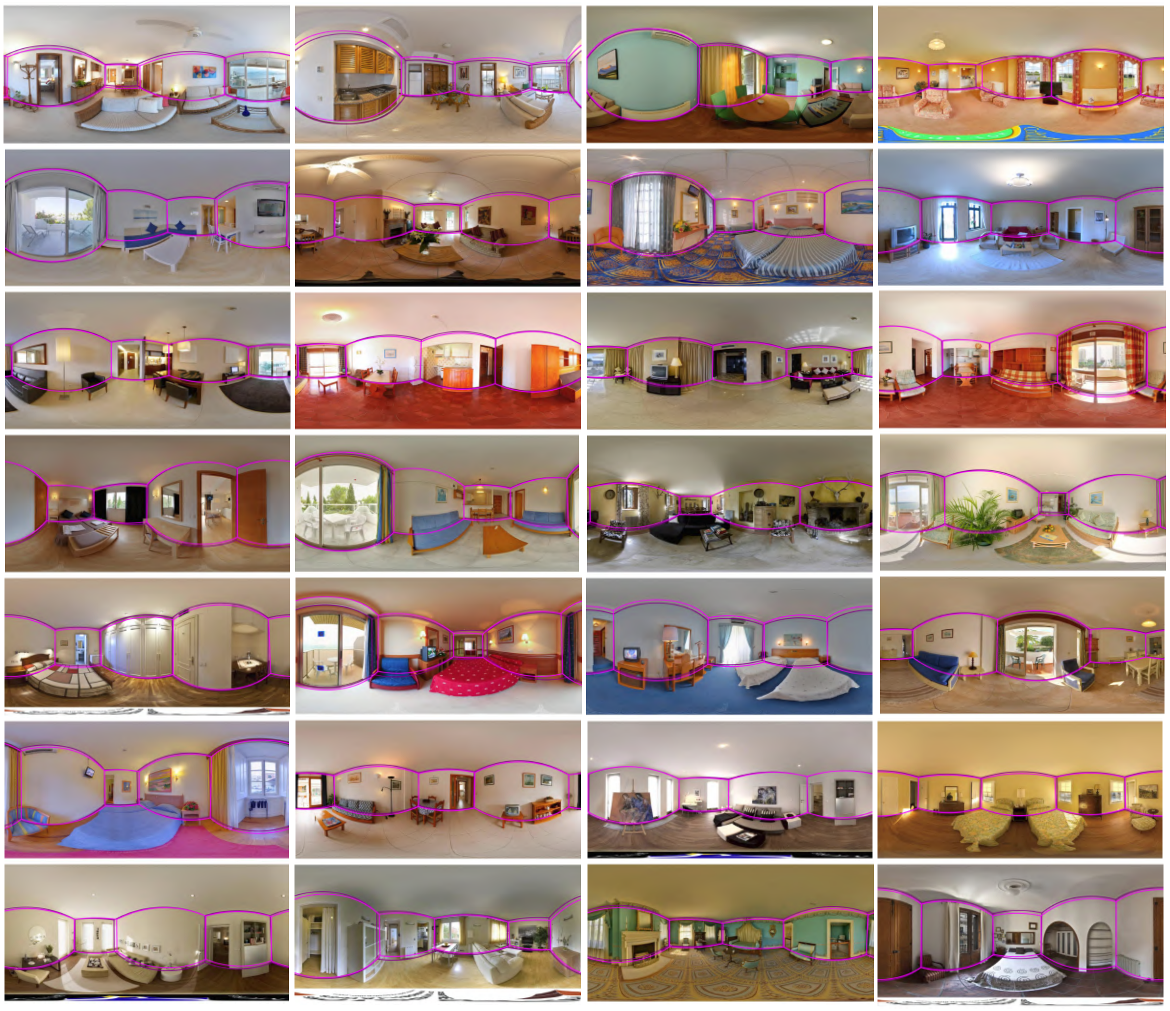


Fig. 5.11 Layout predictions (light magenta) and ground truth (dark magenta) on the SUN360 annotation dataset (Xiao et al., 2012). Best viewed in color.

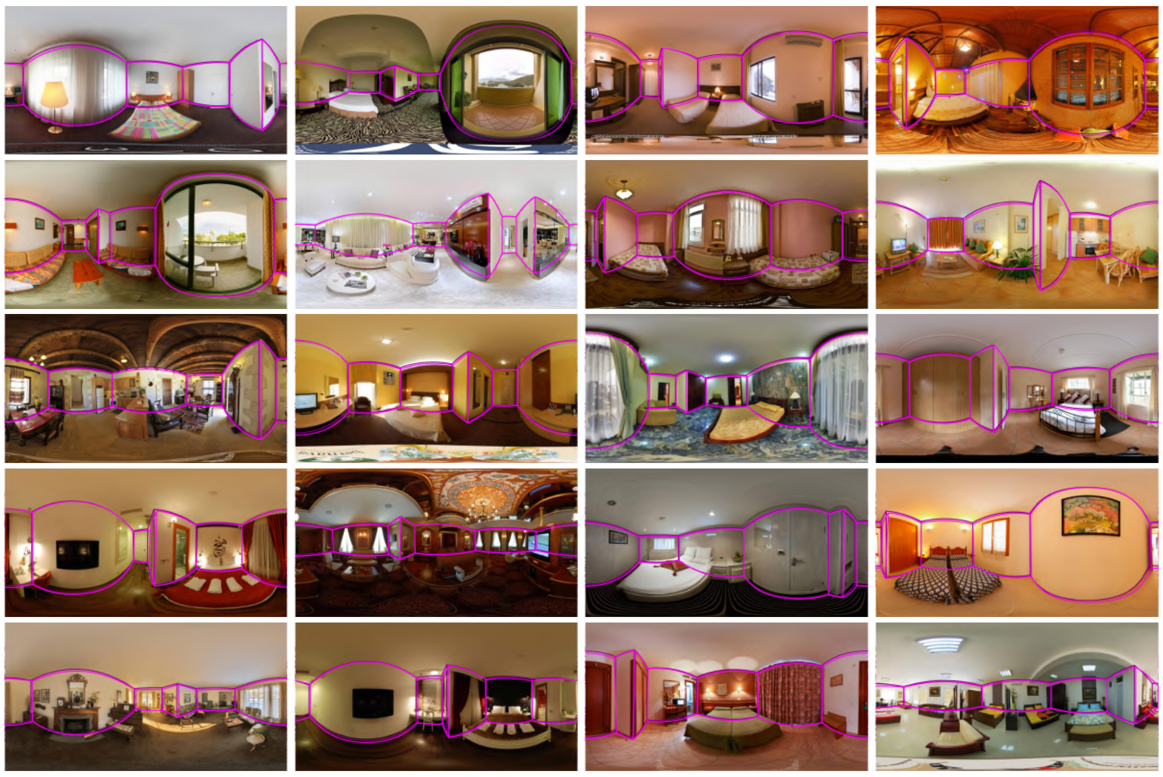


Fig. 5.12 Layout predictions (light magenta) and ground truth (dark magenta) for **complex room geometries** on the SUN360 annotation dataset (Xiao et al., 2012). Best viewed in color.

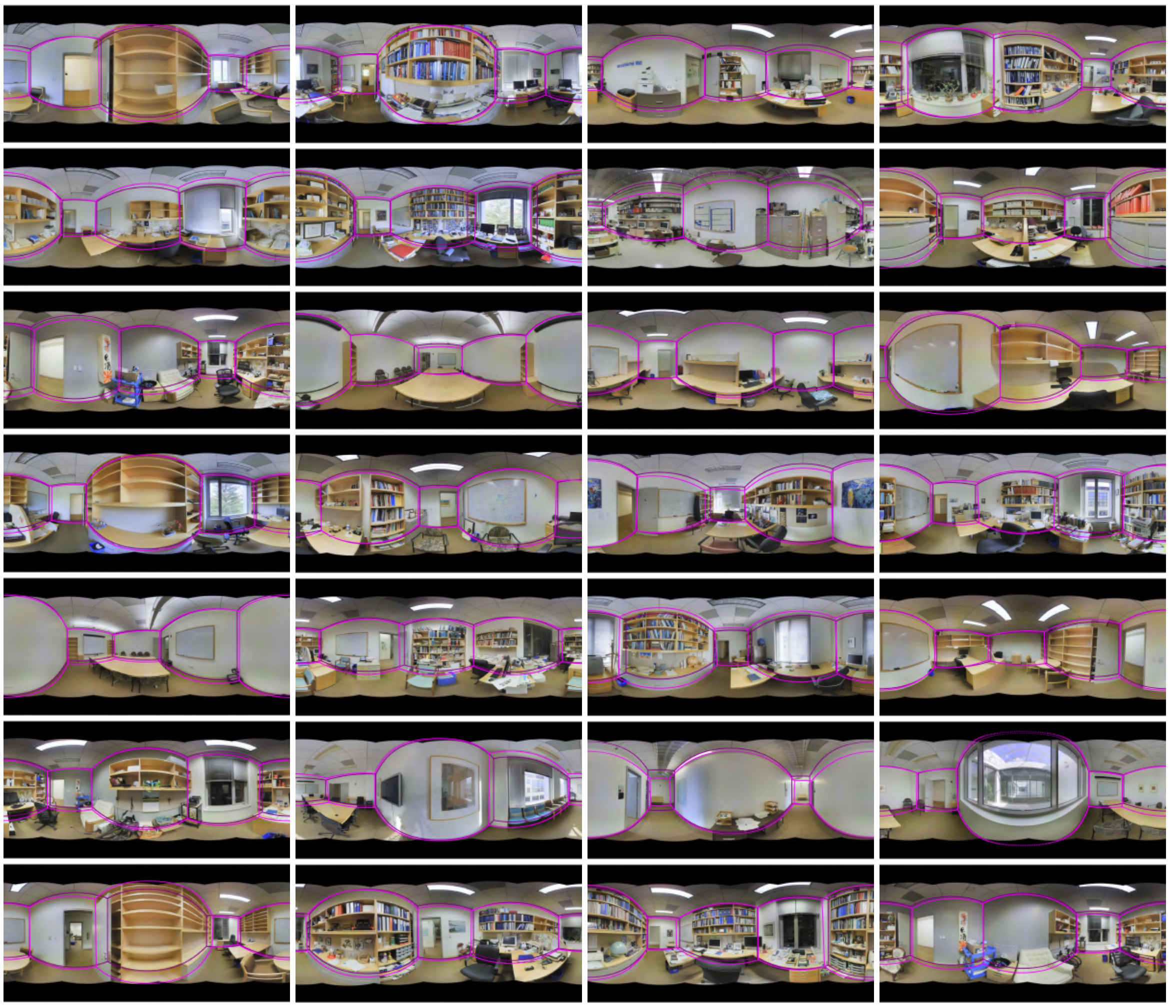


Fig. 5.13 Layout predictions (light magenta) and ground truth (dark magenta) on the Stanford 2D-3D annotation dataset (Armeni et al., 2017). Best viewed in color.

Chapter 6

Monocular and RGB-D SLAM on Dynamic Environments

The assumption of scene rigidity is typical in SLAM algorithms. Such a strong assumption limits the use of most visual SLAM systems in populated real-world environments, which are the target of several relevant applications like service robotics or autonomous vehicles.

In this chapter we present DynaSLAM, a visual SLAM system that, building on ORB-SLAM2 (Mur-Artal and Tardós, 2017), adds the capabilities of dynamic object detection and background inpainting. DynaSLAM is robust in dynamic scenarios for monocular, stereo and RGB-D configurations. We are capable of detecting the moving objects either by multi-view geometry, deep learning or both. Having a static map of the scene allows inpainting the frame background that has been occluded by such dynamic objects.

We evaluate our system in public monocular, stereo and RGB-D datasets. We study the impact of several accuracy/speed trade-offs to assess the limits of the proposed methodology. DynaSLAM outperforms the accuracy of standard visual SLAM baselines in highly dynamic scenarios. And it also estimates a map of the static parts of the scene, which is a must for long-term applications in real-world environments.

6.1 Introduction

SLAM is a prerequisite for many robotic applications, for example collision-less navigation. SLAM techniques estimate jointly a map of an unknown environment and the robot pose within such map, only from the data streams of its on-board sensors. The map allows the robot to continually localize within the same environment without accumulating drift. This is in contrast to odometry approaches that integrate the incremental motion estimated within a local window and are unable to correct the drift when revisiting places.

Visual SLAM, where the main sensor is a camera, has received a high degree of attention and research efforts over the last years. The minimalistic solution of a monocular camera has practical advantages with respect to size, power and cost, but also several challenges such as the unobservability of the scale or state initialization. By using more complex setups, like stereo or RGB-D cameras, these issues are solved and the robustness of visual SLAM systems can be greatly improved.

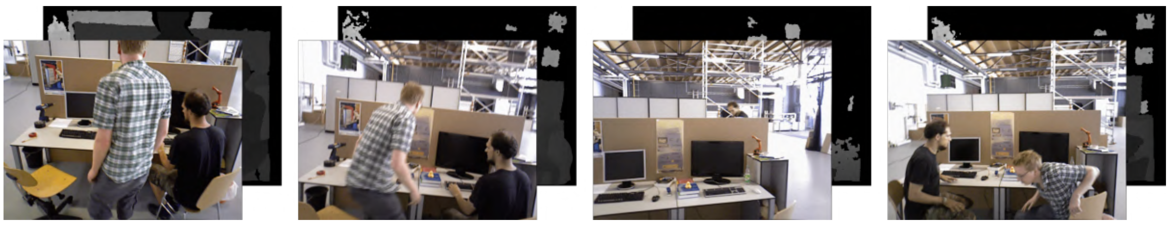
The research community has addressed SLAM from many different angles. However, the vast majority of the approaches and datasets assume a static environment. As a consequence, they can only manage small fractions of dynamic content by classifying them as outliers to such static model. Although the static assumption holds for some robotic applications, it limits the applicability of visual SLAM in many relevant cases, such as intelligent autonomous systems operating in populated real-world environments over long periods of time.

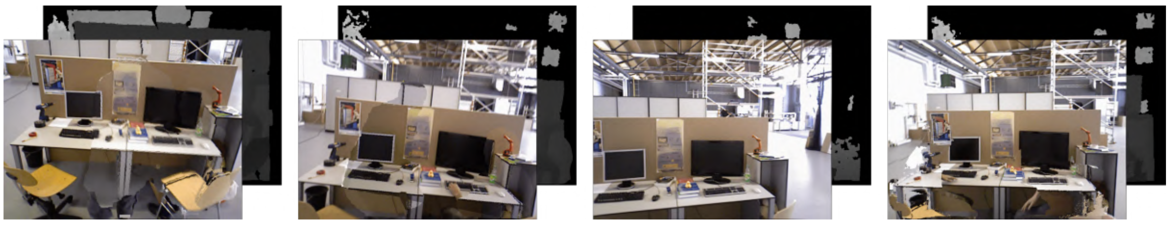
Visual SLAM can be classified into feature-based methods (Klein and Murray, 2007, Mur-Artal et al., 2015), that rely on salient points matching and can only estimate a sparse reconstruction; and direct methods (Stühmer et al., 2010, Newcombe et al., 2011, Graber et al., 2011), which are able to estimate in principle a completely dense reconstruction by the direct minimization of the photometric error and TV regularization. Some direct methods focus on the high-gradient areas estimating semi-dense maps (Engel et al., 2014, 2017).

None of the above methods, considered the state of the art, address the very common problem of dynamic objects in the scene, *e.g.*, people walking, bicycles or cars. Detecting and dealing with dynamic objects in visual SLAM reveals several challenges for both mapping and tracking, including:

1. How to detect such dynamic objects in the images to:
   (a) Prevent the tracking algorithm from using matches that belong to dynamic objects.
   (b) Prevent the mapping algorithm from including moving objects as part of the 3D map.
2. How to complete the part of the 3D map that is temporally occluded by a moving object.

Many applications would greatly benefit from progress along these lines. Among others, augmented reality, autonomous vehicles, and medical imaging. All of them could for instance safely reuse maps from previous runs. Detecting and dealing with dynamic objects is a requisite to estimate stable maps, useful for long-term applications. If the dynamic


(a) Input RGB-D frames with dynamic content.


(b) Output RGB-D frames. Dynamic content has been removed. Occluded background has been reconstructed with information from previous views.

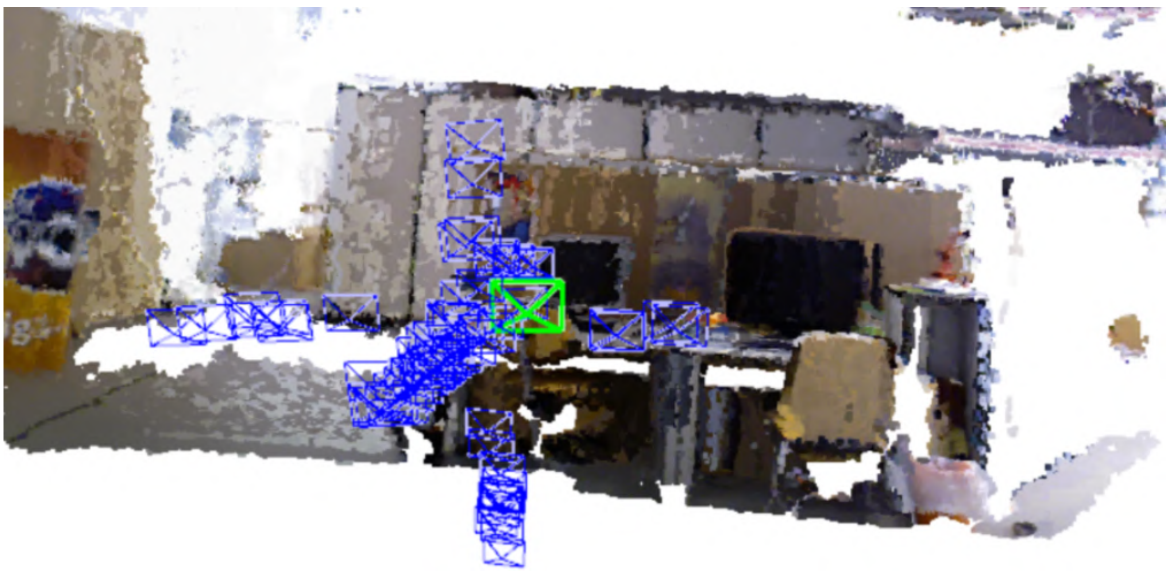

(c) Map of the static part of the scene, after removal of the dynamic objects.

Fig. 6.1 Overview of DynaSLAM results for the RGB-D case.

content is not detected, it becomes part of the 3D map, complicating its usability for tracking or relocation purposes.

In this work we propose an on-line algorithm to deal with dynamic objects in RGB-D, stereo and monocular SLAM. This is done by adding a front-end stage to the state-of-the-art ORB-SLAM2 system (Mur-Artal and Tardós, 2017), with the purpose of having a more accurate tracking and a reusable map of the scene. In the monocular and stereo cases our proposal is to use a CNN to pixel-wise segment the *a priori* dynamic objects in the frames (*e.g.*, people and cars), so that the SLAM algorithm does not extract features on them. In the

RGB-D case we propose to combine multi-view geometry models and deep-learning-based algorithms for detecting dynamic objects and, after having removed them from the images, inpaint the occluded background with the correct information of the scene (Fig. 6.1).

The rest of the chapter is structured as follows: section 6.2 discusses related work, section 6.3 gives the details of our proposal, section 6.4 details the experimental results, and section 6.5 presents the conclusions and lines for future work.

6.2 Related Work

Dynamic objects are, in most SLAM systems, classified as spurious data and therefore neither included in the map nor used for camera tracking. The most typical outlier rejection algorithms are RANSAC (*e.g.*, in ORB-SLAM (Mur-Artal et al., 2015, Mur-Artal and Tardós, 2017)) and robust cost functions (*e.g.*, in PTAM by Klein and Murray (2007)).

There are several SLAM systems that address more specifically the dynamic scene content. Within feature-based SLAM methods, some of the most relevant on dealing with dynamic scenes are the following. Tan et al. (2013) that detect changes that take place in the scene by projecting the map features into the current frame for appearance and structure validation. Wangsiripitak and Murray (2009) track known 3D dynamic objects in the scene. Similarly, Riazuelo et al. (2017) deal with human activity by detecting and tracking people. More recently, the work of Li and Lee (2017) uses depth edges points, which have an associated weight indicating its probability of belonging to a dynamic object.

Direct methods are, in general, more sensitive to dynamic objects in the scene. The most relevant works specifically designed for dynamic scenes are mentioned bellow. Alcantarilla et al. (2012) detect moving objects by means of a scene flow representation with stereo cameras. Wang and Huang (2014) segment the dynamic objects in the scene using RGB optical flow. Kim and Kim (2016) propose to obtain the static parts of the scene by computing the difference between consecutive depth images projected over the same plane. Sun et al. (2017) calculate the difference in intensity between consecutive RGB images. Pixel classification is done with the segmentation of the quantized depth image.

All the methods –both feature-based and direct ones– that map the static scene parts only from the information contained in the sequence (Mur-Artal and Tardós, 2017, Mur-Artal et al., 2015, Tan et al., 2013, Li and Lee, 2017, Alcantarilla et al., 2012, Wang and Huang, 2014, Kim and Kim, 2016, Sun et al., 2017, Concha and Civera, 2015a), fail to estimate lifelong models when an *a priori* dynamic object remains static, *e.g.*, parked cars or people sitting. On the other hand, Wangsiripitak and Murray (2009), and Riazuelo et al. (2017) would detect those *a priori* dynamic objects, but would fail to detect changes produced by

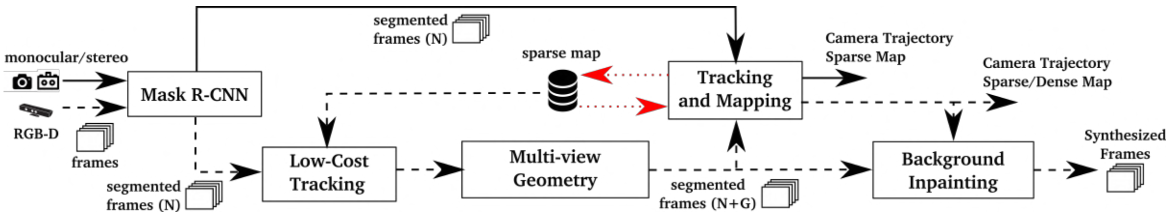


Fig. 6.2 Block diagram of our proposal. In the stereo and monocular pipeline (black continuous line) the images pass through a Convolutional Neural Network (Mask R-CNN) for computing the pixel-wise semantic segmentation of the *a priori* dynamic objects before being used for the mapping and tracking. In the RGB-D case (black dashed line) a second approach based on multi-view geometry is added for a more accurate motion segmentation, for which we need a low-cost tracking algorithm. Once the position of the camera is known (Tracking and Mapping output), we can inpaint the background occluded by dynamic objects. The red dotted line represents the data flow of the stored sparse map.

static objects, *e.g.*, a chair a person is pushing, or a ball that someone has thrown. That is, the former approach succeeds in detecting *moving* objects, and the second one in detecting several *movable* objects. Our proposal, DynaSLAM, combines multi-view geometry and deep learning in order to address both situations. Similarly, Ambrus et al. (2016) segment dynamic objects by combining a dynamic classifier and multi-view geometry.

6.3 DynaSLAM System Description

Fig. 6.2 shows an overview of our system. First of all, the RGB channels pass through a CNN that segments out pixel-wise all the *a priori* dynamic content, *e.g.*, people or vehicles.

In the RGB-D case, we use multi-view geometry to improve the dynamic content segmentation in two ways. First, we refine the segmentation of the dynamic objects previously obtained by the CNN. Second, we label as dynamic new object instances that are static most of the time (*i.e.*, detect *moving* objects that were not set to *movable* in the CNN stage).

For that purpose, it is necessary to know the camera pose, for which a low-cost tracking module has been implemented to localize the camera within the already created scene map. These segmented frames are the ones which are used to obtain the camera trajectory and the map of the scene. Notice that if the moving objects in the scene are not within the CNN classes, the multi-view geometry stage would still detect the dynamic content, but the accuracy might decrease.

Once this full dynamic object detection and localization of the camera have been done, we aim to reconstruct the occluded background of the current frame with static information from previous views. These synthetic frames are relevant for applications like augmented and virtual reality, and place recognition in lifelong mapping.

In the monocular and stereo cases, the images are segmented by the CNN so that keypoints belonging to the *a priori* dynamic objects are neither tracked nor mapped.

All the different stages are described in depth in the next subsections (6.3.1 to 6.3.5).

6.3.1 Segmentation of Potentially Dynamic Content using a CNN

For detecting dynamic objects we propose to use a CNN that obtains a pixel-wise semantic segmentation of the images. In our experiments we use Mask R-CNN (He et al., 2017), which is the state of the art for object instance segmentation. Mask R-CNN can obtain both pixel-wise semantic segmentation and the instance labels. For this work we use the pixel-wise semantic segmentation information, but the instance labels could be useful in future work for the tracking of the different moving objects. We use the TensorFlow implementation by Matterport[1].

The input of Mask R-CNN is the RGB original image. The idea is to segment those classes that are potentially dynamic or movable (person, bicycle, car, motorcycle, airplane, bus, train, truck, boat, bird, cat, dog, horse, sheep, cow, elephant, bear, zebra and giraffe). We consider that, for most environments, the dynamic objects likely to appear are included within this list. If other classes were needed, the network, trained on MS COCO (Lin et al., 2014), could be fine-tuned with new training data.

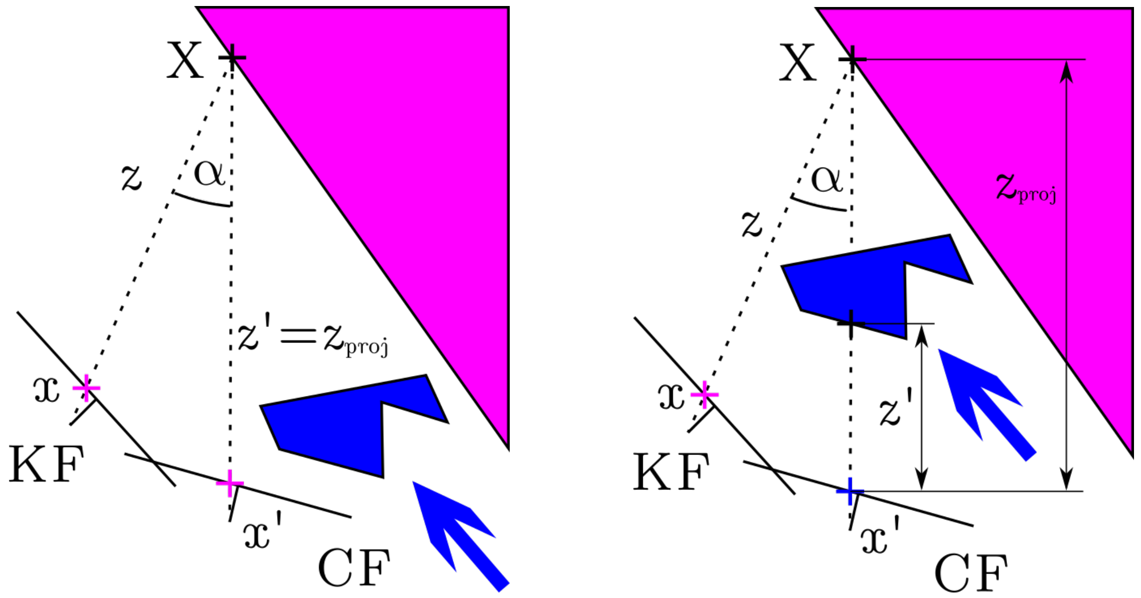
The output of the network, assuming that the input is an RGB image of size $m \times n \times 3$, is a matrix of size $m \times n \times l$, where l is the number of objects in the image. For each output channel $i \in l$ a binary mask is obtained. By combining all the channels into one, we can obtain the segmentation of all dynamic objects appearing in one image of the scene.

6.3.2 Low-Cost Tracking

After the potentially dynamic content has been segmented, the pose of the camera is tracked using the static part of the image. Because the segment contours usually become high-gradient areas, salient point features tend to appear. We do not consider the features in such contour areas.

The tracking implemented at this stage of the algorithm is a simpler and therefore computationally lighter version of the one in ORB-SLAM2 (Mur-Artal and Tardós, 2017). It projects the map features in the image frame, searches for the correspondences in the static areas of the image, and minimizes the reprojection error to optimize the camera pose.

[1] `https://github.com/matterport/Mask_RCNN`



(a) Keypoint x' belongs to a static object ($z' = z_{proj}$).

(b) Keypoint x' belongs to a dynamic object ($z' \ll z_{proj}$).

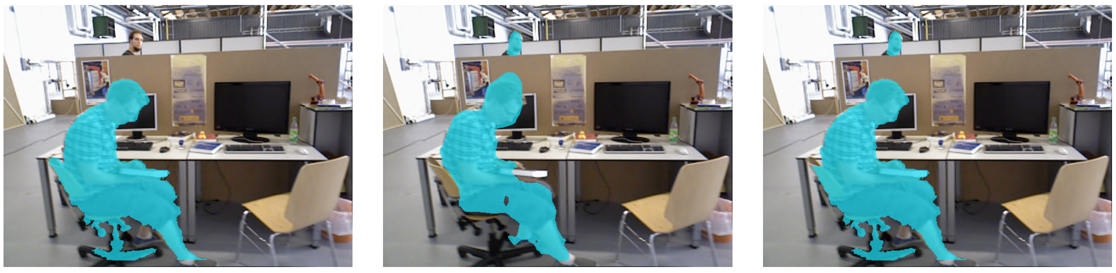
Fig. 6.3 Keypoint x from the Key Frame (KF) is projected into the Current Frame (CF) using its depth and camera pose, resulting in point x' with depth z'. The projected depth z_{proj} is then computed. A pixel is labeled as dynamic if the difference $\Delta z = z_{proj} - z'$ is greater than a threshold τ_z.

6.3.3 Segmentation of Dynamic Content using Mask R-CNN and Multi-view Geometry

By using Mask R-CNN, most of the dynamic objects can be segmented and not used for tracking and mapping. However, there are objects that cannot be detected by this approach because they are not *a priori* dynamic, but movable. Examples of the latest are a book carried by someone, a chair that someone is moving, or even furniture changes in long-term mapping. The approach utilized for dealing with these cases is detailed in this section.

For each input frame, we select the previous keyframes that have the highest overlaps. This is done by taking into account both the distance and the rotation between the new frame and each of the keyframes, similarly to Tan et al. (2013). The number of overlapping keyframes has been set to 5 in our experiments, as a compromise between computational cost and accuracy in the detection of dynamic objects.

We then compute the projection of each keypoint x from the previous keyframes into the current frame, obtaining the keypoints x', as well as their projected depth z_{proj}, computed from the camera motion. Notice that the keypoints x come from the features extractor algorithm used in ORB-SLAM2. For each keypoint, whose corresponding 3D point is X, we



(a) Using Multi-view Geometry. (b) Using Deep Learning. (c) Using Geometry and Deep Learning.

Fig. 6.4 Detection and segmentation of dynamic objects using multi-view geometry (left), deep learning (middle), and a combination of both geometric and learning methods (right). Notice that Fig. 6.4a cannot detect the person behind the desk, Fig. 6.4b cannot segment the book carried by the person, and the combination of the two (Fig. 6.4c) is the best performing.

calculate the angle between the back-projections of x and x', *i.e.*, their parallax angle α. If this angle is greater than 30°, the point might be occluded, and will be ignored from then on. We observed that, in the TUM dataset, for parallax angles greater than 30° static objects were considered as dynamic due to their viewpoint difference. We obtain the depth of the remaining keypoints in the current frame z' (directly from the depth measurement), taking into account the reprojection error, and we compare them with z_{proj}. If the difference $\Delta z = z_{proj} - z'$ is over a threshold τ_z, keypoint x' is considered to belong to a dynamic object. This idea is shown in Fig. 6.3. To set the threshold τ_z, we manually tagged the dynamic objects of 30 images within the TUM dataset, and evaluated both the precision and recall of our method for different thresholds τ_z. By maximizing the expression $0.7 \times Precision + 0.3 \times Recall$, we concluded that $\tau_z = 0.4m$ is a reasonable choice.

Some of the keypoints labeled as dynamic lay on the borders of moving objects, and might cause problems. To avoid this, we use the information given by the depth images. If a keypoint is set as dynamic, but a patch around itself in the depth map has high variance, we change the label to static.

So far, we know which keypoints belong to dynamic objects, and which ones do not. To classify all the pixels belonging to dynamic objects, we grow the region in the depth image around the dynamic pixels (Gerlach et al., 2014). An example of a RGB frame and its corresponding dynamic mask can be seen in Fig. 6.4a.

The results of the CNN (Fig. 6.4b) can be combined with those of this geometric method for full dynamic object detection (Fig. 6.4c). We can find strengths and limitations in both methods, hence the motivation for their combined use. For geometric approaches, the main problem is that initialization is not trivial because of its multi-view nature. Learning methods and their impressive performance using a single view, do not have such initialization problems. Their main limitation though is that objects that are supposed to be static can be moved, and the method is not able to identify them. This last case can be solved using multi-view consistency tests.

These two ways of facing the moving objects detection problem are illustrated in Fig. 6.4. In Fig. 6.4a we see that the person in the back, which is potentially a dynamic object, is not detected. There are two reasons for this. First, the difficulties that RGB-D cameras face when measuring the depth of distant objects. And second, the fact that reliable features lie on defined, and therefore nearby, parts of the image. Albeit, this person is detected by the deep learning method (Fig. 6.4b). Apart from this, on one hand we see in the Fig. 6.4a that not only is detected the person in the front of the image, but also the book he is holding and the chair he is sitting on. On the other hand, in the Fig. 6.4b the two people are the only objects detected as dynamic, and also their segmentation is less accurate. If only the deep learning method is used, a *floating book* would be left in the images and would incorrectly become part of the 3D map.

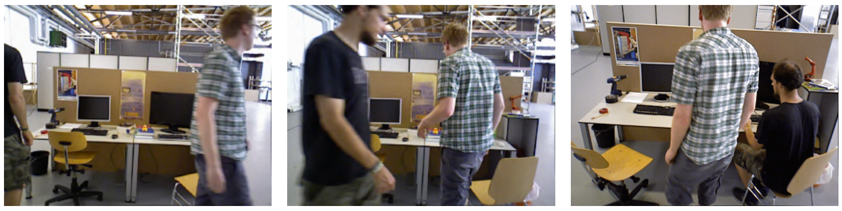
Because of the advantages and disadvantages of both methods, we consider that they are complementary and therefore their combined use is an effective way of achieving accurate tracking and mapping. In order to achieve this goal, if an object has been detected with both approaches, the segmentation mask should be that of the geometrical method. If an object has only been detected by the learning based method, the segmentation mask should contain this information too. The final segmented image of the example in the previous paragraph can be seen in the Fig. 6.4c. The segmented dynamic parts are removed from the current frame and from the map.

6.3.4 Tracking and Mapping

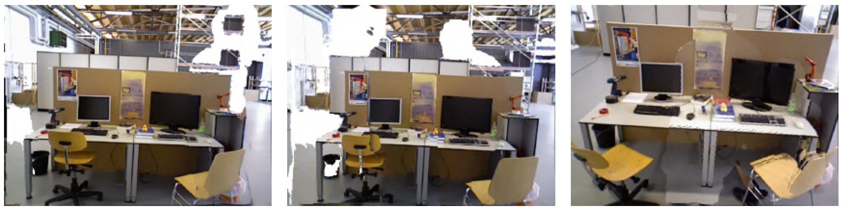
The input to this stage of the system contains the RGB and depth images, as well as their segmentation mask. We extract ORB features in the image segments classified as static. As the segment contours are high-gradient areas, the keypoints falling in this intersection have to be removed.

6.3.5 Background Inpainting

For every removed dynamic object, we aim at inpainting the occluded background with static information from previous views, so that we can synthesize a realistic image without moving content. We believe that such synthetic frames, containing the static structure of


(a) RGB original images.

(b) Depth original image.


(c) Inpainted RGB images.

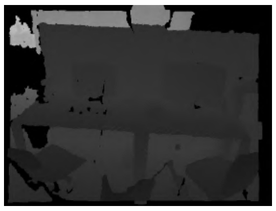

(d) Inpainted depth image.

Fig. 6.5 Qualitative results of our approach. In Fig. 6.5a we show three RGB input frames, and in Fig. 6.5c we show the output of our system, in which all dynamic objects have been detected and the background has been reconstructed. Figs. 6.5b and 6.5d show respectively the depth input and output, which has also been processed. Figure best viewed in electronic format.

the environment, are useful for applications such as virtual and augmented reality, and for relocation and camera tracking after the map is created.

Since we know the position of the previous and current frames, we project into the dynamic segments of the current frame the RGB and depth channels from a set of all the previous keyframes (the last 20 in our experiments). Some gaps have no correspondences and are left blank: some areas cannot be inpainted because their correspondent part of the scene has not appeared so far in the keyframes, or, if it has appeared, it has no valid depth information. These gaps cannot be reconstructed with geometrical methods and would need a more elaborate inpainting technique. Fig. 6.5 shows the resulting synthetic images for three input frames from different sequences of the TUM benchmark. Notice how the dynamic content has been successfully segmented and removed. Also, most of the segmented parts have been properly inpainted with information from the static background.

Another application of these synthesized frames would be the following: if the frames dynamic areas are inpainted with the static content, the system can work as a SLAM system under the staticity assumption using the inpainted images.

6.4 Experimental Results

We have evaluated our system in the public datasets TUM RGB-D and KITTI and compared to other state-of-the-art SLAM systems in dynamic environments, using when possible results published in the original papers. Furthermore we have compared our system against the original ORB-SLAM2 to quantify the improvement of our approach in dynamic scenes. In this case, the results for some sequences were not published and we have ourselves completed their evaluation. Mur-Artal and Tardós (2017) propose to run each sequence five times and show median results, to account for the non-deterministic nature of the system. We have run each sequence ten times, as dynamic objects are prone to increase this non-deterministic effect.

6.4.1 TUM Dataset

The TUM RGB-D dataset (Sturm et al., 2012b) is composed of 39 sequences recorded with a Microsoft Kinect sensor in different indoor scenes at full frame rate (30Hz). Both the RGB and the depth images are available, together with the ground-truth trajectory, the latest recorded by a high-accuracy motion-capture system. In the sequences named *sitting* (*s*) there are two people sitting in front of a desk while speaking and gesticulating, *i.e.*, there is a low degree of motion. In the sequences named *walking* (*w*), two people walk both in the background and the foreground and sit down in front of the desk. This dataset is highly dynamic and therefore challenging for standard SLAM systems. For both types of sequences *sitting* (*s*) and *walking* (*w*) there are four types of camera motions: (1) halfsphere (half): the camera moves following the trajectory of a 1-meter diameter half sphere, (2) xyz: the camera moves along the x-y-z axes, (3) rpy: the camera rotates over roll, pitch and yaw axes, and (4) static: the camera is kept static manually.

We use the absolute trajectory RMSE as the error metric for our experiments, as proposed by Sturm et al. (2012b).

The results of different variations of our system for six sequences within this dataset are shown in Table 6.1. Firstly, DynaSLAM (N) is the system in which only Mask R-CNN segments out the *a priori* dynamic objects. Secondly, in DynaSLAM (G) the dynamic objects have been only detected with the multi-view geometry method based on depth changes. Thirdly, DynaSLAM (N+G) stands for the system in which the dynamic objects have been detected combining both the geometrical and deep learning approaches. Finally, we have considered interesting to analyze the system shown in Fig. 6.6. In this case (N+G+BI), the background inpainting stage (BI) is done before the tracking and mapping. The motivation for this experiment is that, if the dynamic areas are inpainted with the static content, the

Sequence	DynaSLAM (N)	DynaSLAM (G)	DynaSLAM (N+G)	DynaSLAM (N+G+BI)
w_halfsphere	**0.025**	0.035	**0.025**	0.029
w_xyz	**0.015**	0.312	**0.015**	**0.015**
w_rpy	0.040	0.251	**0.035**	0.136
w_static	0.009	0.009	**0.006**	0.007
s_halfsphere	**0.017**	0.018	**0.017**	0.025
s_xyz	0.014	**0.009**	0.015	0.013

Table 6.1 Absolute trajectory RMSE [m] for several variants of DynaSLAM (RGB-D).

system can work as a SLAM system under the staticity assumption using the inpainted images. In this proposal, the ORB features extractor algorithm works both in the real and reconstructed areas of the frames, finding matches with the keypoints of the previously processed keyframes.

According to Table 6.1, the system (N+G) that uses learning and geometry is the most accurate one in most sequences. The improvement over (N) comes from the segmentation of *movable* objects and refinement of the dynamic segments. The system (G) has higher error because it needs motion and its segmentation is only accurate after a small delay, during which the dynamic content introduces some error in the estimation.

Adding the background inpainting stage (BI) before the localization of the camera (Fig. 6.6) usually leads to less accuracy in the tracking. The reason is that the background reconstruction is strongly correlated with the camera poses. Hence, for sequences with purely rotational motion (*rpy*, *halfsphere*), the estimated camera poses have a greater error and lead to a non-accurate background reconstruction. The background inpainting stage (BI) should be done therefore once the tracking stage is finished (Fig. 6.2). The main accomplishment of the background reconstruction is seen in the synthesis of the static images (Fig. 6.5) for applications such as virtual reality or cinematography. The DynaSLAM results shown from now on are from the best variant, that is, (N+G).

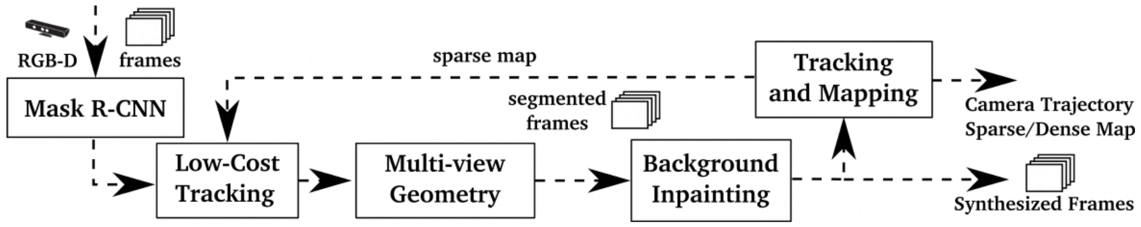


Fig. 6.6 Block diagram of RGB-D DynaSLAM (N+G+BI).

Sequence	ORB-SLAM2 (RGB-D)	DynaSLAM (N+G) (RGB-D)		
	median	median	min	max
w_halfsphere	0.351	**0.025**	0.024	0.031
w_xyz	0.459	**0.015**	0.014	0.016
w_rpy	0.662	**0.035**	0.032	0.038
w_static	0.090	**0.006**	0.006	0.008
s_halfsphere	0.020	**0.017**	0.016	0.020
s_xyz	**0.009**	0.015	0.013	0.015

Table 6.2 Comparison of the RMSE of ATE [m] of DynaSLAM against ORB-SLAM2 (Mur-Artal and Tardós, 2017) for RGB-D cameras. To account for the non-deterministic nature of the system, we show the median, minimum and maximum error of ten runs.

Table 6.2 shows our results on the same sequences, compared against RGB-D ORB-SLAM2. Our method outperforms ORB-SLAM2 in highly dynamic scenarios (*walking*), reaching an error similar to that of the original RGB-D ORB-SLAM2 system in static scenarios. In the case of low-dynamic scenes (*sitting*) the tracking results are slightly worse because the tracked keypoints find themselves further than those belonging to dynamic objects. Albeit, DynaSLAM's map does not contain the dynamic objects that appear along the sequence. Fig. 6.7 shows an example of the estimated trajectories of DynaSLAM and ORB-SLAM2, compared to the ground-truth.

Table 6.3 shows a comparison between our system and several state-of-the-art RGB-D SLAM systems designed for dynamic environments. In account for the effectiveness of our and the state-of-the-art approaches for motion detection (independently of the utilized SLAM system), we also show the respective improvement values against the original SLAM system

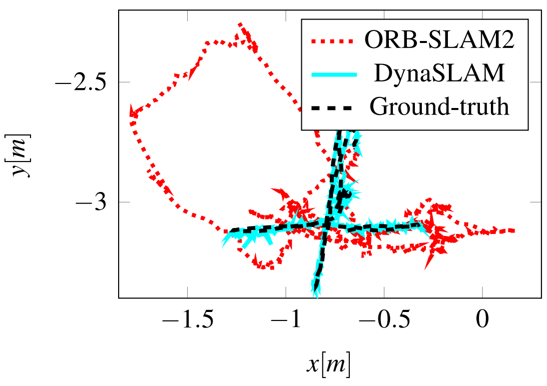


Fig. 6.7 Ground truth and trajectories estimated by DynaSLAM and ORB-SLAM2 in the TUM sequence *f23/walking_xyz*.

Sequence	Depth Edge SLAM	Motion Segmentation DSLAM			Motion Removal DVO-SLAM			DynaSLAM (N+G) (RGB-D)		
		w/o Motion Detection	w/ Motion Detection	Improv. w/ MD	w/o Motion Detection	w/ Motion Detection	Improv. w/ MD	w/o Motion Detection	w/ Motion Detection	Improv. w/ MD
	[m]	[m]	[m]	[%]	[m]	[m]	[%]	[m]	[m]	[%]
w_half	0.049	0.116	0.055	52.59%	0.529	0.125	76.32%	0.351	**0.025**	**92.88%**
w_xyz	0.060	0.202	0.040	80.20%	0.597	0.093	84.38%	0.459	**0.015**	**96.73%**
w_rpy	0.179	0.515	0.076	85.24%	0.730	0.133	81.75%	0.662	**0.035**	**94.71%**
w_stat	0.026	0.470	0.024	**94.89%**	0.212	0.066	69.06%	0.090	**0.006**	93.33%
s_half	0.043	-	-	-	0.062	0.047	**23.70%**	0.020	**0.017**	15.00%
s_xyz	0.040	-	-	-	0.051	0.048	**4.55%**	0.009	**0.015**	X

Depth Edge SLAM by Li and Lee (2017).
Motion Segmentation DSLAM by Wang and Huang (2014)
Motion Removal DVO-SLAM by Sun et al. (2017)
DynaSLAM w/o Motion Detection is ORB-SLAM2 by Mur-Artal and Tardós (2017)

Table 6.3 Absolute trajectory RMSE [m] of DynaSLAM against state-of-the-art RGB-D SLAM systems in dynamic scenes. To evaluate the effectiveness of the specific module addressing dynamic content, we report the improvement with respect to the original SLAM systems (w/o Motion Detection). Our results are estimated using Mask R-CNN and multi-view geometry.

used in every case. DynaSLAM significantly outperforms all of them in all sequences (both high and low dynamic ones). The error is, in general, around 1-2 cm, similar to that of the state of the art in static scenes. Our motion detection approach also outperforms the other methods.

ORB-SLAM, the monocular version of ORB-SLAM2, is generally more accurate than the RGB-D one in dynamic scenes, due to their different initialization algorithms. RGB-D ORB-SLAM2 is initialized and starts the tracking from the very first frame, and hence dynamic objects can introduce errors. ORB-SLAM delays the initialization until there is parallax and consensus using the staticity assumption. Hence, it does not track the camera for the full sequence, sometimes missing a substantial part of it, or even not initializing.

Table 6.4 shows the tracking results and percentage of the tracked trajectory for ORB-SLAM and DynaSLAM (monocular) in the TUM dataset. The initialization in DynaSLAM is always quicker than that of ORB-SLAM. In fact, in highly dynamic sequences, ORB-SLAM initialization only occurs when the moving objects disappear from the scene. In conclusion, although the accuracy of DynaSLAM is slightly lower, it succeeds in bootstrapping the system with dynamic content and producing a map without such content (see Fig. 6.1), to be re-used for long-term applications. The reason why DynaSLAM is slightly less accurate is that the estimated trajectory is longer, and there is therefore room for accumulating errors.

Sequence	ORB-SLAM Mur-Artal and Tardós (2017)		DynaSLAM (Monocular)	
	ATE [m]	% Traj	ATE [m]	% Traj
$fr3/walking\_halfsphere$	**0.017**	87.16	0.021	**97.84**
$fr3/walking\_xyz$	**0.012**	57.63	0.014	**87.37**
$fr2/desk\_with\_person$	**0.006**	95.30	0.008	**97.07**
$fr3/sitting\_xyz$	**0.007**	91.44	0.013	**100.00**

Table 6.4 Absolute trajectory RMSE [m] and percentage of successfully tracked trajectory for both ORB-SLAM and DynaSLAM (monocular).

Sequence	ORB-SLAM2 (Stereo) Mur-Artal and Tardós (2017)			DynaSLAM (Stereo)		
	RPE [%]	RRE [°/100m]	ATE [m]	RPE [%]	RRE [°/100m]	ATE [m]
KITTI 00	**0.70**	**0.25**	**1.3**	0.74	0.26	1.4
KITTI 01	**1.39**	**0.21**	10.4	1.57	0.22	**9.4**
KITTI 02	**0.76**	**0.23**	**5.7**	0.80	0.24	6.7
KITTI 03	0.71	0.18	0.6	**0.69**	0.18	0.6
KITTI 04	0.48	0.13	0.2	**0.45**	**0.09**	0.2
KITTI 05	0.40	0.16	0.8	0.40	0.16	0.8
KITTI 06	0.51	**0.15**	0.8	**0.50**	0.17	0.8
KITTI 07	**0.50**	**0.28**	0.5	0.52	0.29	0.5
KITTI 08	1.05	0.32	3.6	1.05	0.32	**3.5**
KITTI 09	**0.87**	**0.27**	3.2	0.93	0.29	**1.6**
KITTI 10	**0.60**	**0.27**	**1.0**	0.67	0.32	1.2

Table 6.5 Comparison of the RMSE of the ATE [m], the average of the RPE [%] and the RRE [$^\circ/100m$] of DynaSLAM against ORB-SLAM2 system for stereo cameras.

6.4.2 KITTI Dataset

The KITTI Dataset (Geiger et al., 2013) contains stereo sequences recorded from a car in urban and highway environments. Table 6.5 shows our results in the eleven training sequences, compared against stereo ORB-SLAM2. We use two different metrics, the absolute trajectory RMSE proposed in Sturm et al. (2012b), and the average relative translation and rotation errors, proposed in Geiger et al. (2013). Table 6.6 shows the results in the same sequences for the monocular variants of ORB-SLAM and DynaSLAM.

Sequence	ORB-SLAM Mur-Artal and Tardós (2017)	DynaSLAM (Monocular)
KITTI 00	**5.33**	7.55
KITTI 02	**21.28**	26.29
KITTI 03	**1.51**	1.81
KITTI 04	1.62	**0.97**
KITTI 05	4.85	**4.60**
KITTI 06	**12.34**	14.74
KITTI 07	**2.26**	2.36
KITTI 08	46.68	**40.28**
KITTI 09	6.62	**3.32**
KITTI 10	8.80	**6.78**

Table 6.6 Absolute trajectory RMSE [m] for ORB-SLAM and DynaSLAM (monocular).

Note that the results are similar in both the monocular and stereo cases, but the former is more sensitive to dynamic objects and therefore to the additions in DynaSLAM. In some sequences the accuracy of the tracking is improved when not using features belonging to *a priori* dynamic objects, *i.e.*, cars, bicycles, *etc.* An example of this would be the sequences KITTI 01 and KITTI 04, in which all vehicles that appear are moving. In the sequences in which most of the recorded cars and vehicles are parked (hence static), the absolute trajectory RMSE is usually bigger since the keypoints used for tracking are more distant and usually belong to low-texture areas (KITTI 00, KITTI 02, KITTI 06). However, the loop closure and relocalization algorithms work more robustly since the resulting map only contains structural objects, *i.e.*, the map can be re-used and work in long-term applications.

As future work, it is interesting to make a distinction between those *movable* and *moving* objects, by using only RGB information. If a car is detected by the CNN (*movable*) but is not currently moving, its corresponding keypoints should be used for the local tracking, but should not be in the map.

6.4.3 Timing Analysis

To complete the evaluation of our proposal, Table 6.7 shows the average computational time for its different stages. Note that DynaSLAM is not optimized for real-time operation. However, its capability for creating life-long maps of the static scene content are also relevant for running on offline mode.

Sequence	Low-Cost Tracking [ms]	Multi-view Geometry [ms]	Background Inpainting [ms]
w_halfsphere	1.69	333.68	208.09
w_rpy	1.59	235.98	183.56

Table 6.7 DynaSLAM average computational time [ms].

Mur *et al.* show real-time results for and ORB-SLAM2 (Mur-Artal and Tardós, 2017). He et al. (2017) report that Mask R-CNN runs at 195 ms per image on a Nvidia Tesla M40 GPU.

The addition of the multi-view geometry stage is an additional slowdown, due mainly to the region growth algorithm. The background inpainting also introduces a delay, which is another reason why it should be done after the tracking and mapping stage, as it has been shown in Fig. 6.2.

6.5 Conclusions

We have presented a visual SLAM system that, building on ORB-SLAM, adds a motion segmentation approach that makes it robust in dynamic environments for monocular, stereo and RGB-D cameras, offering a solution to a very well known Visual SLAM problem. Our system accurately tracks the camera and creates a static and therefore reusable map of the scene. In the RGB-D case, DynaSLAM is capable of obtaining the synthetic RGB frames with no dynamic content and with the occluded background inpainted, as well as their corresponding synthesized depth frames, which might be together very useful for virtual reality applications. We include a video showing the potential of DynaSLAM [2].

The comparison against the state of the art shows that DynaSLAM achieves in most cases the highest accuracy.

In the videos of the TUM dataset that include Dynamic Objects dataset, at the moment of publishing this work DynaSLAM was the best RGB-D SLAM solution. Currently it is still the most accurate, although works like the one presented in Dai et al. (2020) offer a model that does not require GPU sacrificing on performance, the authors also claim that their proposal could be combined to some of the ideas presented in this chapter. Similar conclusions are found in Vincent et al. (2020). Being the computation time one of the biggest drawback of DynaSLAM according to these recent publication we think is interesting to share the work by Alonso et al. (2020) introducing MiniNet, a real-time semantic segmentation CNN; our proposal in this chapter is not dependent on the CNN we are currently using but it can use a different less time-consuming network as MiniNet. In the monocular case, our

[2] `https://youtu.be/EabI_goFmQs`

accuracy is similar to that of ORB-SLAM, obtaining however a static map of the scene with an earlier initialization.

In the KITTI dataset DynaSLAM is slightly less accurate than monocular and stereo ORB-SLAM, except for those cases in which dynamic objects represent an important part of the scene. However, our estimated map only contains structural objects and can therefore be re-used in long-term applications.

Future works in this line of research have looked, real-time performance Dai et al. (2020), Vincent et al. (2020), Alonso et al. (2020), an RGB-based motion detector, or a more realistic appearance of the synthesized RGB frames by using a more elaborate inpainting technique, *e.g.*, the one used by Pathak et al. (2016) by the use of Generative Adversarial Networks (GANs). This last idea was carried out, posterior to this work, by Bescos et al. (2019) where they use GANs to inpaint already detected dynamic objects. At the moment it has only been tested in simulation with ground-truth segmentation.

Chapter 7

Conclusions

In general, 3D visual perception is far from being fully solved. There are significant research challenges ahead, in particular related to scene understanding. In this thesis we have advanced the state of the art in several areas of this exciting and relevant topic.

The first contribution described in this thesis is on single-view depth estimation. In CAM-Convs (Facil et al., 2019) we have proposed a new type of convolution and demonstrated the advantages of accounting for the camera intrinsic parameters in depth estimation tasks. We have shown that our CAM-Convs allow us to train and test with different cameras; something that has been explored further in López-Antequera et al. (2020). Future work along this line should explore models that leverage the camera intrinsics and, unlike CAM-Convs, do not require to learn how to use them. A major drawback of CAM-Convs is that, as any learning procedure, they are strongly data-dependent. Therefore, a sufficiently well sampled dataset of images taken by different cameras is needed for a reasonable performance, and 1 or 2 cameras might not be enough. On this line, López-Antequera et al. (2020) has started making progress on their proposal of a canonical camera model, similar to the focal length normalization we use in our work.

We have demonstrated in Chapter 3 that traditional multi-view geometry and deep learning can benefit from each other, achieving toghether an accuracy that outperforms both of them separately. It is worth remarking that our work was one of the first addressing this idea. After us, many novel approaches have been proposed. In particular, I would highlight two of them that couple deep learning and multi-view geometry quite tightly, instead of them being two different procedures subsequently merged. On the one side, CodeSLAM (Bloesch et al., 2018) and its following work DeepFactors (Czarnowski et al., 2020) successfully propose a deep neural network that defines a manifold for each depth map, in which traditional multi-view geometry optimization finds the best depth maps according to geometric and photo-metric errors. On the other hand, Zhou et al. (2018) presented DeepTAM, continuing their work

DeMoN in Ummenhofer et al. (2017), proposing an deep neural network that iteratively refines its predictions using multi-view geometry, achieving an impressive accuracy.

On visual place recognition, we have presented three novel approaches for multi-view global descriptors that are robust to changes in the appearance created by different conditions. We have tested it for multiple seasons and for different light conditions. It is also worth remarking that our work is the first one proposing multi-view descriptors based on deep learning, and that our descriptors were further explored in the dataset compiled in Warburg et al. (2020) with similar conclusions as in our work. For future research, it would be interesting to explore condition-invariant local descriptors that would allow to recover a metric pose and not only a topological one (Revaud et al., 2019).

In Chapter 5 we presented CFL and EquiConvs. CFL is a network that achieves state-of-the-art results in indoor layout recovery, and EquiConvs are a special type of convolutions that adapt is shape to the equirectangular distortion. Both contributions can have several applications in Visual SLAM (Salas et al., 2015). EquiConvs is also a general model, from which any deep network using panoramic images can benefit from.

Regarding SLAM in dynamic environments, we have proposed a pipeline to avoid dynamic or movable objects to perturb mapping and tracking algorithms assuming a rigid world. Our main contribution is the design of the DynaSLAM pipeline and the inclusion of a segmentation CNN embedded in a visual SLAM system. We demonstrate that our results are very competitive, and that our proposal outperforms state-of-the-art SLAM systems. A reasonable line for future work would be to focus on tracking the dynamic objects and possibly use it also into its advantage. Preliminary results on this direction can already be seen in Ballester et al. (2020). A potential advantage could be, for example: if an object is being tracked and at some point the camera is occluded by it, the object motion estimation would allow a reasonable estimation of the camera motion for some time.

We can draw a general conclusion for this thesis by writing that, on the one hand, we have proposed several novel methods to use deep networks for 3D perception challenges. And, on the other hand, we have also made contributions within deep learning for this particular domain. On the first set of proposals, we have developed novel methods to fuse multi-view and single-view depths and to detect and remove dynamic objects in visual SLAM. On the second set of proposals, we have developed novel multi-view embeddings for place recognition and two novel convolution types, CAM-Convs and Equiconvs, explicitly including the camera intrinsics and demonstrating better performances for single-view depth learning with multiple cameras and layout estimation from equirectangular images.

7.1 Limitations and Future Work

Deep learning has supposed a great advance in 3D visual perception and it is making its way into visual SLAM. The biggest limitation we found while working on this thesis is the dependency on data, and more specifically on good-quality and diverse, sufficiently well sampled data. To exemplify this, look at the dense depth estimation problem. The progress achieved by using deep learning has no precedents. However, it is very easy to fall into small segments of the problem by evaluating the models in a subset of the real cases, e.g. a relatively small dataset on a very specific and biased domain. A common case is the training on different domains separately or, as we pointed in Chapter 2, commonly used datasets only provide images taken by one type of camera. This is different to traditional 3D vision algorithms that explicitly consider the camera model and do not make any assumption (or the smallest possible number of them) in the type of data or domain a priory. In this thesis we always kept this in mind. CAM-Convs introduce the camera model into convolutions for the first time. We also made use of unbiased depth estimation from a traditional geometry-based triangulation to complement the learned depth prediction. We have adapted standard convolutions to equirectangular distortion in EquiConvs, again taking into account the camera model. Lastly, we have combined deep learning and traditional methods for dynamic object detection in DynaSLAM. We agree with the general thought that deep learning has a great potential for 3D perception. However, future research needs to address the data dependency, creating more complete and general benchmarks (as López-Antequera et al. (2020)) and also models that account for the 3D-to-2D projection and the data noise. We can cite (Czarnowski et al., 2020, Zhou et al., 2020) as examples of the former, and Bayesian deep learning (for example, Gustafsson et al. (2020)) as a promising line of work for the latter. Learning from data might be the key for a complete scene understanding, but, in our believe, only those proposals that complement machine learning with uncertainty, geometric and physical models will achieve the best performance.

References

Abadi, Martín, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. (2016), "Tensorflow: a system for large-scale machine learning." In *OSDI*, volume 16, 265–283.

Alcantarilla, Pablo F, José J Yebes, Javier Almazán, and Luis M Bergasa (2012), "On combining visual SLAM and dense scene flow to increase the robustness of localization and mapping in dynamic environments." In *ICRA*.

Alonso, Inigo, Luis Riazuelo, and Ana C Murillo (2020), "Mininet: An efficient semantic segmentation convnet for real-time robotic applications." *IEEE Transactions on Robotics*.

Ambrus, Rares, John Folkesson, and Patric Jensfelt (2016), "Unsupervised object segmentation through change detection in a long term autonomy scenario." In *Humanoid Robots (Humanoids)*, IEEE.

Amodei, Dario, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. (2016), "Deep speech 2: End-to-end speech recognition in english and mandarin." In *International conference on machine learning*, 173–182.

Arandjelovic, Relja, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic (2016), "NetVLAD: CNN architecture for weakly supervised place recognition." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5297–5307.

Arandjelović, Relja and Andrew Zisserman (2014), "Dislocation: Scalable descriptor distinctiveness for location recognition." In *Asian Conference on Computer Vision*, 188–204, Springer.

Armeni, I., A. Sax, A. R. Zamir, and S. Savarese (2017), "Joint 2D-3D-Semantic Data for Indoor Scene Understanding." *ArXiv*.

Armeni, Iro, Sasha Sax, Amir R Zamir, and Silvio Savarese (2017), "Joint 2D-3D-semantic data for indoor scene understanding." *arXiv preprint arXiv:1702.01105*.

Arroyo, Roberto, Pablo F Alcantarilla, Luis M Bergasa, and Eduardo Romera (2016), "Fusion and binarization of CNN features for robust topological localization across seasons." In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, 4656–4663, IEEE.

Ballester, Irene, Alejandro Fontan, Javier Civera, Klaus H Strobl, and Rudolph Triebel (2020), "Dot: Dynamic object tracking for visual slam." *arXiv preprint arXiv:2010.00052*.

Bampis, Loukas, Angelos Amanatiadis, and Antonios Gasteratos (2016), “Encoding the description of image sequences: A two-layered pipeline for loop closure detection.” In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4530–4536, IEEE.

Bao, Sid Yingze, Min Sun, and Silvio Savarese (2011), “Toward coherent object detection and scene layout understanding.” *Image and Vision Computing*, 29, 569–579.

Bay, Herbert, Tinne Tuytelaars, and Luc Van Gool (2006), “Surf: Speeded up robust features.” In *European conference on computer vision*, 404–417, Springer.

Bescos, Berta, José Neira, Roland Siegwart, and Cesar Cadena (2019), “Empty cities: Image inpainting for a dynamic-object-invariant space.” In *2019 International Conference on Robotics and Automation (ICRA)*, 5460–5466, IEEE.

Bloesch, Michael, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J Davison (2018), “CodeSLAM-Learning a Compact, Optimisable Representation for Dense Visual SLAM.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Cadena, Cesar, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard (2016), “Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age.” *IEEE Transactions on robotics*, 32, 1309–1332.

Cao, Yuanzhouhan, Zifeng Wu, and Chunhua Shen (2016), “Estimating depth from monocular images as classification using deep fully convolutional residual networks.” *arXiv preprint arXiv:1605.02305*.

Chakrabarti, Ayan, Jingyu Shao, and Gregory Shakhnarovich (2016), “Depth from a Single Image by Harmonizing Overcomplete Local Network Predictions.” In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS’16, 2666–2674, Curran Associates Inc., USA, URL http://dl.acm.org/citation.cfm?id=3157382.3157396.

Chen, Weifeng, Zhao Fu, Dawei Yang, and Jia Deng (2016), “Single-Image Depth Perception in the Wild.” In *Advances in Neural Information Processing Systems 29* (D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, eds.), 730–738, Curran Associates, Inc., URL http://papers.nips.cc/paper/6489-single-image-depth-perception-in-the-wild.pdf.

Chen, Zetao, Adam Jacobson, Niko Sunderhauf, Ben Upcroft, Lingqiao Liu, Chunhua Shen, Ian Reid, and Michael Milford (2017), “Deep Learning Features at Scale for Visual Place Recognition.” *arXiv preprint arXiv:1701.05105*.

Chen, Zetao, Obadiah Lam, Adam Jacobson, and Michael Milford (2014), “Convolutional neural network-based place recognition.” *arXiv preprint arXiv:1411.1509*.

Chen, Zetao, Lingqiao Liu, Inkyu Sa, Zongyuan Ge, and Margarita Chli (2018), “Learning context flexible attention model for long-term visual place recognition.” *IEEE Robotics and Automation Letters*, 3, 4015–4022.

Cohen, Taco S, Mario Geiger, Jonas Köhler, and Max Welling (2018), "Spherical cnns." *arXiv:1801.10130*.

Concha, Alejo and Javier Civera (2014), "Using superpixels in monocular SLAM." In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, 365–372, IEEE.

Concha, Alejo and Javier Civera (2015a), "DPPTAM: Dense piecewise planar tracking and mapping from a monocular sequence." In *IEEE/RSJ international conference on intelligent robots and systems*.

Concha, Alejo and Javier Civera (2015b), "An evaluation of robust cost functions for rgb direct mapping." In *ECMR*, IEEE.

Concha, Alejo, Muhammad Wajahat Hussain, Luis Montano, and Javier Civera (2014), "Manhattan and Piecewise-Planar Constraints for Dense Monocular Mapping." In *Robotics: Science and systems*.

Concha, Alejo, Wajahat Hussain, Luis Montano, and Javier Civera (2015), "Incorporating scene priors to dense monocular mapping." *Autonomous Robots*, 39, 279–292.

Czarnowski, J, T Laidlow, R Clark, and AJ Davison (2020), "Deepfactors: Real-time probabilistic dense monocular slam." *IEEE Robotics and Automation Letters*, 5, 721–728, URL http://dx.doi.org/10.1109/lra.2020.2965415.

Dai, Angela, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner (2017a), "ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes." In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*.

Dai, Jifeng, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei (2017b), "Deformable convolutional networks." *CoRR, abs/1703.06211*, 1, 3.

Dai, Weichen, Yu Zhang, Ping Li, Zheng Fang, and Sebastian Scherer (2020), "Rgb-d slam in dynamic environments using point correlations." *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Dasgupta, Saumitro, Kuan Fang, Kevin Chen, and Silvio Savarese (2016), "Delay: Robust spatial layout estimation for cluttered indoor scenes." In *Conference on Computer Vision and Pattern Recognition*, 616–624.

Delage, Erick, Honglak Lee, and Andrew Y Ng (2006), "A dynamic bayesian network model for autonomous 3D reconstruction from a single indoor image." In *Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, 2418–2428.

Donahue, Jeff, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell (2014), "Long-term recurrent convolutional networks for visual recognition and description." *arXiv preprint arXiv:1411.4389*.

Dosovitskiy, Alexey, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox (2015), "Flownet: Learning optical flow with convolutional networks." In *Proceedings of the International Conference on Computer Vision*, 2758–2766.

Dusmanu, Mihai, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler (2019), "D2-net: A trainable cnn for joint detection and description of local features." *arXiv preprint arXiv:1905.03561*.

Eigen, David and Rob Fergus (2015), "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture." In *Proceedings of the IEEE International Conference on Computer Vision*, 2650–2658.

Eigen, David, Christian Puhrsch, and Rob Fergus (2014), "Depth map prediction from a single image using a multi-scale deep network." In *Advances in neural information processing systems*, 2366–2374.

Engel, Jakob, Vladlen Koltun, and Daniel Cremers (2017), "Direct sparse odometry." *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Engel, Jakob, Thomas Schöps, and Daniel Cremers (2014), "LSD-SLAM: Large-scale direct monocular SLAM." In *European Conference on Computer Vision*, 834–849, Springer.

Ens, John and Peter Lawrence (1993), "An investigation of methods for determining depth from focus." *IEEE Transactions on pattern analysis and machine intelligence*, 15, 97–108.

Fácil, José M, Alejo Concha, Luis Montesano, and Javier Civera (2017), "Single-View and Multi-View Depth Fusion." *IEEE Robotics and Automation Letters*, 2, 1994–2001.

Facil, Jose M, Benjamin Ummenhofer, Huizhong Zhou, Luis Montesano, Thomas Brox, and Javier Civera (2019), "CAM-Convs: Camera-Aware Multi-Scale Convolutions for Single-View Depth." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 11826–11835.

Fan, Haoqiang, Hao Su, and Leonidas J Guibas (2017), "A point set generation network for 3d object reconstruction from a single image." In *CVPR*, 2463–2471.

Fernandez-Labrador, Clara, Jose M Facil, Alejandro Perez-Yus, Cedric Demonceaux, and Jose J Guerrero (2018a), "Panoroom: From the sphere to the 3d layout." *arXiv:1808.09879*.

Fernandez-Labrador, Clara, Alejandro Perez-Yus, Gonzalo Lopez-Nicolas, and Jose J Guerrero (2018b), "Layouts from panoramic images with geometry and deep learning." *Robotics and Automation Letters*, 3, 3153–3160.

Flint, Alex, David Murray, and Ian Reid (2011), "Manhattan scene understanding using monocular, stereo, and 3d features." In *Computer Vision (ICCV), 2011 International Conference on*, 2228–2235, IEEE.

Fouhey, David F, Vincent Delaitre, Abhinav Gupta, Alexei A Efros, Ivan Laptev, and Josef Sivic (2014), "People watching: Human actions as a cue for single view geometry." *International journal of computer vision*, 110, 259–274.

Fouhey, David F, Abhinav Gupta, and Martial Hebert (2013), "Data-driven 3d primitives for single image understanding." In *Proceedings of the IEEE International Conference on Computer Vision*, 3392–3399.

Fu, Huan, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao (2018), "Deep ordinal regression network for monocular depth estimation." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2002–2011.

Furlan *et al* (2013), "Free your camera: 3d indoor scene understanding from arbitrary camera motion." *BMVC*.

Gálvez-López, Dorian and Juan D Tardos (2012), "Bags of binary words for fast place recognition in image sequences." *IEEE Transactions on Robotics*, 28, 1188–1197.

Garcia-Fidalgo, Emilio and Alberto Ortiz (2015), "Vision-based topological mapping and localization methods: A survey." *Robotics and Autonomous Systems*, 64, 1–20.

Garcia-Garcia, Alberto, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, Pablo Martinez-Gonzalez, and Jose Garcia-Rodriguez (2018), "A survey on deep learning techniques for image and video semantic segmentation." *Applied Soft Computing*, 70, 41–65.

Garg, Sourav, Niko Sünderhauf, and Michael Milford (2019), "Semantic-Geometric Visual Place Recognition: A New Perspective for Reconciling Opposing Views." *International Journal of Robotics Research*.

Geiger, Andreas, Philip Lenz, Christoph Stiller, and Raquel Urtasun (2013), "Vision meets robotics: The KITTI dataset." *IJRR*, 32, 1231–1237.

Geiger, Andreas, Philip Lenz, and Raquel Urtasun (2012), "Are we ready for autonomous driving? the kitti vision benchmark suite." In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 3354–3361, IEEE.

Gerlach, Nicolaas Lucius, Gerrit Jacobus Meijer, Dirk-Jan Kroon, Ewald Maria Bronkhorst, Stefaan Jozef Bergé, and Thomas Jan Jaap Maal (2014), "Evaluation of the potential of automatic segmentation of the mandibular canal." *BJOMS*.

Godard, Clément, Oisin Mac Aodha, and Gabriel Brostow (2018), "Digging into self-supervised monocular depth estimation." *arXiv preprint arXiv:1806.01260*.

Godard, Clément, Oisin Mac Aodha, and Gabriel J Brostow (2017), "Unsupervised monocular depth estimation with left-right consistency." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 270–279.

Gomez, Raul, Jaume Gibert, Lluis Gomez, and Dimosthenis Karatzas (2020), "Exploring hate speech detection in multimodal publications." In *The IEEE Winter Conference on Applications of Computer Vision*, 1470–1478.

Gomez-Ojeda, Ruben, Jesus Briales, and Javier Gonzalez-Jimenez (2016), "Pl-svo: Semi-direct monocular visual odometry by combining points and line segments." In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4211–4216, IEEE.

Gomez-Ojeda, Ruben, Manuel Lopez-Antequera, Nicolai Petkov, and Javier Gonzalez-Jimenez (2015), "Training a convolutional neural network for appearance-invariant place recognition." *arXiv preprint arXiv:1505.07428*.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016), *Deep learning*. MIT press.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014), "Generative adversarial nets." In *Advances in neural information processing systems*, 2672–2680.

Gordo, Albert, Jon Almazán, Jerome Revaud, and Diane Larlus (2016), "Deep image retrieval: Learning global representations for image search." In *European conference on computer vision*, 241–257, Springer.

Graber, Gottfried, Thomas Pock, and Horst Bischof (2011), "Online 3D reconstruction using convex optimization." In *2011 IEEE International Conference on Computer Vision Workshops*, 708–711, IEEE.

Gustafsson, Fredrik K, Martin Danelljan, and Thomas B Schon (2020), "Evaluating scalable bayesian deep learning methods for robust computer vision." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 318–319.

Gutiérrez-Gómez, Daniel, Walterio Mayol-Cuevas, and JJ Guerrero (2015), "Inverse Depth for Accurate Photometric and Geometric Error Minimisation in RGB-D Dense Visual Odometry." In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, 83–89, IEEE.

Handa, Ankur, Richard A Newcombe, Adrien Angeli, and Andrew J Davison (2011), "Applications of legendre-fenchel transformation to computer vision problems." *Department of Computing at Imperial College London. DTR11-7*, 45.

Harris, Christopher G, Mike Stephens, et al. (1988), "A combined corner and edge detector." In *Alvey vision conference*, volume 15, 10–5244, Citeseer.

He, Kaiming, Georgia Gkioxari, Piotr Dollár, and Ross Girshick (2017), "Mask R-CNN." *arXiv preprint arXiv:1703.06870*.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016), "Deep residual learning for image recognition." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

He, Lei, Guanghui Wang, and Zhanyi Hu (2018), "Learning depth from single images with deep neural network embedding focal length." *IEEE Transactions on Image Processing*, 27, 4676–4689.

Hedau, V., D. Hoiem, and D. Forsyth (2009a), "Recovering the spatial layout of cluttered rooms." In *International Conference on Computer Vision*, 1849–1856.

Hedau, Varsha, Derek Hoiem, and David Forsyth (2009b), "Recovering the spatial layout of cluttered rooms." In *2009 IEEE 12th international conference on computer vision*, 1849–1856, IEEE.

Hochreiter, Sepp and Jürgen Schmidhuber (1997), "Long short-term memory." *Neural computation*, 9, 1735–1780.

Hoiem, Derek, Alexei A Efros, and Martial Hebert (2005), “Automatic photo pop-up.” *ACM transactions on graphics (TOG)*, 24, 577–584.

Huang, Po-Han, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang (2018), “Deepmvs: Learning multi-view stereopsis.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2821–2830.

Hussain, Wajahat, Javier Civera, Luis Montano, and Martial Hebert (2016), “Dealing with small data and training blind spots in the Manhattan world.” In *Winter Conference on Applications of Computer Vision (WACV)*, 1–9, IEEE.

Joo *et al* (2018), “Globally optimal inlier set maximization for atlanta frame estimation.” *CVPR*.

Karsch, Kevin, Varsha Hedau, David Forsyth, and Derek Hoiem (2011), “Rendering synthetic objects into legacy photographs.” *ACM Transactions on Graphics (TOG)*, 30, 157.

Kehl, Wadim, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab (2017), “SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again.” In *Proceedings of the International Conference on Computer Vision (ICCV 2017), Venice, Italy*, 22–29.

Kendall, Alex, Roberto Cipolla, et al. (2017), “Geometric loss functions for camera pose regression with deep learning.” In *Proc. CVPR*, volume 3, 8.

Kendall, Alex, Matthew Grimes, and Roberto Cipolla (2015), “PoseNet: A convolutional network for real-time 6-DOF camera relocalization.” In *Proceedings of the IEEE international conference on computer vision*, 2938–2946.

Kim, Deok-Hwa and Jong-Hwan Kim (2016), “Effective Background Model-Based RGB-D Dense Visual Odometry in a Dynamic Environment.” *IEEE T-RO*.

Kingma, Diederik P and Jimmy Ba (2014), “Adam: A method for stochastic optimization.” *arXiv preprint arXiv:1412.6980*.

Klein, Georg and David Murray (2007), “Parallel tracking and mapping for small AR workspaces.” In *ISMAR*, 225–234.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012), “Imagenet classification with deep convolutional neural networks.” In *Advances in neural information processing systems*, 1097–1105.

Kuznietsov, Yevhen, Jörg Stückler, and Bastian Leibe (2017), “Semi-supervised deep learning for monocular depth map prediction.” In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 6647–6655.

Laina, Iro, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab (2016), “Deeper depth prediction with fully convolutional residual networks.” In *3D Vision (3DV), 2016 Fourth International Conference on*, 239–248, IEEE.

Lamarca, Jose, Shaifali Parashar, Adrien Bartoli, and JMM Montiel (2020), “Defslam: Tracking and mapping of deforming scenes from monocular sequences.” *IEEE Transactions on robotics*, ?, ?–?

LeCun, Yann, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel (1989), “Backpropagation applied to handwritten zip code recognition.” *Neural computation*, 1, 541–551.

Lee, C., V. Badrinarayanan, T. Malisiewicz, and A. Rabinovich (2017), “RoomNet: End-to-end room layout estimation.” In *International Conference on Computer Vision*.

Lee, David C, Martial Hebert, and Takeo Kanade (2009), “Geometric reasoning for single image structure recovery.” In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2136–2143.

Lee, Seong Hun and Javier Civera (2019), “Loosely-Coupled Semi-Direct Monocular SLAM.” *IEEE Robotics and Automation Letters*, 4, 399–406.

Li, Jun, Reinhard Klein, and Angela Yao (2016), “Learning fine-scaled depth maps from single rgb images.” *arXiv preprint arXiv:1607.00730*.

Li, Ruibo, Ke Xian, Chunhua Shen, Zhiguo Cao, Hao Lu, and Lingxiao Hang (2018), “Deep attention-based classification network for robust depth prediction.” *arXiv preprint arXiv:1807.03959*.

Li, Shile and Dongheui Lee (2017), “RGB-D SLAM in Dynamic Environments Using Static Point Weighting.” *IEEE RA-L*, 2, 2263–2270.

Li, Zhengqi and Noah Snavely (2018), “Megadepth: Learning single-view depth prediction from internet photos.” In *Computer Vision and Pattern Recognition (CVPR)*.

Lin, Tsung-Yi, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick (2014), “Microsoft coco: Common objects in context.” In *ECCV*.

Liu, Chenxi, Alexander G. Schwing, Kaustav Kundu, Raquel Urtasun, and Sanja Fidler (2015a), “Rent3d: Floor-plan priors for monocular layout estimation.” In *The Conference on Computer Vision and Pattern Recognition (CVPR)*.

Liu, Fayao, Chunhua Shen, and Guosheng Lin (2015b), “Deep convolutional neural fields for depth estimation from a single image.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5162–5170.

Liu, Li, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen (2020), “Deep learning for generic object detection: A survey.” *International journal of computer vision*, 128, 261–318.

Liu, Rosanne, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski (2018), “An intriguing failing of convolutional neural networks and the coordconv solution.” *arXiv preprint arXiv:1807.03247*.

López-Antequera, Manuel, Pau Gargallo, Markus Hofinger, and Samuel Rota (2020), “Mapillary planet-scale depth dataset.” In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Lopez-Antequera, Manuel, Ruben Gomez-Ojeda, Nicolai Petkov, and Javier Gonzalez-Jimenez (2017), “Appearance-invariant place recognition by discriminatively training a convolutional neural network.” *Pattern Recognition Letters*, 92, 89–95.

Lowe, David G (2004), “Distinctive image features from scale-invariant keypoints.” *International journal of computer vision*, 60, 91–110.

Lowry, Stephanie and Henrik Andreasson (2018), “Lightweight, viewpoint-invariant visual place recognition in changing environments.” *IEEE Robotics and Automation Letters*, 3, 957–964.

Lowry, Stephanie and Michael J Milford (2016), “Supervised and unsupervised linear learning techniques for visual place recognition in changing environments.” *IEEE Transactions on Robotics*, 32, 600–613.

Lowry, Stephanie, Niko Sünderhauf, Paul Newman, John J Leonard, David Cox, Peter Corke, and Michael J Milford (2016), “Visual place recognition: A survey.” *IEEE Transactions on Robotics*, 32, 1–19.

Luo, Zixin, Tianwei Shen, Lei Zhou, Jiahui Zhang, Yao Yao, Shiwei Li, Tian Fang, and Long Quan (2019), “Contextdesc: Local descriptor augmentation with cross-modality context.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2527–2536.

Luo, Zixin, Tianwei Shen, Lei Zhou, Siyu Zhu, Runze Zhang, Yao Yao, Tian Fang, and Long Quan (2018), “Geodesc: Learning local descriptors by integrating geometry constraints.” In *Proceedings of the European Conference on Computer Vision (ECCV)*, 168–183.

Ma, Fangchang, Guilherme Venturelli Cavalheiro, and Sertac Karaman (2019), “Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera.” In *2019 International Conference on Robotics and Automation (ICRA)*, 3288–3295, IEEE.

Mallya, A. and S. Lazebnik (2015), “Learning informative edge maps for indoor scene layout prediction.” In *International Conference on Computer Vision*, 936–944.

Marimont, RB and MB Shapiro (1979), “Nearest neighbour searches and the curse of dimensionality.” *IMA Journal of Applied Mathematics*, 24, 59–70.

Mayer, Nikolaus, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox (2016), “A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4040–4048.

McManus, Colin, Winston Churchill, Will Maddern, Alexander D Stewart, and Paul Newman (2014), “Shady dealings: Robust, long-term visual localisation using illumination invariance.” In *2014 IEEE international conference on robotics and automation (ICRA)*, 901–906, IEEE.

McManus, Colin, Winston Churchill, Ashley Napier, Ben Davis, and Paul Newman (2013), “Distraction suppression for vision-based pose estimation at city scales.” In *2013 IEEE International Conference on Robotics and Automation*, 3762–3769, IEEE.

Middelberg, Sven, Torsten Sattler, Ole Untzelmann, and Leif Kobbelt (2014), "Scalable 6-dof localization on mobile devices." In *European conference on computer vision*, 268–283, Springer.

Mikolajczyk, Krystian, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, Jiri Matas, Frederik Schaffalitzky, Timor Kadir, and Luc Van Gool (2005), "A comparison of affine region detectors." *International journal of computer vision*, 65, 43–72.

Milford, Michael J and Gordon F Wyeth (2012), "SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights." In *International Conference on Robotics and Automation*, 1643–1649.

Mur-Artal, Raul, JMM Montiel, and Juan D Tardos (2015), "ORB-SLAM: a versatile and accurate monocular SLAM system." *Robotics, IEEE Transactions on*, 31, 1147–1163.

Mur-Artal, Raul and Juan D Tardós (2017), "ORB-SLAM2: An open-source slam system for monocular, stereo, and RGB-D cameras." *IEEE T-RO*.

Murillo, Ana C, Gautam Singh, Jana Kosecká, and José Jesús Guerrero (2013), "Localization in urban environments using a panoramic gist descriptor." *IEEE Transactions on Robotics*, 29, 146–160.

Naseer, Tayyab, Wolfram Burgard, and Cyrill Stachniss (2018), "Robust visual localization across seasons." *IEEE Transactions on Robotics*, 34, 289–302.

Nathan Silberman, Pushmeet Kohli, Derek Hoiem and Rob Fergus (2012), "Indoor segmentation and support inference from RGBD Images." In *ECCV*.

Neubert, Peer, Stefan Schubert, and Peter Protzel (2015), "Exploiting intra database similarities for selection of place recognition candidates in changing environments." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Workshop on Visual Place Recognition in Changing Environments*.

Newcombe, Richard A, Steven J Lovegrove, and Andrew J Davison (2011), "DTAM: Dense tracking and mapping in real-time." In *Computer Vision (ICCV), 2011 IEEE International Conference on*, 2320–2327, IEEE.

Newman, Paul, David Cole, and Kin Ho (2006), "Outdoor slam using visual appearance and laser ranging." In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006*, 1180–1187, IEEE.

Noh, Hyeonwoo, Andre Araujo, Jack Sim, Tobias Weyand, and Bohyung Han (2017), "Large-scale image retrieval with attentive deep local features." In *Proceedings of the IEEE International Conference on Computer Vision*, 3456–3465.

Olid, Daniel, José M Fácil, and Javier Civera (2018), "Single-View Place Recognition under Seasonal Changes." *arXiv preprint arXiv:1808.06516*.

Ono, Yuki, Eduard Trulls, Pascal Fua, and Kwang Moo Yi (2018), "Lf-net: learning local features from images." In *Advances in neural information processing systems*, 6234–6244.

Pais, G Dias, Tiago J Dias, Jacinto C Nascimento, and Pedro Miraldo (2019), "Omnidrl: Robust pedestrian detection using deep reinforcement learning on omnidirectional cameras." *arXiv preprint arXiv:1903.00676*.

Parkhi, Omkar M, Andrea Vedaldi, and Andrew Zisserman (2015), "Deep face recognition."

Pathak, Deepak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros (2016), "Context encoders: Feature learning by inpainting." In *CVPR*.

Pepperell, Edward, Peter I Corke, and Michael J Milford (2014), "All-environment visual place recognition with smart." In *2014 IEEE international conference on robotics and automation (ICRA)*, 1612–1618, IEEE.

Perez-Yus, Alejandro, Daniel Gutiérrez-Gómez, Gonzalo Lopez-Nicolas, and JJ Guerrero (2017), "Stairs detection with odometry-aided traversal from a wearable rgb-d camera." *Computer Vision and Image Understanding*, 154, 192–205.

Pérez-Yus, Alejandro, Gonzalo López-Nicolás, and Jose J Guerrero (2014), "Detection and modelling of staircases using a wearable depth sensor." In *European Conference on Computer Vision*, 449–463, Springer.

Pinies, Pedro, Lina Maria Paz, and Paul Newman (2015), "Dense mono reconstruction: Living with the pain of the plain plane." In *2015 IEEE International Conference on Robotics and Automation*, 5226–5231.

Piniés, Pedro, Lina María Paz, and Paul Newman (2015), "Too much TV is bad: Dense reconstruction from sparse laser with non-convex regularisation." In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 135–142, IEEE.

Pronobis, Andrzej, Barbara Caputo, Patric Jensfelt, and Henrik I Christensen (2006), "A discriminative approach to robust visual place recognition." In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3829–3836.

Qiu, Jiaxiong, Zhaopeng Cui, Yinda Zhang, Xingdi Zhang, Shuaicheng Liu, Bing Zeng, and Marc Pollefeys (2019), "Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3313–3322.

Ren, Yuzhuo, Shangwen Li, Chen Chen, and C-C Jay Kuo (2016), "A coarse-to-fine indoor layout estimation (cfile) method." In *ACCV*, 36–51.

Revaud, Jerome, Cesar De Souza, Martin Humenberger, and Philippe Weinzaepfel (2019), "R2d2: Reliable and repeatable detector and descriptor." In *Advances in Neural Information Processing Systems*, 12405–12415.

Riazuelo, Luis, Luis Montano, and JMM Montiel (2017), "Semantic visual slam in populated environments." In *2017 European conference on mobile robots (ECMR)*, 1–7, IEEE.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015), "U-net: Convolutional networks for biomedical image segmentation." In *International Conference on Medical image computing and computer-assisted intervention*, 234–241, Springer.

Rublee, Ethan, Vincent Rabaud, Kurt Konolige, and Gary Bradski (2011), "Orb: An efficient alternative to sift or surf." In *Computer Vision (ICCV), 2011 IEEE international conference on*, 2564–2571, IEEE.

Russakovsky, Olga, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. (2015), "Imagenet large scale visual recognition challenge." *International Journal of Computer Vision*, 115, 211–252.

Salas, Marta, Wajahat Hussain, Alejo Concha, Luis Montano, Javier Civera, and JMM Montiel (2015), "Layout aware visual tracking and mapping." In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 149–156, IEEE.

Savinov, Nikolay, Akihito Seki, Lubor Ladicky, Torsten Sattler, and Marc Pollefeys (2017), "Quad-networks: unsupervised learning to rank for interest point detection." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1822–1830.

Saxena, Ashutosh, Jamie Schulte, and Andrew Y Ng (2007), "Depth estimation using monocular and stereo cues." In *IJCAI*, volume 7.

Saxena, Ashutosh, Min Sun, and Andrew Y Ng (2009), "Make3D: Learning 3D scene structure from a single still image." *IEEE transactions on pattern analysis and machine intelligence*, 31, 824–840.

Schonberger, Johannes L and Jan-Michael Frahm (2016), "Structure-from-motion revisited." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4104–4113.

Schwing, Alexander G, Sanja Fidler, Marc Pollefeys, and Raquel Urtasun (2013), "Box in the box: Joint 3D layout and object reasoning from single images." In *International Conference on Computer Vision*, 353–360.

Shen, Shaojie, Nathan Michael, and Vijay Kumar (2011), "Autonomous multi-floor indoor navigation with a computationally constrained mav." In *2011 IEEE International Conference on Robotics and Automation*, 20–25, IEEE.

Shrivastava, Ashish, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb (2017), "Learning from Simulated and Unsupervised Images through Adversarial Training." In *CVPR*, 2242–2251.

Silberman, Nathan, Derek Hoiem, Pushmeet Kohli, and Rob Fergus (2012), "Indoor segmentation and support inference from rgbd images." In *European Conference on Computer Vision*, 746–760, Springer.

Simonyan, Karen and Andrew Zisserman (2014), "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556*.

Song, Shuran and Jianxiong Xiao (2016), "Deep sliding shapes for amodal 3d object detection in rgb-d images." In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 808–816.

Stühmer, Jan, Stefan Gumhold, and Daniel Cremers (2010), “Real-time dense geometry from a handheld camera.” In *Joint Pattern Recognition Symposium*, 11–20, Springer.

Sturm, J., N. Engelhard, F. Endres, W. Burgard, and D. Cremers (2012a), “A benchmark for the evaluation of RGB-D SLAM systems.” In *International Conference on Intelligent Robot Systems (IROS)*.

Sturm, Jürgen, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers (2012b), “A benchmark for the evaluation of RGB-D SLAM systems.” In *IROS*.

Sturm, Peter and Steve Maybank (1999), “A method for interactive 3d reconstruction of piecewise planar objects from single images.” In *The 10th British machine vision conference (BMVC’99)*, 265–274.

Sun, Yuxiang, Ming Liu, and Max Q-H Meng (2017), “Improving RGB-D SLAM in dynamic environments: A motion removal approach.” *RAS*.

Sünderhauf, Niko, Sareh Shirazi, Feras Dayoub, Ben Upcroft, and Michael Milford (2015a), “On the performance of convnet features for place recognition.” In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, 4297–4304, IEEE.

Sünderhauf, Niko, Sareh Shirazi, Adam Jacobson, Feras Dayoub, Edward Pepperell, Ben Upcroft, and Michael Milford (2015b), “Place recognition with convnet landmarks: Viewpoint-robust, condition-robust, training-free.” *Proceedings of Robotics: Science and Systems XII*.

Sundermeyer, Martin, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel (2018), “Implicit 3D Orientation Learning for 6D Object Detection from RGB Images.” In *Proceedings of the European Conference on Computer Vision (ECCV)*, 699–715.

Tan, Wei, Haomin Liu, Zilong Dong, Guofeng Zhang, and Hujun Bao (2013), “Robust monocular SLAM in dynamic environments.” In *ISMAR*, 209–218.

Tang, Chengzhou and Ping Tan (2018), “Ba-net: Dense bundle adjustment network.” *arXiv preprint arXiv:1806.04807*.

Tatarchenko, Maxim, Alexey Dosovitskiy, and Thomas Brox (2017), “Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs.” In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, volume 2, 8.

Tateno, Keisuke, Nassir Navab, and Federico Tombari (2018), “Distortion-aware convolutional filters for dense prediction in panoramic images.” In *Proceedings of the European Conference on Computer Vision (ECCV)*, 707–722.

Tateno, Keisuke, Federico Tombari, Iro Laina, and Nassir Navab (2017), “CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2.

Tompson, Jonathan, Murphy Stein, Yann Lecun, and Ken Perlin (2014), “Real-time continuous pose recovery of human hands using convolutional networks.” *ACM Transactions on Graphics (ToG)*, 33, 1–10.

Torii, Akihiko, Josef Sivic, Tomas Pajdla, and Masatoshi Okutomi (2013), "Visual place recognition with repetitive structures." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 883–890.

Torralba, Antonio and Alexei A Efros (2011), "Unbiased look at dataset bias." In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 1521–1528, IEEE.

Triggs, Bill, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon (1999), "Bundle adjustment—a modern synthesis." In *International workshop on vision algorithms*, 298–372, Springer.

Tsai, Grace, Changhai Xu, Jingen Liu, and Benjamin Kuipers (2011), "Real-time indoor scene understanding using bayesian filtering with motion cues." In *Computer Vision (ICCV), 2011 International Conference on*, 121–128, IEEE.

Uhrig, Jonas, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger (2017), "Sparsity Invariant CNNs." In *International Conference on 3D Vision (3DV)*.

Ummenhofer, Benjamin, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox (2017), "DeMoN: Depth and Motion Network for learning monocular stereo." In *IEEE Conference on computer vision and pattern recognition (CVPR)*, volume 5, 6.

Vincent, Jonathan, Mathieu Labbé, Jean-Samuel Lauzon, François Grondin, Pier-Marc Comtois-Rivet, and François Michaud (2020), "Dynamic object tracking and masking for visual slam." *arXiv preprint arXiv:2008.00072*.

Vysotska, Olga and Cyrill Stachniss (2016), "Lazy data association for image sequences matching under substantial appearance changes." *IEEE Robotics and Automation Letters*, 1, 213–220.

Vysotska, Olga and Cyrill Stachniss (2017), "Relocalization under substantial appearance changes using hashing." In *Proceedings of the IROS Workshop on Planning, Perception and Navigation for Intelligent Vehicles*, volume 24.

Wang, Peng, Xiaohui Shen, Zhe Lin, Scott Cohen, Brian Price, and Alan L Yuille (2015), "Towards unified depth and semantic prediction from a single image." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2800–2809.

Wang, Sen, Ronald Clark, Hongkai Wen, and Niki Trigoni (2017), "DeepVO: Towards end-to-end visual odometry with deep recurrent convolutional neural networks." In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, 2043–2050, IEEE.

Wang, Sen, Ronald Clark, Hongkai Wen, and Niki Trigoni (2018), "End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks." *The International Journal of Robotics Research*, 37, 513–542.

Wang, Yiming, Tongfei Chen, Hainan Xu, Shuoyang Ding, Hang Lv, Yiwen Shao, Nanyun Peng, Lei Xie, Shinji Watanabe, and Sanjeev Khudanpur (2019), "Espresso: A fast end-to-end neural speech recognition toolkit." In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 136–143, IEEE.

Wang, Youbing and Shoudong Huang (2014), "Motion segmentation based robust RGB-D SLAM." In *WCICA*, 3122–3127, IEEE.

Wangsiripitak, Somkiat and David W Murray (2009), "Avoiding moving outliers in visual SLAM by tracking moving objects." In *ICRA*, 375–380, IEEE.

Warburg, Frederik, Soren Hauberg, Manuel Lopez-Antequera, Pau Gargallo, Yubin Kuang, and Javier Civera (2020), "Mapillary street-level sequences: A dataset for lifelong place recognition." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Weerasekera, Chamara Saroj, Thanuja Dharmasiri, Ravi Garg, Tom Drummond, and Ian Reid (2018), "Just-in-time reconstruction: Inpainting sparse maps using single view depth predictors as priors." *arXiv preprint arXiv:1805.04239*.

Weyand, Tobias, Ilya Kostrikov, and James Philbin (2016), "Planet-photo geolocation with convolutional neural networks." In *European Conference on Computer Vision*, 37–55, Springer.

Wohlhart, Paul and Vincent Lepetit (2015), "Learning descriptors for object recognition and 3d pose estimation." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3109–3118.

Xiao, J., K.A. Ehinger, A. Oliva, and A. Torralba (2012), "Recognizing scene viewpoint using panoramic place representation." In *Conference on Computer Vision and Pattern Recognition*, 2695–2702.

Xiao, Jianxiong, Andrew Owens, and Antonio Torralba (2013), "SUN3D: A database of big spaces reconstructed using SfM and object labels." In *Proceedings of the IEEE International Conference on Computer Vision*, 1625–1632.

Xu, Binbin, Wenbin Li, Dimos Tzoumanikas, Michael Bloesch, Andrew Davison, and Stefan Leutenegger (2019), "Mid-fusion: Octree-based object-level multi-instance dynamic slam." In *2019 International Conference on Robotics and Automation (ICRA)*, 5231–5237, IEEE.

Xu, J., B. Stenger, T. Kerola, and T. Tung (2017), "Pano2CAD: Room layout from a single panorama image." In *Winter Conference on Applications of Computer Vision*, 354–362.

Yang, Shang-Ta, Fu-En Wang, Chi-Han Peng, Peter Wonka, Min Sun, and Hung-Kuo Chu (2018), "Dula-net: A dual-projection network for estimating room layouts from a single rgb panorama." *arXiv:1811.11977*.

Yang, Shichao and Sebastian Scherer (2019), "Cubeslam: Monocular 3-d object slam." *IEEE Transactions on Robotics*, 35, 925–938.

Zhan, Huangying, Ravi Garg, Chamara Saroj Weerasekera, Kejie Li, Harsh Agarwal, and Ian Reid (2018), "Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 340–349.

Zhang, Jian, Chen Kan, Alexander G Schwing, and Raquel Urtasun (2013), "Estimating the 3d layout of indoor scenes and its clutter from depth sensors." In *2013 International Conference on Computer Vision*, 1273–1280, IEEE.

Zhang, Lei, Shuai Wang, and Bing Liu (2018), "Deep learning for sentiment analysis: A survey." *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8, e1253.

Zhang, W., W. Zhang, K. Liu, and J. Gu (2017), "Learning to predict high-quality edge maps for room layout estimation." *Transactions on Multimedia*, 19, 935–943.

Zhang, Yinda and Thomas Funkhouser (2018), "Deep Depth Completion of a Single RGB-D Image." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 175–185.

Zhang, Yinda, Shuran Song, Ping Tan, and Jianxiong Xiao (2014), "PanoContext: A whole-room 3D context model for panoramic scene understanding." In *European Conference on Computer Vision*, 668–686, Springer.

Zhao, Hao, Ming Lu, Anbang Yao, Yiwen Guo, Yurong Chen, and Li Zhang (2017), "Physics inspired optimization on semantic transfer features: An alternative method for room layout estimation." *arXiv:1707.00383*.

Zhou, Huizhong, Benjamin Ummenhofer, and Thomas Brox (2018), "DeepTAM: Deep tracking and mapping." In *European Conference on Computer Vision (ECCV)*.

Zhou, Huizhong, Benjamin Ummenhofer, and Thomas Brox (2020), "Deeptam: Deep tracking and mapping with convolutional neural networks." *International Journal of Computer Vision*, 128, 756–769.

Zhou, Tinghui, Matthew Brown, Noah Snavely, and David G Lowe (2017), "Unsupervised learning of depth and ego-motion from video." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1851–1858.

Zhou, Tinghui, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros (2016), "View synthesis by appearance flow." In *European conference on computer vision*, 286–301, Springer.

Zoran, D., P. Isola, D. Krishnan, and W. T. Freeman (2015), "Learning Ordinal Relationships for Mid-Level Vision." In *2015 IEEE International Conference on Computer Vision (ICCV)*, 388–396.

Zou, Chuhang, Alex Colburn, Qi Shan, and Derek Hoiem (2018), "Layoutnet: Reconstructing the 3d room layout from a single rgb image." In *Proceedings Conference on Computer Vision and Pattern Recognition*, 2051–2059.

Appendix A

Detailed Experiments for Camera-Aware Convolutions

A.1 Experiments on Stanford Dataset

A.1.1 2D-3D Semantics Stanford Dataset

In our experiments we used the 2D-3D Semantics Dataset (Armeni et al., 2017), that contains RGB-D equirectangular images. With these images we are able to generate synthetic images with different camera intrinsics. This is essential to evaluate the influence of such intrinsic camera parameters minimizing the effect of the dataset bias.

The dataset is divided into 6 different areas, see Table A.1. The areas in the dataset represent parts of buildings with similarities in their appearance. We used the official train and test splits as suggested by Armeni et al. (2017). The suggested 3-fold cross-validation scheme is shown in Table A.2. All the experiments in this appendix, unless explicitly stated, are performed on the 2D-3D Semantics Dataset.

When training on this dataset, we generate images by randomizing camera parameters. Some parameters remain fixed depending on the experiment (*e.g.* sensor size is fixed during training). Different values of these parameters can be found in Table A.3. Figure A.1 shows several examples of images generated with different camera intrinsics.

A.1.2 Notation

The notation for sensor sizes and focal lengths used during the evaluation is in Table 2.3. As an example, if a network has been trained with sensor sizes 192×256 and 224×224, and focal length 72, we will denote this model as $s_2 s_3 f_{72}$. In some experiments we use a

Area #	**Eq. Images**
	(# images)
1	190
2	299
3	85
4	258
5	373
6	208
Total	**1413**

Table A.1 Statistics of images in the 2D-3D Semantics Dataset (Armeni et al., 2017). Number of images per area.

Fold #	**Training**	**Testing**
	(Area #)	(Area #)
1	1,2,3,4,6	6
2	1,3,5,6	2,4
3	2,4,5	1,3,6

Table A.2 Area assignation for the 3-fold cross validation scheme presented by Armeni et al. (2017) for the 2D-3D Semantics Dataset.

Camera Params	**Range**
	(uniform random sample)
yaw	$(-180°, 180°)$
$pitch$	$(-6°, 6°)$
$roll$	$(-6°, 6°)$
x, y, z	*fixed*[*]
f (focal length)	*depending experiment*
$w \times h$	*depending experiment*

[*] the position of the camera provided for the equirectangular images remains unchanged, we do not use this parameter to generate more images.

Table A.3 Statistics of images in the 2D-3D Semantics Dataset (Armeni et al., 2017). Number of images per area.[*] The position of the camera provided for the equi-rectangular images remains unchanged, we do not use this parameter to generate more images.

random distribution for the focal length. As an example, if the synthesized focal lengths are uniformly distributed between 72 and 128, the model will be denoted as $\mathscr{U} f_{72} f_{128}$.

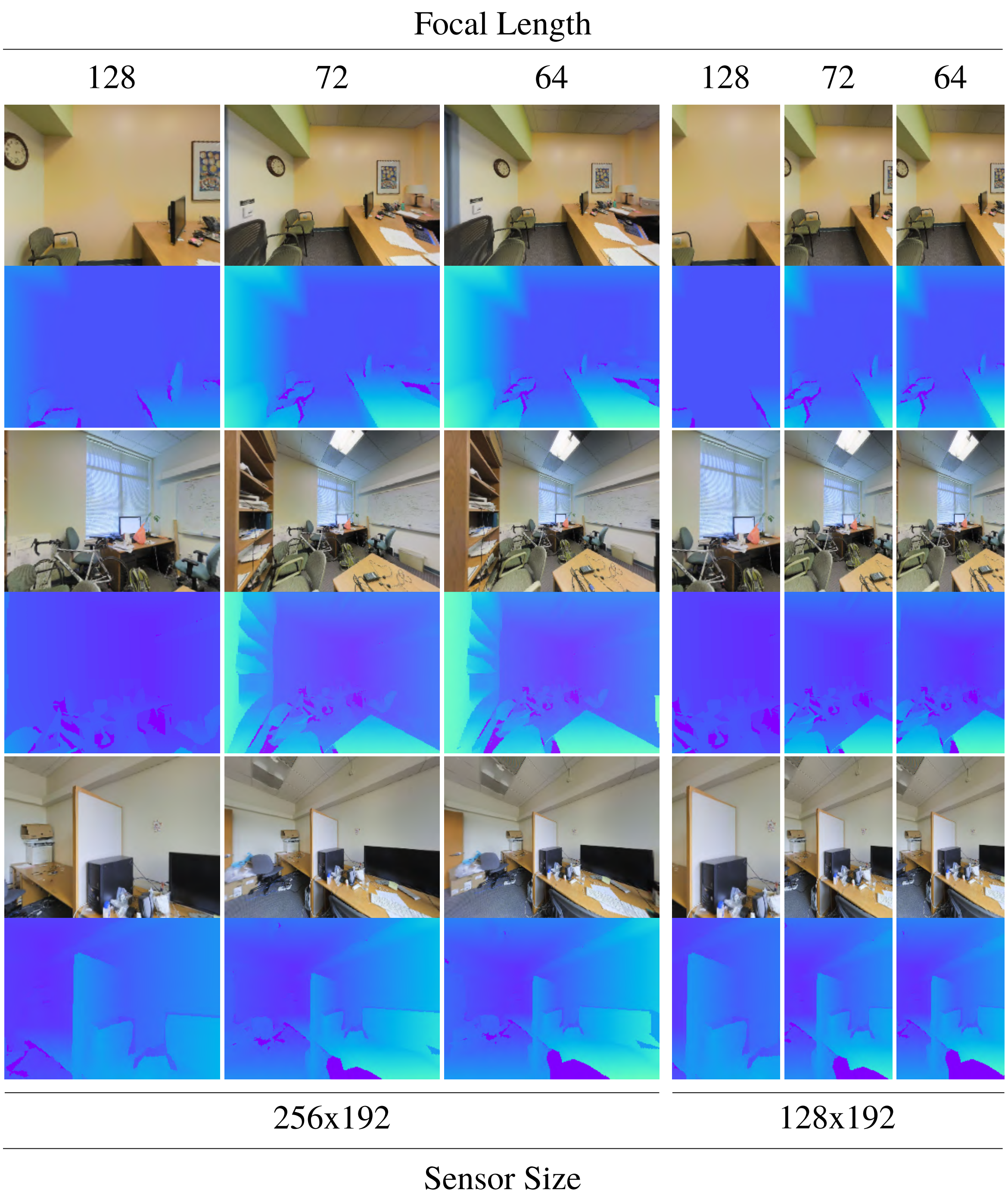


Fig. A.1 Examples of 3 different scenes (in rows) for which we generated several images with different intrinsic parameters.

A.1.3 Influence of Context in Single View Depth

In this section we present the complete results for the experiment analyzing the influence of context. Single-view depth prediction is a task heavily related to context, and that benefits

Focal Length	**Fold #**	abs.rel	l1.inv	rmse	sc.inv	sq.rel	$< 1.25^1$	$< 1.25^2$	$< 1.25^3$
pixels		: 1	$1/m$	m	$log(m)$	: 1	%	%	%
	1	0.17	0.188	0.385	0.0286	0.0428	70.1	96.1	99.3
f_{64}	2	0.198	0.218	0.432	0.05	0.0691	65.1	90.7	98.3
	3	0.155	0.151	0.326	0.0305	0.0451	78.9	97.1	99.6
	1	0.168	0.169	0.411	0.0292	0.0427	70.8	96.2	99.3
f_{72}	2	0.206	0.203	0.474	0.052	0.074	62.8	90.4	97.9
	3	0.147	0.143	0.31	0.0302	0.0403	80.3	97.2	99.6
	1	0.197	0.146	0.547	0.0342	0.0553	61.0	93.6	99.0
f_{128}	2	0.23	0.156	0.586	0.0525	0.0837	56.1	88.2	97.7
	3	0.166	0.117	0.409	0.0329	0.0483	74.5	96.4	99.5
f_{64}	μ	**0.17**	0.184	**0.378**	**0.0347**	**0.048**	**72.1**	**95.5**	**99.2**
f_{72}	μ	**0.17**	0.17	0.4	0.0354	0.0483	71.9	95.3	**99.2**
f_{128}	μ	0.195	**0.141**	0.51	0.0387	0.0606	64.4	93.3	99.0
		smaller is better					*bigger is better*		

Table A.4 Experiment comparing performance on single view depth prediction by changing focal lengths. We evaluate three different focal lengths (f_{64},f_{72} and f_{128}) fixing the sensor size to s_1. The upper part of the table shows the results for each fold. The bottom part shows the median μ of the three folds. As expected having more context improves the depth prediction. In this experiment, focal length f_{64} has the best performance.

Sensor Size	**Fold #**	abs.rel	l1.inv	rmse	sc.inv	sq.rel	$< 1.25^1$	$< 1.25^2$	$< 1.25^3$
pixels		: 1	$1/m$	m	$log(m)$	: 1	%	%	%
	1	0.17	0.188	0.385	0.0286	0.0428	70.1	96.1	99.3
s_1	2	0.198	0.218	0.432	0.05	0.0691	65.1	90.7	98.3
	3	0.155	0.151	0.326	0.0305	0.0451	78.9	97.1	99.6
	1	0.208	0.155	0.598	0.0348	0.0607	55.1	92.4	98.9
s_4	2	0.24	0.162	0.606	0.051	0.0901	54.2	87.4	97.8
	3	0.171	0.121	0.425	0.0319	0.0492	73.0	96.5	99.5
s_1	μ	**0.17**	0.184	**0.378**	**0.0347**	**0.048**	**72.1**	**95.5**	**99.2**
s_4	μ	0.204	**0.146**	0.54	0.0384	0.0637	61.3	93.0	99.0
		smaller is better					*bigger is better*		

Table A.5 Experiment comparing performance on single view depth prediction by changing sensor sizes. We evaluate two different sensor sizes (s_4 ands_1) fixing the focal length to f_{64}. The upper part of the table shows the results for each fold. The bottom part shows the median μ of the three folds. As expected having more context improves the depth prediction. In this experiment, sensor size s_1 has the best performance.

significantly from having more image content.

Context by reducing focal length: Table A.4 shows a detailed version of the context influence experiment. In this experiment we train our network without CAM-Convs with images of different focal lengths (f_{64}, f_{72} and f_{128}) and a fixed sensor size s_1. We report the results for three folds separately, and the median error.

Context by augmenting sensor size: Table A.5 shows a detailed version of the experiment of different versions of the network without CAM-Convs trained on different sensor sizes (s_1, s_4) with a fixed focal length f_{64}. The results are shown for the three folds separately and the median error.

As expected, in both experiments, the results are better for those images with a wider field of view, as the camera captures a larger part of the scene. In the case of fixed sensor size, this corresponds to the camera with the smallest focal length; in the case of fixed focal length, it is the camera with the biggest sensor size. As a curiosity, we found that the L_1 norm on inverse depth (l1.inv in the table) obtains a better score with smaller context in both experiments. We attribute this to the smaller weight that this metric gives to large errors, suggesting that context does not improve the errors uniformly but reduces by a bigger amount large prediction errors.

A.1.4 Focal Length Overfitting

In this section we show the complete results for the focal length overfitting experiments. Table A.6 shows the results by fold and Table A.7 shows the median for all the images on every fold. Unless it is specified otherwise, all the networks have been trained with focal length normalization. We distinguish between three test sets, each one of them generated with one focal length: f_{64}, f_{72} and f_{128}. We train networks on each one of these focal lengths and also train on multiple focal lengths with ($f_{72}f_{128}$ and $\mathscr{U} f_{72}f_{128}$) and without focal length normalization ($f_{72}f_{128}{}^*$).

Notice how networks that have been trained and tested with the same focal length obtain the best results. Also observe how networks that have been trained only in one focal length, generalize poorly when they are tested on different focal length. We also evaluated training on multiple focal lengths without focal length normalization ($f_{72}f_{128}{}^*$ in the table). This does not only perform worse than other approaches, but its convergence was difficult. This is expected, as not normalizing leads to inconsistent target depths.

We found that networks trained on smaller focal lengths tend, in general, to generalize better to bigger focals than the opposite case. See for example the test set f_{128}.

A.1.5 Sensor Size Overfitting

In this section we show the complete results for the two sensor size overfitting experiments. We compare the network version without CAM-Convs trained and tested on different sensor sizes.

Sensor size overfitting: Our first experiment, see Table A.8, shows a by-fold cross-comparison of training a testing in three different sensor sizes, s_1, s_2 and s_3. Results show that training

Test f	**Train f**	**Fold #**	abs.rel	l1.inv	rmse	sc.inv	sq.rel	$< 1.25^1$	$< 1.25^2$	$< 1.25^3$
pixels	*pixels*		: 1	$1/m$	m	$log(m)$	: 1	%	%	%
		1	0.173	0.187	0.391	0.0286	0.0431	69.5	96.0	99.3
	f_{64}	2	0.2	0.212	0.423	0.0492	0.0708	65.2	91.1	98.3
		3	0.154	0.16	0.359	0.0347	0.0422	78.1	96.7	99.4
		1	0.185	0.203	0.424	0.0293	0.0481	64.5	95.2	99.2
	f_{72}	2	0.205	0.223	0.457	0.0519	0.0706	63.2	90.3	98.0
		3	0.145	0.158	0.305	0.0301	0.0376	80.9	97.3	99.5
		1	0.288	0.377	0.647	0.0424	0.101	25.7	75.5	94.6
f_{64}	f_{128}	2	0.246	0.314	0.604	0.0662	0.0885	46.8	81.9	95.2
		3	0.206	0.259	0.45	0.0415	0.0604	56.3	91.0	98.3
		1	0.427	0.685	0.835	0.0405	0.196	02.2	23.6	73.7
	$f_{72}f_{128}$*	2	0.359	0.526	0.765	0.0635	0.149	11.8	54.6	86.6
		3	0.428	0.726	1.0	0.0798	0.227	06.8	30.6	67.1
		1	0.183	0.199	0.414	0.0311	0.048	66.0	95.0	99.2
	$f_{72}f_{128}$	2	0.205	0.217	0.433	0.0514	0.074	64.3	90.6	98.1
		3	0.148	0.154	0.313	0.0307	0.0403	80.0	97.2	99.5
		1	0.186	0.204	0.429	0.0314	0.0493	64.0	94.5	99.1
	$\mathscr{U} f_{72}f_{128}$	2	0.213	0.224	0.457	0.055	0.08	62.2	89.6	97.7
		3	0.155	0.159	0.318	0.0321	0.0442	78.3	96.9	99.5
		1	0.18	0.178	0.43	0.0306	0.0468	67.3	95.5	99.2
	f_{72}	2	0.208	0.203	0.474	0.0524	0.0747	62.5	89.8	97.8
		3	0.148	0.146	0.315	0.0302	0.0393	80.3	97.2	99.5
		1	0.272	0.32	0.643	0.0407	0.0925	30.8	79.6	95.8
	f_{128}	2	0.238	0.271	0.608	0.0624	0.0854	50.7	84.0	95.9
f_{72}		3	0.193	0.22	0.441	0.0392	0.055	60.6	93.1	98.7
		1	0.408	0.582	0.862	0.0407	0.181	03.4	30.2	79.1
	$f_{72}f_{128}$*	2	0.345	0.45	0.798	0.0632	0.141	15.5	60.3	88.4
		3	0.406	0.6	1.02	0.0801	0.208	09.4	36.8	73.0
		1	0.18	0.181	0.432	0.0312	0.0468	66.9	95.2	99.2
	f_{72},f_{128}	2	0.21	0.198	0.458	0.0525	0.0766	63.6	90.0	97.9
		3	0.153	0.148	0.328	0.0311	0.0428	78.7	97.0	99.6
		1	0.183	0.184	0.445	0.0312	0.0483	65.6	94.8	99.1
	$\mathscr{U} f_{72}f_{128}$	2	0.219	0.205	0.475	0.0545	0.0809	61.7	89.2	97.7
		3	0.161	0.149	0.332	0.0322	0.0464	77.4	96.6	99.5
		1	0.179	0.124	0.497	0.0353	0.0504	69.3	95.6	99.3
	f_{72}	2	0.259	0.15	0.623	0.0553	0.105	51.3	84.7	97.4
		3	0.206	0.129	0.472	0.0359	0.0734	66.6	94.2	99.4
		1	0.211	0.158	0.605	0.0352	0.0608	53.9	92.1	98.7
	f_{128}	2	0.232	0.159	0.593	0.053	0.0846	55.6	88.1	97.7
f_{128}		3	0.172	0.127	0.424	0.0346	0.0485	72.0	96.1	99.4
		1	0.261	0.202	0.755	0.046	0.0906	37.8	81.5	97.1
	$f_{72}f_{128}$*	2	0.287	0.208	0.754	0.0658	0.116	39.3	78.0	94.7
		3	0.275	0.215	0.974	0.0806	0.134	40.9	76.7	93.4
		1	0.183	0.134	0.513	0.0329	0.0493	66.3	95.0	99.3
	$f_{72}f_{128}$	2	0.23	0.142	0.576	0.0501	0.0872	57.5	88.6	98.1
		3	0.178	0.12	0.437	0.0329	0.0552	73.1	96.1	99.5
		1	0.184	0.131	0.509	0.0334	0.0497	65.7	95.2	99.3
	$\mathscr{U} f_{72}f_{128}$	2	0.245	0.147	0.596	0.0521	0.0958	54.5	87.1	97.8
		3	0.189	0.123	0.449	0.0332	0.0633	70.5	95.6	99.5
			smaller is better					*bigger is better*		

* the network has been trained without focal length normalization, it is included in this table as a baseline.

Table A.6 Experiments on multiple focal lengths, we evaluate on f_{64},f_{72} and f_{128}. This table shows the results for each fold on each one of the test sets.

Test f	**Train f**	**Fold #**	abs.rel	l1.inv	rmse	sc.inv	sq.rel	$< 1.25^1$	$< 1.25^2$	$< 1.25^3$
pixels	*pixels*		: 1	$1/m$	m	$log(m)$	: 1	%	%	%
f_{64}	$f_{64}{}^*$	μ	**0.17**	**0.184**	**0.378**	**0.0347**	**0.048**	**72.1**	**95.5**	**99.2**
	f_{64}	μ	0.172	**0.184**	0.392	0.0364	0.0488	71.6	95.2	**99.2**
	f_{72}	μ	0.174	0.193	0.395	0.0354	0.0486	70.4	95.0	99.1
	f_{128}	μ	0.247	0.318	0.572	0.0483	0.0826	42.9	83.6	96.4
	$f_{72}f_{128}{}^*$	μ	0.41	0.659	0.864	0.0614	0.193	6.3	34.4	74.7
	$f_{72}f_{128}$	μ	0.176	0.189	0.387	0.0361	0.0503	70.5	94.9	99.1
	$\mathscr{U}\,f_{72}f_{128}$	μ	0.182	0.194	0.398	0.0374	0.0533	68.8	94.4	99.1
f_{72}	$f_{72}{}^*$	μ	**0.17**	**0.17**	**0.4**	**0.0354**	**0.0483**	**71.9**	**95.3**	**99.2**
	f_{72}	μ	0.175	0.174	0.407	0.0364	0.0503	70.6	94.9	99.1
	f_{128}	μ	0.235	0.272	0.564	0.0459	0.076	46.7	86.5	97.1
	$f_{72},f_{128}{}^*$	μ	0.391	0.552	0.888	0.0609	0.179	8.6	41.2	79.5
	f_{72},f_{128}	μ	0.178	0.175	0.404	0.0364	0.052	70.3	94.8	**99.2**
	$\mathscr{U}\,f_{72}f_{128}$	μ	0.184	0.179	0.414	0.0378	0.0553	68.9	94.4	99.1
f_{128}	$f_{128}{}^*$	μ	0.195	0.141	0.51	0.0387	**0.0606**	64.4	93.3	99.0
	f_{72}	μ	0.213	0.133	0.524	0.0411	0.0744	63.4	92.5	99.0
	f_{128}	μ	0.202	0.149	0.532	0.0396	0.0625	61.4	92.7	98.9
	$f_{72},f_{128}{}^*$	μ	0.273	0.208	0.813	0.063	0.11	39.6	78.6	95.1
	f_{72},f_{128}	μ	**0.194**	**0.132**	**0.504**	**0.038**	0.0616	**66.2**	**93.9**	**99.2**
	$\mathscr{U}\,f_{72}f_{128}$	μ	0.202	0.134	0.512	0.0385	0.0663	64.1	93.5	99.1
					smaller is better				*bigger is better*	

* the network has been trained without focal length normalization, it is included in this table as a baseline.

Table A.7 Experiments on multiple focal lengths, we evaluate on f_{64},f_{72} and f_{128}. This table shows the median μ results for each fold on each one of the test sets.

Test	**Train**	**Fold #**	abs.rel	l1.inv	rmse	sc.inv	sq.rel	$< 1.25^1$	$< 1.25^2$	$< 1.25^3$
			: 1	$1/m$	m	$log(m)$	: 1	%	%	%
s_1	s_1	1	0.181	0.151	0.475	0.0319	0.0481	66.8	95.2	99.3
		2	0.229	0.169	0.526	0.0533	0.085	59.3	88.5	98.0
		3	0.171	0.132	0.391	0.0317	0.0528	75.0	96.5	99.5
	s_2	1	0.21	0.18	0.537	0.0351	0.0595	53.3	92.4	98.8
		2	0.24	0.189	0.581	0.0565	0.0873	54.5	86.9	97.2
		3	0.182	0.159	0.422	0.0393	0.0505	66.6	95.2	99.3
	s_3	1	0.199	0.168	0.519	0.0328	0.0555	59.0	93.9	99.0
		2	0.228	0.172	0.535	0.0523	0.0831	59.5	88.3	97.8
		3	0.164	0.135	0.38	0.0333	0.0452	75.5	96.8	99.5
s_2	s_1	1	0.163	0.129	0.432	0.0339	0.0426	72.6	96.2	99.5
		2	0.243	0.185	0.506	0.0546	0.0945	56.4	89.6	98.0
		3	0.196	0.146	0.393	0.0312	0.0652	70.6	95.5	99.5
	s_2	1	0.152	0.121	0.409	0.0263	0.037	77.4	97.2	99.5
		2	0.207	0.162	0.493	0.0462	0.0748	64.8	91.7	98.5
		3	0.152	0.123	0.347	0.0293	0.0415	79.6	97.3	99.7
	s_3	1	0.157	0.126	0.428	0.0294	0.0398	73.9	96.6	99.5
		2	0.22	0.17	0.485	0.0478	0.0811	62.8	91.3	98.5
		3	0.158	0.123	0.353	0.028	0.0457	78.7	96.9	99.7
s_3	s_1	1	0.16	0.129	0.428	0.0312	0.0413	72.9	96.2	99.4
		2	0.231	0.171	0.51	0.0513	0.0868	60.1	89.9	98.2
		3	0.179	0.135	0.387	0.0302	0.0566	73.6	96.2	99.6
	s_2	1	0.172	0.144	0.438	0.0284	0.0429	70.4	96.1	99.3
		2	0.209	0.164	0.511	0.0483	0.0761	63.2	90.4	98.3
		3	0.16	0.131	0.367	0.0335	0.0433	76.9	96.9	99.5
	s_3	1	0.169	0.141	0.442	0.0293	0.0429	70.8	95.9	99.3
		2	0.212	0.16	0.491	0.047	0.0764	63.5	90.8	98.4
		3	0.154	0.123	0.355	0.0287	0.0431	79.1	97.2	99.6
					smaller is better				*bigger is better*	

Table A.8 Experiments on multiple sensor sizes, we evaluate on s_1,s_2 and s_3. We show the versions of the networks in which we use one single sensor size for training. Notice the lack of generalization of the network despite being a FCN. This table shows the results for each fold on each one of the test sets.

Test	**Train**	**Fold #**	abs.rel	l1.inv	rmse	sc.inv	sq.rel	$< 1.25^1$	$< 1.25^2$	$< 1.25^3$
			$:1$	$1/m$	m	$log(m)$	$:1$	%	%	%
s_1	$s_1\ s_2$	1	0.182	0.151	0.514	0.0365	0.0528	67.4	94.2	98.9
		2	0.229	0.18	0.585	0.056	0.0862	57.6	87.7	97.4
		3	0.171	0.133	0.394	0.0332	0.0526	75.1	96.4	99.5
	$s_1\ s_2^\dagger$	1	0.172	0.139	0.456	0.033	0.0462	70.7	95.7	99.4
		2	0.237	0.179	0.574	0.0575	0.0921	56.9	86.8	97.4
		3	0.172	0.135	0.396	0.033	0.0514	74.2	96.4	99.5
	$s_1\ s_2\ s_3$	1	0.178	0.151	0.487	0.0401	0.0514	69.1	94.4	98.8
		2	0.224	0.179	0.531	0.0547	0.0818	59.6	88.2	97.7
		3	0.165	0.14	0.388	0.0355	0.0464	75.1	96.2	99.4
	$s_1\ s_2^\dagger s_3^\dagger$	1	0.191	0.159	0.536	0.0352	0.0531	62.9	93.7	98.9
		2	0.236	0.19	0.613	0.059	0.0912	54.7	86.1	97.1
		3	0.296	0.187	3.08	0.1	1.57	54.6	85.8	95.4
	$s_3\ s_1^\dagger s_2^\dagger$	1	0.203	0.172	0.573	0.0407	0.0589	59.3	92.2	98.4
		2	0.252	0.214	0.671	0.0638	0.0963	49.1	83.3	95.7
		3	0.172	0.138	0.393	0.0349	0.0511	73.9	96.1	99.5
s_2	$s_1\ s_2$	1	0.159	0.127	0.455	0.0321	0.0416	74.4	96.2	99.3
		2	0.209	0.17	0.521	0.051	0.0746	62.3	91.0	98.0
		3	0.161	0.125	0.362	0.0282	0.0476	78.2	97.0	99.7
	$s_1\ s_2^\dagger$	1	0.147	0.116	0.393	0.0279	0.037	78.5	97.4	99.6
		2	0.21	0.165	0.494	0.0476	0.0762	63.1	91.6	98.4
		3	0.163	0.125	0.357	0.0264	0.0457	77.9	97.3	99.7
	$s_1\ s_2\ s_3$	1	0.156	0.128	0.443	0.0345	0.0418	74.9	96.1	99.2
		2	0.21	0.175	0.489	0.0507	0.0757	62.6	90.8	98.1
		3	0.15	0.126	0.345	0.0312	0.0411	79.6	96.9	99.6
	$s_1\ s_2^\dagger s_3^\dagger$	1	0.159	0.13	0.457	0.0303	0.0403	73.6	96.1	99.4
		2	0.203	0.171	0.51	0.0485	0.0708	63.1	91.2	98.2
		3	0.267	0.174	1.71	0.0796	0.483	58.4	88.1	96.6
	$s_3\ s_1^\dagger s_2^\dagger$	1	0.154	0.124	0.466	0.0327	0.0401	75.8	96.2	99.3
		2	0.213	0.179	0.552	0.0549	0.0759	60.5	89.9	97.6
		3	0.168	0.127	0.365	0.0255	0.0504	76.9	97.0	99.7
s_3	$s_1\ s_2$	1	0.165	0.134	0.463	0.0328	0.0436	72.5	95.7	99.2
		2	0.213	0.171	0.543	0.0509	0.0768	61.1	90.2	98.0
		3	0.166	0.128	0.372	0.0302	0.049	76.7	96.9	99.6
	$s_1\ s_2^\dagger$	1	0.149	0.117	0.4	0.0283	0.0374	77.5	97.2	99.6
		2	0.219	0.167	0.525	0.0512	0.0816	61.8	90.4	98.2
		3	0.171	0.129	0.377	0.0285	0.0506	75.4	96.9	99.6
	$s_1\ s_2\ s_3$	1	0.161	0.136	0.445	0.0359	0.0436	74.1	95.6	99.1
		2	0.209	0.17	0.506	0.0503	0.0748	62.8	90.6	98.1
		3	0.155	0.127	0.363	0.0324	0.0424	78.4	96.7	99.5
	$s_1\ s_2^\dagger s_3^\dagger$	1	0.162	0.136	0.462	0.0311	0.042	72.4	95.7	99.3
		2	0.214	0.173	0.549	0.0514	0.0756	60.7	90.3	98.0
		3	0.286	0.178	2.39	0.087	0.912	56.3	86.7	96.2
	$s_3\ s_1^\dagger s_2^\dagger$	1	0.167	0.135	0.494	0.035	0.0446	71.4	95.2	99.0
		2	0.224	0.187	0.602	0.0588	0.0845	58.0	88.1	97.1
		3	0.171	0.13	0.375	0.0295	0.0531	75.0	96.7	99.7
					smaller is better				*bigger is better*	

† the image size has been resized to the first one in the list.

Table A.9 Experiments on multiple sensor sizes, we evaluate on s_1,s_2 and s_3. We show the versions of the networks in which we use multiple sensor sizes for training. Notice the lack of generalization despite being using FCN. This table shows the results for each fold on each one of the test sets.

Test	**Train**	abs.rel	l1.inv	rmse	sc.inv	sq.rel	$< 1.25^1$	$< 1.25^2$	$< 1.25^3$
		: 1	$1/m$	m	$log(m)$	: 1	%	%	%
s_1	s_1	0.189	0.15	**0.46**	**0.037**	**0.0585**	67.6	**94.3**	**99.2**
	s_2	0.206	0.175	0.51	0.0422	0.0621	58.5	92.2	98.7
	s_3	0.193	0.158	0.476	0.0378	0.058	65.4	93.8	99.0
	$s_1\ s_2$	0.191	0.153	0.484	0.0401	0.0606	67.1	93.6	98.9
	$s_1\ s_2^\dagger$	0.188	**0.149**	0.468	0.0391	0.059	68.0	94.2	99.1
	$s_1\ s_2\ s_3$	**0.184**	0.154	0.464	0.0424	0.0571	**68.5**	93.8	98.9
	$s_1\ s_2^\dagger s_3^\dagger$	0.239	0.179	0.742	0.064	0.111	56.9	88.3	97.2
	$s_3\ s_1^\dagger s_2^\dagger$	0.206	0.174	0.53	0.044	0.0654	60.9	91.6	98.4
s_2	s_1	0.197	0.151	0.44	0.038	0.0637	66.7	94.5	99.3
	s_2	**0.166**	**0.133**	0.412	0.0323	0.0468	**74.7**	96.1	99.4
	s_3	0.174	0.138	0.42	0.0334	0.0514	72.5	95.7	99.4
	$s_1\ s_2$	0.173	0.139	0.436	0.0352	0.052	72.5	95.3	99.3
	$s_1\ s_2^\dagger$	0.169	0.134	**0.408**	**0.0318**	**0.0484**	73.9	**96.2**	**99.5**
	$s_1\ s_2\ s_3$	0.168	0.14	0.422	0.0376	0.0498	73.4	95.3	99.2
	$s_1\ s_2^\dagger s_3^\dagger$	0.209	0.16	0.622	0.0514	0.083	63.7	91.7	98.1
	$s_3\ s_1^\dagger s_2^\dagger$	0.174	0.141	0.443	0.0355	0.0521	71.7	95.1	99.2
s_3	s_1	0.184	0.143	0.44	0.0357	0.0574	69.7	94.9	99.3
	s_2	0.177	0.145	0.435	0.0356	0.05	70.6	95.2	99.2
	s_3	0.174	0.14	**0.425**	**0.0336**	**0.0504**	71.9	95.4	99.3
	$s_1\ s_2$	0.178	0.143	0.451	0.0365	0.0537	70.8	94.9	99.2
	$s_1\ s_2^\dagger$	0.175	**0.136**	0.427	0.0342	0.0516	72.3	**95.6**	**99.4**
	$s_1\ s_2\ s_3$	**0.171**	0.143	0.432	0.0383	**0.0504**	**72.5**	95.0	99.1
	$s_1\ s_2^\dagger s_3^\dagger$	0.219	0.164	0.662	0.0552	0.0949	62.1	90.8	97.9
	$s_3\ s_1^\dagger s_2^\dagger$	0.183	0.149	0.473	0.039	0.0577	68.6	94.1	99.0
		smaller is better					*bigger is better*		

† the image size has been resized to the first one in the list.

Table A.10 Experiments on multiple sensor sizes, we evaluate on s_1,s_2 and s_3. Notice the lack of generalization despite being using FCN. This table shows median results for all the folds on each one of the test sets. This table shows that weight sharing during training is the policy that scale the best to more sensor sizes. All the train and test sets focal lengths have been randomly sample between f_{72} and f_{128}.

and testing on the same sensor size always performs better. This suggest overfitting to the sensor size, and shows that the oftenly claimed invariance to the image size of fully convolutional networks (FCN) does not hold for single-view depth prediction.

In our second experiment we allow training on multiple image sizes. In order to do that, we propose two different approaches.

Training on different image sizes (1): Our fist approach is naïve *image resizing*. This means we train the network on a single image input size and resize those images that have a different size to make them fit. This, surprisingly, proved to work with a small number of different sensor sizes (see training rows $s_1 s_2{}^\dagger$ of Table A.9). However it performs poorly when we augment the number of sensor sizes at training time, see training rows $s_1 s_2 s_3{}^\dagger$ and $s_3 s_1 s_2{}^\dagger$. This make sense, as resizing deforms the images a well as their intrinsic parameters. With only two image sizes ($s_1 s_2{}^\dagger$) resizing does not create inconsistencies, and the network it is able to learn from it. But, when adding more image sizes, and vertical and horizonal resizing are needed, the problem becomes more complex and may create inconsistencies in the data. Therefore it is an approach that does not scale.

Training on different image sizes (2): Our second approach uses a *Siamese Architecture* with weight sharing during training. This model seems more coherent, as it keeps the original images (no resizing). However, standard FCN networks were not able to learn from several sensor sizes, and their performance was exactly the same as if training in the test sensor size. *E.g.*, see rows $s_1 s_2$ and $s_1 s_2 s_3$ in Table A.9. In this case adding more sensor sizes during training $s_1 s_2 s_3$ improves. This suggest that adding more could help. However, stacking infinite neural networks (weight sharing) during training is also not scalable.

We show a summary of all the results, related to sensor size overfitting, in Table A.10. Notice how training and testing on the same sensor size outperforms training and testing on the different sensor sizes. It also performs better in most cases compared with training with multiple sensor sizes, even if the test sensor size is included.

A.1.6 Generalization with CAM-Convs

In this section we show the complete evaluation of CAM-Convs in the 2D-3D Semantics Dataset from Stanford (Armeni et al., 2017).

Test	**Train**	**Fold #**	abs.rel	l1.inv	rmse	sc.inv	sq.rel	$< 1.25^1$	$< 1.25^2$	$< 1.25^3$
		: 1	$1/m$	m	$log(m)$	: 1	%	%	%	
s_1	s_1	1	0.181	0.151	0.475	0.0319	0.0481	66.8	95.2	99.3
		2	0.229	0.169	0.526	0.0533	0.085	59.3	88.5	98.0
		3	0.171	0.132	0.391	0.0317	0.0528	75.0	96.5	99.5
	CAM-C s_1 s_2	1	0.173	0.147	0.459	0.0266	0.0433	69.5	96.2	99.4
		2	0.205	0.165	0.503	0.0458	0.0693	62.6	91.3	98.6
		3	0.154	0.12	0.356	0.0261	0.0429	80.0	97.6	99.7
s_2	s_2	1	0.152	0.121	0.409	0.0263	0.037	77.4	97.2	99.5
		2	0.207	0.162	0.493	0.0462	0.0748	64.8	91.7	98.5
		3	0.152	0.123	0.347	0.0293	0.0415	79.6	97.3	99.7
	CAM-C s_1 s_2	1	0.159	0.132	0.425	0.0226	0.037	74.8	97.4	99.6
		2	0.185	0.158	0.448	0.0408	0.0589	67.1	93.3	98.9
		3	0.139	0.109	0.321	0.022	0.0357	84.2	98.2	99.7
s_3	s_3	1	0.169	0.141	0.442	0.0293	0.0429	70.8	95.9	99.3
		2	0.212	0.16	0.491	0.047	0.0764	63.5	90.8	98.4
		3	0.154	0.123	0.355	0.0287	0.0431	79.1	97.2	99.6
	CAM-C s_1 s_2	1	0.163	0.136	0.419	0.0233	0.0378	73.3	97.0	99.5
		2	0.193	0.161	0.467	0.0414	0.0629	66.4	92.7	98.8
		3	0.145	0.112	0.338	0.0234	0.0386	82.2	98.0	99.7
s_1	s_1	μ	0.189	0.15	0.46	0.037	0.0585	67.6	94.3	99.2
	CAM-C	μ	**0.175**	**0.144**	**0.433**	**0.0312**	**0.0498**	**71.0**	**95.7**	**99.4**
s_2	s_2	μ	0.166	0.133	0.412	0.0323	0.0468	74.7	96.1	99.4
	CAM-C	μ	**0.158**	**0.131**	**0.39**	**0.0265**	**0.0417**	**76.5**	**97.0**	**99.5**
s_3	s_3	μ	0.174	0.14	0.425	0.0336	0.0504	71.9	95.4	99.3
	CAM-C	μ	**0.164**	**0.134**	**0.402**	**0.0283**	**0.0441**	**74.6**	**96.6**	**99.5**
			smaller is better					*bigger is better*		

[†] the image size has been resized to the first one in the list.

Table A.11 .Experiments on multiple sensor sizes, we evaluate on s_1,s_2 and s_3. Notice the excelent generalization by including EquiConvs, compare with Table A.10. The upper rows show by-fold results, and the bottom ones show median results for all the folds on each one of the test sets. EquiConvs outperforms even the **same-camera baseline**, and even when the test sensor size s_3 is not among the training ones. All the train and test sets focal lengths have been randomly sample between f_{72} and f_{128}

Train	**Fold #**	abs.rel	l1.inv	rmse	sc.inv	sq.rel	$< 1.25^1$	$< 1.25^2$	$< 1.25^3$
		: 1	$1/m$	m	$log(m)$	: 1	%	%	%
	1	0.155	0.224	0.341	0.0303	0.0383	74.1	95.9	99.2
$s_5 f_{64}$	2	0.203	0.274	0.335	0.0514	0.0742	68.0	91.5	98.3
	3	0.144	0.197	0.242	0.0305	0.0394	81.1	97.0	99.6
	1	0.205	0.246	0.316	0.0549	0.0691	63.7	93.2	99.1
$s_1 f_{64}$	2	0.299	0.353	0.373	0.0756	0.149	48.4	82.1	95.7
	3	0.249	0.294	0.323	0.0524	0.103	57.0	90.3	98.6
	1	0.301	0.293	0.4	0.0348	0.127	43.3	91.6	99.1
$s_1 s_2 \mathscr{U} f_{72} f_{128}{}^{\dagger}$	2	0.389	0.415	0.467	0.0651	0.217	30.8	75.7	94.6
	3	0.421	0.413	0.46	0.0343	0.231	23.2	77.2	97.3
	1	0.171	0.227	0.358	0.0481	0.0528	71.3	94.4	98.8
$s_1 s_2 \mathscr{U} f_{72} f_{128}$	2	0.239	0.325	0.418	0.0735	0.0971	57.2	86.7	96.4
	3	0.187	0.241	0.267	0.0379	0.0625	72.8	94.9	99.3
	1	0.172	0.237	0.34	0.0525	0.0526	71.0	94.1	98.7
$s_1 s_2 s_3 \mathscr{U} f_{72} f_{128}$	2	0.245	0.346	0.379	0.076	0.0998	55.4	86.1	96.2
	3	0.165	0.236	0.264	0.0418	0.0509	75.3	94.9	99.2
	1	0.142	0.2	0.279	0.0307	0.035	77.4	96.8	99.6
CAM-C $s_1 s_2 \mathscr{U} f_{72} f_{128}$	2	0.228	0.298	0.322	0.0549	0.0863	60.5	90.9	98.2
	3	0.175	0.217	0.269	0.0271	0.0544	76.3	96.7	99.7
$s_5 f_{64}$	μ	0.163	0.227	0.309	0.0356	0.0462	75.1	95.3	99.3
$s_1\ f_{64}$	μ	0.245	0.292	0.337	0.0598	0.101	56.6	89.2	98.2
$s_1 s_2 \mathscr{U} f_{72} f_{128}{}^{\dagger}$	μ	0.369	0.369	0.44	0.0427	0.188	32.0	81.9	97.6
$s_1 s_2 \mathscr{U} f_{72} f_{128}$	μ	0.196	0.262	0.343	0.0511	0.0688	67.4	92.6	98.6
$s_1 s_2 s_3 \mathscr{U} f_{72} f_{128}$	μ	0.191	0.269	0.328	0.055	0.0647	67.7	92.5	98.5
CAM-C$\mathscr{U} f_{72} f_{128}$	μ	0.177	0.236	**0.289**	0.0362	0.0541	71.9	**95.4**	**99.4**
				smaller is better				*bigger is better*	

[†] the image size has been resized to the first one in the list.
[*] the network has been trained without focal length normalization.

Table A.12 In order to test the generalization capabilities of the networks we have tested our best performing networks on a different camera model. Upper part shows by-fold results, bottom part shows median results for all the folds on each one of the test sets. CAM-Convs proved to achieve close to the **same-camera baseline** performance while none of the other models show good performance. The test has been done with s_5 images taken with f_{64}.

PP Shift	Crop	Resize	Model	abs.rel	l1.inv	rmse	sc.inv	sq.rel	$< 1.25^1$	$< 1.25^2$	$< 1.25^3$
(w,h)	$W \times H$	$W \times H$		$: 1$	$1/m$	m	$log(m)$	$: 1$	%	%	%
-	-	-	Laina et al. (2016)	0.43	0.323	1.34	0.0518	0.202	5.4	24.8	64.8
			ours	**0.228**	**0.114**	**0.732**	**0.0496**	**0.076**	**52.0**	**87.9**	**98.2**
-	-	256×192	Laina et al. (2016)	0.298	0.107	0.784	**0.041**	0.157	53.0	89.1	97.1
			ours	**0.234**	**0.101**	**0.681**	0.0441	**0.0852**	**56.3**	**90.5**	**99.4**
-	640×448	320×224	Laina et al. (2016)	**0.175**	**0.0743**	**0.533**	**0.0392**	**0.0626**	**76.4**	**94.6**	98.9
			ours	0.211	0.0911	0.657	0.0425	0.0687	62.3	92.3	**99.5**
-	380×380	256×256	Laina et al. (2016)	0.245	0.125	0.826	**0.0328**	0.0766	36.7	89.9	98.7
			ours	**0.196**	**0.0854**	**0.658**	0.0334	**0.058**	**62.8**	**93.0**	**99.6**
$(40,-50)$	352×352	256×256	Laina et al. (2016)	0.279	0.156	0.902	0.0317	0.0919	23.9	82.9	98.1
			ours	**0.194**	**0.0864**	**0.64**	**0.0303**	**0.0552**	**63.3**	**93.7**	**99.7**
$(-20,15)$	186×465	128×320	Laina et al. (2016)	0.285	0.158	0.948	0.0281	0.0957	22.7	79.4	98.3
			ours	**0.197**	**0.0883**	**0.668**	**0.0245**	**0.0556**	**62.0**	**94.8**	**99.9**

Table A.13 Experiments on NYUv2 Silberman et al. (2012). We compare our model on six different cameras (see Figure A.3 for details on how to generate images from different cameras) against Laina Laina et al. (2016). Notice how the errors of our network are almost constant despite changing the camera parameters. The performance of Laina et al. (2016) degrades as the test camera parameters move away from the training camera parameters. The first test (first two rows) was done with the original image size of the dataset, which is 640×480.

Improving same-camera baseline: In Table A.11 we compare our model with CAM-Convs trained on two different sensor sizes s_1 and s_2 and tested in those two plus an different sensor size s_3. In all cases focal length has been randomly sampled between f_{72} and f_{128}. This experiment shows that our model with CAM-Convs is the best one on the three test sets, proving that CAM-Convs generalize to multiple sensor sizes. Compare with results in Table A.10.

Generalizing to a different camera: We sought for a more extreme case of generalization. We tested our trained networks on a complete different camera with sensor size s_5 and focal f_{64}, numbers can be seen in Table A.12. In this experiments we show that our model with CAM-Convs is the only one capable of performing at a similar level to the same-camera baseline.

Qualitative results comparing network trained on same data with and without CAM-Convs can be seen in Figure A.2.

A.2 NYU Experiment

In our last experiment we demonstrate how CAM-Convs can generalize across datasests by training on four datasets with different cameras (KITTI (Uhrig et al., 2017), ScanNet(Dai et al., 2017a), MegaDepth Li and Snavely (2018) and Sun3D(Xiao et al., 2013) and testing on a different one (NYUv2 (Silberman et al., 2012)).

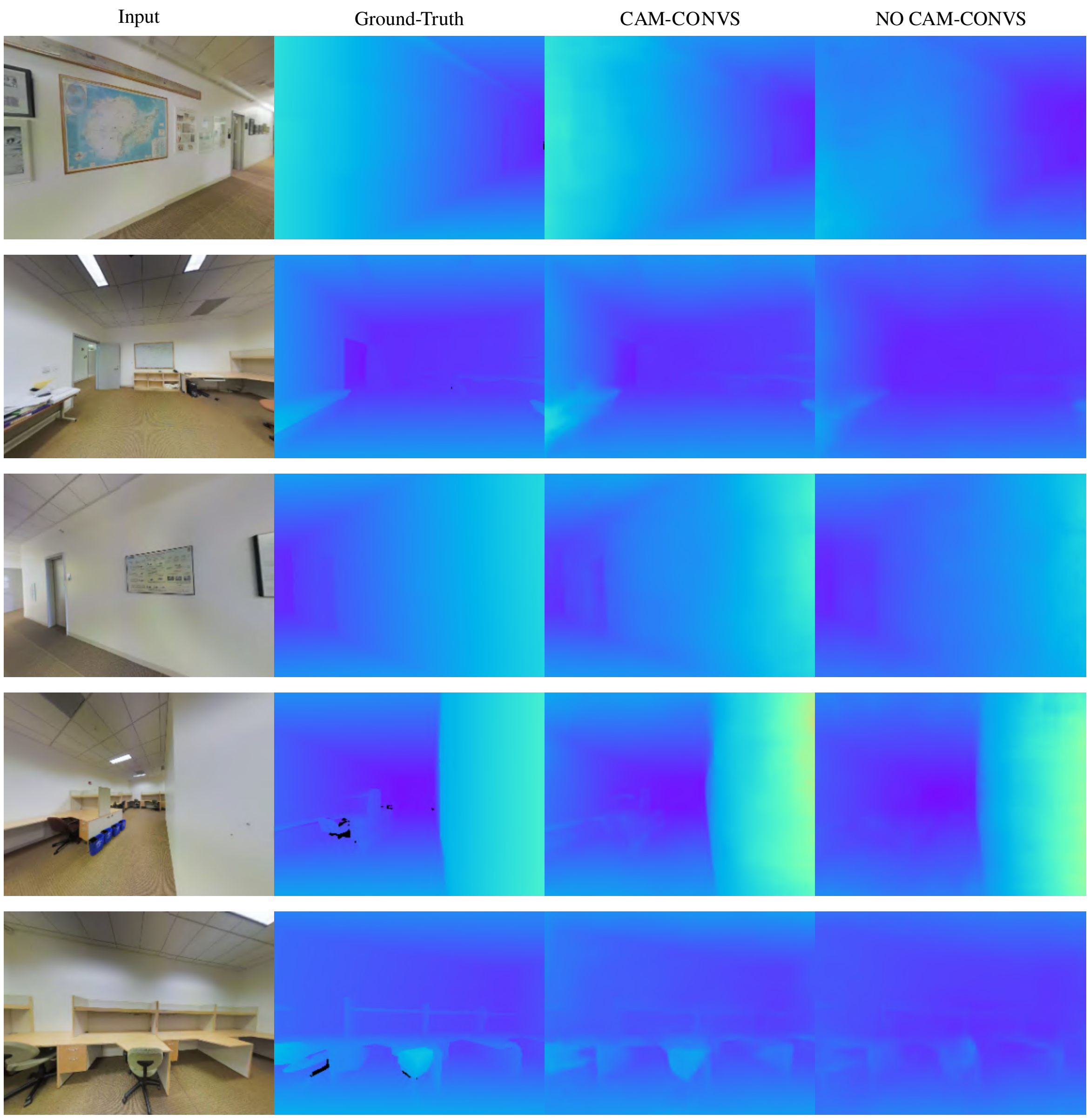


Fig. A.2 Qualitative results on 2D-3D Semantincs Stanford Dataset (Armeni et al., 2017). Sensor size 320×256 and focal length f_{64}. Networks have been trained on 256×192 and 192×256 with focal lengths between f_{72} and f_{128}.

A.2.1 Training

We trained our network for three different sensor sizes (320×320, 256×256 and 224×224) using weight sharing. We did not apply focal length normalization. We augmented the training data by synthetically generating new camera parameters (see Figure A.3) for the given training images:

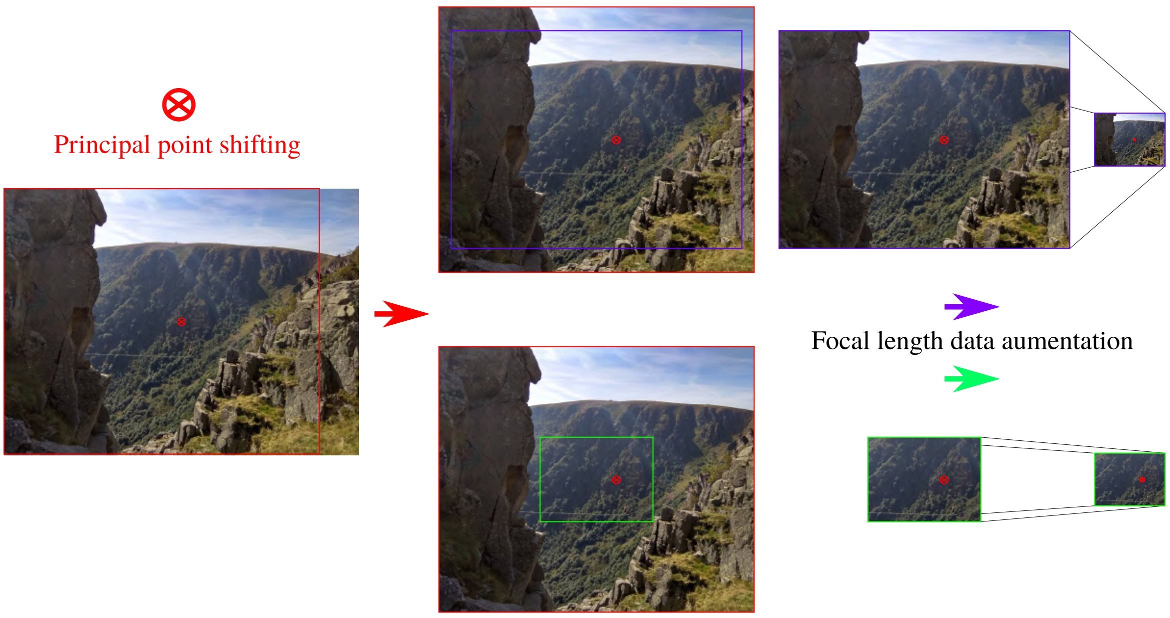


Fig. A.3 Overview of the steps for camera data augmentation.

Principal point shifting: In order to move the principal point in the image, we use cropping. We randomly shift the principal point in the original images with a maximum of 50 pixels on each direction.

Random focal length: Every branch of the network has a fixed sensor size. In order to randomize the focal length we apply a random-size crop (in the center of the image) and resize it to the sensor size of the branch. The crop aspect ratio will be the same as the branch sensor size. Focal length is recalculated afterwards.

A.2.2 Testing

We test 6 different cameras on the NYUv2 dataset, presented by Silberman et al. (2012). To create these cameras we apply the same augmentation pipeline as shown in Figure A.3. For each of the 6 cameras we apply the corresponding augmentation parameters to the whole test set. Table A.13 shows numbers for each camera. We compare our results with Laina et al. (2016), which is a state-of-the-art method on this dataset. Our network has not been fine-tuned for any of the 6 cameras. Results show that our method adapts better to different cameras, while for Laina et al. (2016) the performance drops significantly when the camera changes (the 3rd camera in the table corresponds to the intrinsics of the training data used in Laina et al. (2016)).