Samuel Bruno Berenguel Baeta

Learning the surroundings: 3D scene understanding from omnidirectional images

Director/es
Guerrero Campo, José Jesús
Bermúdez Cameo, Jesús

ISSN 2254-7606

Universidad de Zaragoza
Servicio de Publicaciones

ISSN 2254-7606

Universidad Zaragoza
1542

Tesis Doctoral

LEARNING THE SURROUNDINGS: 3D SCENE UNDERSTANDING FROM OMNIDIRECTIONAL IMAGES

Autor

Samuel Bruno Berenguel Baeta

Director/es

Guerrero Campo, José Jesús
Bermúdez Cameo, Jesús

UNIVERSIDAD DE ZARAGOZA
Escuela de Doctorado

Programa de Doctorado en Ingeniería de Sistemas e Informática

2023

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

Learning the surroundings:
3D scene understanding from omnidirectional images

Ph.D. Thesis
Samuel Bruno
Berenguel Baeta

Advisors:
Jesús Bermúdez Cameo
José Jesús Guerrero Campo

Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza

Ph.D. Dissertation

Learning the surroundings: 3D scene understanding from omnidirectional images

Author:

Samuel Bruno Berenguel Baeta

Advisors:

Jesús Bermúdez Cameo
José Jesús Guerrero Campo

Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza

December 2023

Agradecimientos

Gracias mamá. Gracias hermano. Gracias familia. Las primeras palabras de esta tesis van para vosotros. Son bastantes años los que llevo en la universidad, con sus altibajos, pero vosotros siempre habéis estado a mi lado. Me habéis animado y apoyado, incluso cuando no entendíais lo que estudiaba o la investigación que he hecho estos últimos años. Las palabras no llegan a expresar lo mucho que agradezco que estéis junto a mi.

También quiero agradecer a todos los amigos que me han acompañado en este camino. Los amigos de siempre, Carlos, Diego, Dani, Carlos, Álvaro, Manu, Guille, Luis, que entre risas me habéis hecho llegar hasta aquí. Los amigos de la carrera, Ari, Pedro, Mario, Casti, Jorge, Asen, hemos crecido juntos como ingenieros, aunque cada uno estemos en un lugar distinto. Nunca olvidaré esas cervezas riendo y recordando los años de universidad, viéndola ahora desde el otro lado de la clase. Y todas las nuevas amistades que han surgido en el laboratorio. No solo nos hemos animado unos a otros en temporada de congresos y post-revisiones (*maldito revisor 2....*), sino que los *team building* y ”pequeños”*caffe breaks* han sido momentos con intercambios de ideas y sobretodo, muchas risas. Gracias a Rafa y Carlos por vuestra música; Javieres, Juanjo y Julio por tantas charlas en los cafés (y cervezas); Samu por sus asados; Nacho con su grandísimo humor, juegos y excursiones; Lu-chan con su visión más lingüística; Eloy, María y Javi por los nuevos aires al laboratorio; la comunidad internacional encabezada por Cesar, Rodrigo y Sof; los profes más enrollados del departamento, Alejandro, Edgar, Edu, Rosario, Gonzalo y; por supuesto, también te lo quiero agradecer a ti, Irene, por cambiarme la vida de una forma que nunca soñé fuera posible. Gracias a todos vosotros el doctorado ha sido una de las mejores experiencias de mi vida.

Finalmente, y no me olvido, quiero agradecer a quienes han hecho posible esta experiencia. Ellos, que han tenido que aguantar mis inseguridades y corregir mis errores. Ellos, que me han animado a continuar con lo que me gustaba, con la verdad y realidad como bandera, pero siempre levantándome los ánimos y encendiendo la llama de la ilusión para poder seguir adelante. Quiero terminar agradeciendo a mis directores por todo lo que han hecho por mi. Gracias Jesús. Gracias Josechu.

Abstract

Neural networks have become widespread all around the world and are used for many different applications. These new methods are able to recognize music and audio, generate full texts from simple ideas and obtain detailed and relevant information from images and videos. The possibilities of neural networks and deep learning methods are uncountable, becoming the main tool for research and new applications in our daily-life. At the same time, omnidirectional and 360 images are also becoming widespread in industry and in consumer society, causing omnidirectional computer vision to gain attention. From 360 images, we capture all the information surrounding the camera in a single shot.

The combination of deep learning methods and omnidirectional computer vision have attracted many researchers to this new field. From a single omnidirectional image, we obtain enough information of the environment to make a neural network understand its surroundings and interact with the environment. For applications such as navigation and autonomous driving, the use of omnidirectional cameras provide information all around the robot, person or vehicle, while conventional perspective cameras lack this context information due to their narrow field of view. Even if some applications can include several conventional cameras to increase the system's field of view, tasks where weight is more important (i.e. guidance of visually impaired people or navigation of autonomous drones), the less cameras we need to include, the better.

In this thesis, we focus in the joint use of omnidirectional cameras, deep learning, geometry and photometric methods. We evaluate different approaches to handle omnidirectional images, adapting previous methods to the distortion of omnidirectional projection models and also proposing new solutions to tackle the challenges of this kind of images. For indoor scene understanding, we propose a novel neural network that jointly obtains semantic segmentation and depth maps from single equirectangular panoramas. Our network manages, with a new convolutional approach, to leverage the context information provided by the panoramic image and exploit the combined information of semantics and depth. In the same topic, we combine deep learning and geometric solvers to recover the scaled structural layout of indoor environments from single non-central panoramas. This combination provides a fast implementation, thanks to the learning approach, and accurate result, due to the geometric solvers. Additionally, we also propose several approaches of network adaptation to the distortion of omnidirectional projection models for outdoor navigation and domain adaptation of previous solutions. All in all, this thesis looks for finding novel and innovative solutions to take advantage of omnidirectional cameras while overcoming the challenges they pose.

Resumen

Las redes neuronales se han extendido por todo el mundo, siendo utilizadas en una gran variedad de aplicaciones. Estos métodos son capaces de reconocer música y audio, generar textos completos a partir de ideas simples u obtener información detallada y relevante de imágenes y videos. Las posibilidades que ofrecen las redes neuronales y métodos de aprendizaje profundo son incontables, convirtiéndose en la principal herramienta de investigación y nuevas aplicaciones en nuestra vida diaria. Al mismo tiempo, las imágenes omnidireccionales se están extendiendo dentro de la industria y nuestra sociedad, causando que la visión omnidireccional gane atención. A partir de imágenes 360 capturamos toda la información que rodea a la cámara en una sola toma.

La combinación del aprendizaje profundo y la visión omnidireccional ha atraído a muchos investigadores. A partir de una única imagen omnidireccional se obtiene suficiente información del entorno para que una red neuronal comprenda sus alrededores y pueda interactuar con el entorno. Para aplicaciones como navegación y conducción autónoma, el uso de cámaras omnidireccionales proporciona información en torno del robot, persona o vehículo, mientras que las cámaras convencionales carecen de esta información contextual debido a su reducido campo de visión. Aunque algunas aplicaciones pueden incluir varias cámaras convencionales para aumentar el campo de visión del sistema, tareas en las que el peso es importante (P.ej. guiado de personas con discapacidad visual o navegación de drones autónomos), un número reducido de dispositivos es altamente deseable.

En esta tesis nos centramos en el uso conjunto de cámaras omnidireccionales, aprendizaje profundo, geometría y fotometría. Evaluamos diferentes enfoques para tratar con imágenes omnidireccionales, adaptando métodos a los modelos de proyección omnidireccionales y proponiendo nuevas soluciones para afrontar los retos de este tipo de imágenes. Para la comprensión de entornos interiores, proponemos una nueva red neuronal que obtiene segmentación semántica y mapas de profundidad de forma conjunta a partir de un único panorama equirectangular. Nuestra red logra, con un nuevo enfoque convolucional, aprovechar la información del entorno proporcionada por la imagen panorámica y explotar la información combinada de semántica y profundidad. En el mismo tema, combinamos aprendizaje profundo y soluciones geométricas para recuperar el diseño estructural, junto con su escala, de entornos de interior a partir de un único panorama no central. Esta combinación de métodos proporciona una implementación rápida, debido a la red neuronal, y resultados precisos, gracias a las soluciones geométricas. Además, también proponemos varios enfoques para la adaptación de redes neuronales a la distorsión de modelos de proyección omnidireccionales para la navegación y la adaptación del dominio soluciones previas. En términos generales, esta tesis busca encontrar soluciones novedosas e innovadoras para aprovechar las ventajas de las cámaras omnidireccionales y superar los desafíos que plantean.

Contents

Chapter 1

Introduction

1.1 Motivation

Deep learning, neural networks, artificial intelligence... These terms appear everyday on any research field, in the news and in many conversations. One of the threads of thought is that artificial intelligence is going to take over the world as in the famous film, Matrix. In the meantime, researchers that work and develop deep learning methods for autonomous driving struggle to make networks differentiate between a carriage and a lorry[1]. Nevertheless, it is true that deep learning methods have become widespread in many research fields and applications, obtaining impressive results for many different tasks and from diverse sources of information (i.e. images Goodfellow et al. (2014), audio Hernandez-Olivan and Beltran (2022), text Brown et al. (2020), video Caron et al. (2021)).

Among these sources of information, images are the most used and the computer vision community was the first to vastly exploit the power and versatility of neural networks. From the first convolutional neural network (CNN) presented for digit recognition LeCun et al. (1989) to the newest generative models Rombach et al. (2022), the computer vision research community has been entangled with deep learning methods, pushing the future forward. Furthermore, given that vision is the human sense from which more information we can obtain, the use of cameras and images have been standard practice from several applications, i.e. autonomous driving, navigation, recognition, robot grasping, scene understanding, augmented reality... Some of these tasks, such as visual odometry or 3D reconstruction, were accurately solved with classical methods (i.e. photometry, multi-view geometry). However, these classical methods need prior knowledge or several constraints to get the final results and, often, the solutions are obtained with computationally expensive algorithms which lead to slow implementations for more complex problems. On the other hand, deep learning methods provide good results in a faster way, but not as accurate. In this crossroad appears a humble research line yet to be exploited: combining classic and novel methods, i.e. use deep learning methods as first solution for the geometric method, obtaining a good first estimation and refining the final solution with the more accurate geometric

[1]Video taken from Youtube: https://youtube.com/shorts/oYmDIPNd9hM?feature=share

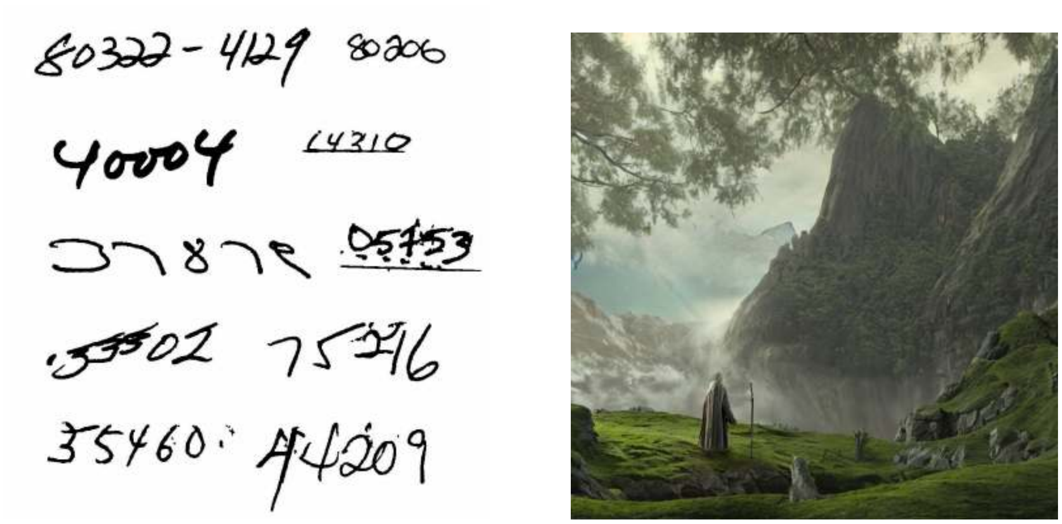


Figure 1.1: Two examples of past and present in the computer vision community. On the left side, the data used by LeCun et al. (1989) for digit recognition. On the right side, image generated by Rombach et al. (2022) from a text prompt.

solution. This combinations allows to solve complex geometric problems in a much faster fashion and with less initial constraints.

As mentioned before, vision and images are a source of information of great wealth and possibilities. And today, we have more possibilities than ever since we have a great variety of cameras in the market with different properties as well as a world wide web to obtain images and information from all over the world. The most commonly used camera is the perspective, or pin-hole, camera. Most of deep learning methods, photometric algorithms and computer vision solutions are dedicated to this kind of camera. It presents several advantages as it is its price, since it is really cheap, and size, small enough to be integrated in most devices. It is also convenient since the image does not present mayor distortions, more than small artefacts or vignetting in the corners of the image. However, this camera present a mayor disadvantage for most robotics applications and autonomous systems: lack of environment's context information around the camera. This camera presents a narrow field of view (i.e. typically, the field of view ranges from 40 to 60 degrees). With this few information of the environment, it is difficult for a computer to understand, interpret and interact with it surroundings with the same efficiency as human beings (i.e. the horizontal field of view of each human eye is about 140 degrees and up to 220 degrees with the combination of both eyes). Luckily for the computer vision community, new devices appear everyday that can overcome this mayor disadvantage.

In recent years, new devices have appeared that allows to obtain omnidirectional images with low computational expense and at a low price. From fisheye lenses and super wide field of view lenses to 360 cameras, the market is flooded with new possibilities. The computer vision research community reflects this new opportunities increasing the number of solutions, applications and methods with these devices. The new challenges that omnidirectional cameras present and the numerous advantages have taken a lot of interest and new research lines are appearing everyday. These advantages come with new challenges, such as the image distortion, which must be addressed in a smart way to achieve good performances. The bigger size of current omnidirectional devices with respect to perspective cameras creates a technical disadvantage that is tackled in the case of multi-camera setups, reducing the image processing time and number of components. Furthermore, novel omnidirectional devices are jet to come, such as non-central systems. Even if we do not have many commercial devices, non-central systems provide more advantages to the omnidirectional view. Contrary to central systems, from calibrated non-central cameras we can retrieve the scale of the environment, solving one of the main challenges of computer vision. However, the slightly different image distortion and sensitivity to noise of these non-central systems are big challenges that need to be solved. However, even if omnidirectional cameras present different challenges than perspective cameras, omnidirectional cameras present one main advantage: their field of view. From a single image, information of all the surroundings of the camera is captured and can be processed at once. This is a key aspect for some computer vision and robotics tasks, such as navigation, autonomous driving and 3D scene understanding. Besides, when neural networks enter in the frame, providing more context information improve their performance, obtaining more accurate results and more information of the environment at once. So, this is the research context in which this thesis takes action.

In this thesis we have studied different solutions with the use of omnidirectional cameras, deep learning and geometric solutions. We search solutions for different problems, from scene understanding to navigation. We also explore the use of several omnidirectional cameras where we will see that the distortion of these projection models is not always a problem and it can be the solution. Our research aims to exploit the many advantages of omnidirectional cameras and adapt current deep learning methods to these challenging images. We also explore new methods and tools to attain novel solutions which take advantage of omnidirectional vision and take a step forward the general use of these cameras. The document you are about to read is a collection of our efforts in this topic, providing enough explanations to replicate our research and experiments evaluating the performance of the several methods and solutions developed in the last years. As author of this thesis, I hope you enjoy our work, that you find it inspiring and enriching and may help future generations in the advance of this wonderful research field.

1.2 Life of a Ph.D student

In this section I present the results of my PhD career. These results are my several contributions, such as publications, public code and datasets. I also include mentorship opportunities in the time of the PhD.

1.2.1 Contributions and publications

In our first line of research, we gathered several central and non-central projections systems and propose a tool for synthetic image generation from virtual environments. The main advantage of our work is the possibility of generating custom-made datasets with ground truth knowledge for several tasks as object detection, semantic or instance segmentation, depth estimation, visual odometry, etc. Besides, due to the recent increase of deep learning approaches for specialized tasks and the lack of annotated visual data, this image generator can increase the number and size of existing datasets, helping in the development of novel and more precise deep learning methods. As a proof of concept, we created and presented a synthetic dataset of omnidirectional images for depth estimation, instance and semantic segmentation and layout reconstruction. The main publications in this field are:

- *Omnidirectional Image Data-set for Computer Vision Applications*,
  Bruno Berenguel-Baeta, Jesus Bermudez-Cameo, Jose J. Guerrero.
  Jornada de Jóvenes investigadores del I3A, Volume 8 (Actas de la IX Jornada de Jóvenes Investigadores del I3A), length 2 pages.
  Universidad de Zaragoza, 2020.
  ISSN:2341-4790.
  DOI:doi.org/10.26754/jjii3a.4869

- *OmniSCV: an Omnidirectional Synthetic Image Generator for Computer Vision*,
  Bruno Berenguel-Baeta, Jesus Bermudez-Cameo, Jose J. Guerrero.
  Sensors, Volume 20, Number 7, Page 2066, length 24 pages.
  MDPI, 2020.
  DOI:doi.org/10.3390/s20072066

In the field of indoor scene understanding, we have several contributions in two different lines. In a first line, we search to make a computer understand its surroundings, giving tools and information to interact with the objects and navigate in the environment. We achieve 360 information with the use of equirectangular panoramas and deep learning, providing depth and semantic information. We propose the use of projection-model-aware convolutions and convolutions in the frequential domain to adapt the network to the panoramic distortion. The related publications in this research are:

- *FreDSNet: Monocular Depth and Semantic Segmentation from Equirectangular Panoramas*,
  Bruno Berenguel-Baeta, Jesus Bermudez-Cameo, Jose J. Guerrero.
  2023, *Journal paper submitted on 19th April 2023 and is currently Under review*, length 14 pages.

- *FreDSNet: Joint Monocular Depth and Semantic Segmentation with Fast Fourier Convolutions from Single Panoramas*,
  Bruno Berenguel-Baeta, Jesus Bermudez-Cameo, Jose J. Guerrero.
  International Conference on Robotics and Automation, Pages 6080–6086, length 6 pages.
  IEEE Xplore, 2023.
  DOI:doi.org/10.1109/ICRA48891.2023.10161142

In a second line of research in the indoor scene understanding field, we aim to recover 3D scaled layouts from single panoramas. For this purpose, we present a combination of deep learning and geometric methods with the use of non-central panoramas, exploiting their unique geometrical properties. With deep learning methods we solve the photometric problem in an efficient way while we obtain accurate 3D reconstructions with geometric methods. From this second line of research, we have the following publications:

- *Atlanta Scaled Layouts from Non-Central Panoramas*,
  Bruno Berenguel-Baeta, Jesus Bermudez-Cameo, Jose J. Guerrero.
  Pattern Recognition, Volume 129, Pages 108740, length 12 pages.
  Elsevier, 2022.
  DOI:doi.org/10.1016/j.patcog.2022.108740

- *Scaled 360 Layouts: Revisiting Non-Central Panoramas*,
  Bruno Berenguel-Baeta, Jesus Bermudez-Cameo, Jose J. Guerrero.
  OmniCV Workshop of the International Conference on Computer Vision and Pattern Recognition, Pages 3702–3705, length 4 pages.
  IEEE/CVF, 2021.
  DOI:doi.org/10.1109/CVPRW53098.2021.00410

With the use of deep learning methods, the distortion of omnidirectional images can be a big issue that many researches have tried to solve. Current computer vision methods mostly use perspective images, with low distortion, or make smaller crops or patches of omnidirectional images to avoid the problems generated by the distortion of these images. Even if results are promising, these solutions present discontinuities and loss of context information. With the distortion of omnidirectional images in mind, we aim to help neural networks to handle and adapt to this distortion. Our first approach is to deform the convolutional kernels to fit the distortion of the images, taking advantage of previous deep learning solutions and adapt them to fisheye images. In this line, we have an ongoing work:

- *Convolution kernel adaptation to calibrated fisheye*,
  Bruno Berenguel-Baeta, Maria Santos-Villafranca, Jesus Bermudez-Cameo, Alejandro Perez-Yus, Jose J. Guerrero,
  British Machine Vision Conference 2023, length 10 pages.

Last but not least, navigation of autonomous vehicles and helps for visually impaired people are hot topics in the computer vision and robotics community. Cameras provide lots of information of the environment. And, when omnidirectional cameras are involved, we can obtain up to 360 information surrounding the autonomous system or person, providing better and safer guidance in any direction. In this research line, we have the following publications:

- *Visual Gyroscope: Combination of Deep Learning Features and Direct Alignment for Panoramic Stabilization*,
  Bruno Berenguel-Baeta, Antoine N. Andre, Guillaume Caron, Jesus Bermudez-Cameo, Jose J. Guerrero,
  OmniCV Workshop of the International Conference on Computer Vision and Pattern Recognition, Pages 6444–6447, length 4 pages.
  IEEE/CVF, 2023.
- *Floor Extraction and Door Detection for Visually Impaired Guidance*,
  Bruno Berenguel-Baeta, Manuel Guerrero-Viu, Alejandro de Nova, Jesus Bermudez-Cameo, Jose J. Guerrero,
  International Conference on Control, Automation, Robotics and Vision, Pages 1222–1229, length 7 pages.
  IEEE, 2020.
  DOI:doi.org/10.1109/ICARCV50220.2020.9305464

1.2.2 Public software and datasets

Associated with some of the previous publications, computer programs and code have been generated which has been published in my personal GitHub web-page https://github.com/Sbrunoberenguel. As summary:

- The Omnidirectional Synthetic image generator for Computer Vision (OmniSCV) Berenguel-Baeta et al. (2020a) is publicly available since 2020 in:
  https://github.com/Sbrunoberenguel/OmniSCV
- Code for scaled layout recovery Berenguel-Baeta et al. (2022), including deep learning and geometric solutions is available in:
  https://github.com/Sbrunoberenguel/scaledLayout

- For indoor scene understanding from equirectangular panoramas Berenguel-Baeta et al. (2023b), FreDSNet, can be obtained in:
  https://github.com/Sbrunoberenguel/FreDSNet

- The deep learning part of the code developed in the research stay for equirectangular image stabilization is available in:
  https://github.com/Sbrunoberenguel/HoLiNet
  The photometric part can be found from the same web-page.

- Code to compute the deformable calibrated convolutional kernels is publicly available in
  https://github.com/Sbrunoberenguel/CalibratedConvolutions

Furthermore, published works are accompanied by datasets generated to develop our proposals. One of the datasets can be found from the page of OmniSCV, where a number of omnidirectional images from several projection models are generated from a photorealistic virtual environment.

Besides, after we published our layout recovery solution Berenguel-Baeta et al. (2022), we also published the first dataset of non-central panoramas in the literature. This dataset is published in:

- *Non-central panorama indoor dataset*,
  Bruno Berenguel-Baeta, Jesus Bermudez-Cameo, Jose J. Guerrero,
  Data in Brief, Volume 43, Pages 108375, length 4 pages.
  Elsevier, 2022.
  DOI:https://doi.org/10.1016/j.dib.2022.108375
  Publicly available from:
  https://github.com/jesusbermudezcameo/NonCentralIndoorDataset

1.2.3 Research stays

Along with the research made in the *Universidad de Zaragoza* and, in spite of the COVID-19, I also had the opportunity to travel abroad and make a research stay in the *Joint CNRS-AIST Robotics Laboratory* of Tsukuba, Japan, under the supervision of Prof. Guillaume Caron. For 3 months in my last year as PhD student, I worked hand in hand with Guillaume and a post-doctoral student, Antoine Andre, trying to improve previous research on image stabilization from panoramic cameras. Our joint work was gifted with a publication in the workshop on omnidirectional computer vision of CVPR'23 Berenguel-Baeta et al. (2023a).

1.2.4 And more

Along with the research career, I made some incursions in the academic field mentoring students in their bachelor and master thesis:

- *Depth computation from a stereo fish eye camera*
  Cesar Saenz Rodriguez
  Bachelor thesis in Industrial Engineering at Universidad de Zaragoza, 2023
- *Object detection and semantic segmentation on panoramic images*
  Alejandro de Nova Guerrero
  Master thesis in Industrial Engineering at Universidad de Zaragoza, 2021

1.3 Overview of this document

This document is composed by a collection of chapters where we describe our research with omnidirectional cameras and imagery. As brief summary and introduction, the next chapter is dedicated to present several projection models of omnidirectional cameras, including the examples of devices for the acquisition of the images and real images of these cameras.

In Chapter 3 we present our omnidirectional image simulator. We provide an introduction to virtual environments and the Unreal Engine 4 in which is build the simulator. We also present how we generate the omnidirectional images and several examples of use of these synthetic images.

Chapters 4 and 5 present our works on indoor scene understanding. We first introduce the deep neural network created for semantic segmentation and depth estimation from central panoramas and the results obtained from it. Then, we present the combination of deep learning and geometric methods for scaled layout recovery from non-central panoramas.

The last part of the main body of this thesis is dedicated to our most recent works on practical solutions for navigation and distortion-aware neural networks. Finally, we summarize the thesis' main conclusions and plan our future work. We discuss the research lines presented in the thesis and look a bit into the future, trying to find new solutions to current (or new) problems in the field of omnidirectional computer vision.

Chapter 2

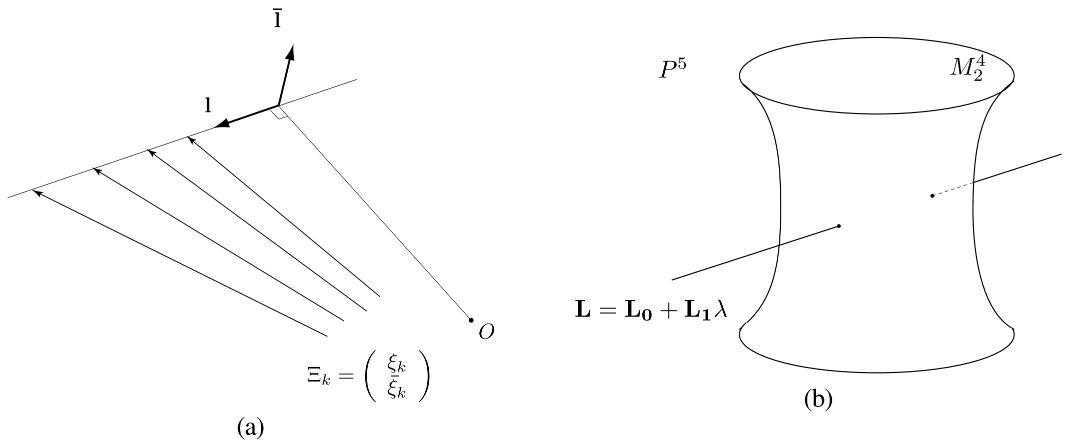
Omnidirectional projection models

Omnidirectional and 360º images are becoming widespread in industry and in consumer society, causing omnidirectional computer vision to gain attention. Their wide field of view allows the gathering of a great amount of information about the environment from only an image. However, the distortion of these images requires the development of specific algorithms for their treatment and interpretation. The first step to understand and use this kind of images is to learn their mathematical projection models. In this chapter, we gather the most used omnidirectional projection models for central and non-central systems.

2.1 Introduction

We summarize and introduce the projection models for several omnidirectional cameras. We are going to explain the relationship between image–plane coordinates and the coordinates of the projection rays or 3D points in the camera reference. We distinguish two types of camera models: central-projection camera models and non-central-projection camera models.
Among the **central projection** cameras, we consider:

- Panoramic images: *Equirectangular*.
- Fisheye cameras Kingslake (1989); Schneider et al. (2009), where we distinguish diverse lenses: *Equi-angular, Stereo-graphic, Equi-solid angle, Orthogonal*.
- Catadioptric systems Baker and Nayar (1999), where we distinguish different mirrors: *Parabolic* and *Hyperbolic*.
- Scaramuzza model for revolution symmetry systems Scaramuzza et al. (2006a,b)
- Kannala–Brandt model for fisheye lenses Kannala and Brandt (2006).



$\bar{\mathbf{l}}$
$\mathbf{l}$
O
$\Xi_k = \begin{pmatrix} \xi_k \\ \bar{\xi}_k \end{pmatrix}$
(a)
P^5
M_2^4
$\mathbf{L} = \mathbf{L_0} + \mathbf{L_1}\lambda$
(b)

Figure 2.1: a) Representation of a 3D line with Plücker coordinates. b) Klein quadric and a 3D line.

Among the **non-central projection** cameras, we consider:

- Non-central panoramas Menem and Pajdla (2004).
- Catadioptric systems, where we distinguish different mirrors: *Spherical* Agrawal and Ramalingam (2013) and *Conical* Lin and Bajcsy (2006).

2.2 Notation: writing math is not confusing

The mathematical notation that we are going to follow is summarized here. Scalar values are presented as standard text (e.g. R_c, φ). For data and vectors that belong to $\mathbb{R}^3$ or $\mathbb{P}^2$, we use bold lower case letters (e.g. **n**), while vectors of greater dimensions are presented as upper case letters in Euler font (e.g. $\mathcal{W}$). Matrices that belong to $\mathbb{R}^{n\times m}\forall\{n>1\wedge m>1\}$ are presented in the serif font as uppercase letters (e.g. A). For projecting rays and 3D lines defined in Plücker coordinates, we use bold uppercase letters (e.g. Ξ, $\mathbf{L}$).

2.2.1 Plücker coordinates

Going into more detail, here we give a brief introduction to Plücker coordinates. An homogeneous representation of a 3D line $\mathbf{L}\in\mathbb{P}^5$ can be described with two vectors $(\mathbf{l}^T,\bar{\mathbf{l}}^T)^T\in\mathbb{R}^3$ with geometrical meaning in the Euclidean geometry, as seen in Fig.2.1a. The first component, $\mathbf{l}\in\mathbb{R}^3$, describes the direction vector of the 3D line while the second component, $\bar{\mathbf{l}}\in\mathbb{R}^3$ is the normal vector of the plane passing through the 3D line and the origin of the reference system. Any 3D point of $\mathbb{P}^5$ corresponding to a 3D line must satisfy the Plücker identity, defined as $\mathbf{l}^T\bar{\mathbf{l}}=0$. This identity is a quadratic constraint which defines a two dimensional subspace called the Klein quadric $M_2{}^4$ Pottmann and Wallner (2001) Fig.2.1b.

2.2.2 How to compute 3D lines

Computing 3D lines from a single image is a particular property we want to exploit. Here we introduce how to compute a 3D line from a non-central projection system. Defining in Plücker coordinates a 3D line as $\mathbf{L}=(\mathbf{l}^T,\bar{\mathbf{l}}^T)^T\in\mathbb{P}^5$ and a projecting ray $\Xi=(\xi^T,\bar{\xi}^T)^T\in\mathbb{P}^5$, their intersection is defined by the side operator Pottmann and Wallner (2001) as:

$$side(\mathbf{L},\Xi)=\mathbf{l}^T\bar{\xi}+\bar{\mathbf{l}}^T\xi=0.$$

Given that a 3D line has four degrees of freedom, we need, at least, 4 independent equations to solve for $\mathbf{L}$. In general, four projecting rays from a 3D line generate four independent constraints from where we can compute the 3D line Teller and Hohmeyer (1999). Notice that in a central system the four equations are not independent given that the four rays are coplanar. We can find some degenerate cases where four projecting rays do not generate independent constraints, e.g. the rays are coplanar with the revolution axis or with the plane containing the circle of optical centres.

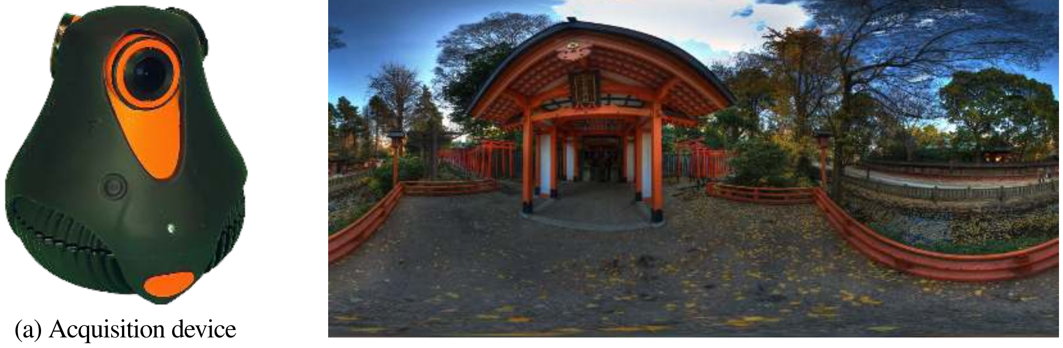
Given that a Plücker coordinate set (without taking into account the Plücker identity) is an over-parametrized description of a line, the solution of a system of four homogeneous equations in $\mathbb{P}^5$ is a one dimensional subspace of $\mathbb{P}^5$. The solution of this equation system can be expressed as a Singular Value Decomposition (SVD) problem as $\mathsf{A}\mathcal{L}=0$, with each row of A defined as: $\mathsf{A}_i=(\bar{\xi}^T{}_i,\xi^T{}_i)^T$. Decomposing A into three matrices as $\mathsf{A}=\mathsf{U}\Sigma\mathsf{V}^T$ such as U,V are orthogonal and Σ diagonal, the sought solution is the null space spanned in the last two columns of V. This null space can be parametrized as $\mathcal{L}=\mathcal{L}_0+\mathcal{L}_1\lambda$ and, imposing the Plücker identity, we can compute the intersection with the Klein Quadric obtaining two solutions (see Fig. 2.1b). One of the solutions is the axis of revolution of the axial system or an arbitrary line and the other is the sought 3D line.

2.3 Central-projection systems

Central-projection cameras are characterized by having a unique optical center. That means, every ray coming from the environment goes through the optical center before arriving at the image. Among omnidirectional systems, panoramas are the most used in computer vision.

The **equirectangular** panorama (Fig. 2.2) is a 360º-field-of-view image that captures the whole environment around the camera. This kind of image is useful to obtain a complete projection of the environment from only one shot. However, this representation presents heavy distortions in the upper and lower part of the image. That is because the equirectangular panorama is based on a spherical projection. Taking the center of the sphere as the optical center, we can define the ray that comes from the environment in spherical coordinates (θ,ϕ). This forward projection model is defined as:

$$(\theta,\varphi)=\left(\arctan\frac{y}{x},\arctan\frac{z}{\sqrt{x^2+y^2}}\right), \tag{2.1}$$


(a) Acquisition device

(b) Acquired image

Figure 2.2: Acquisition device and obtained equirectangular panoramic image.

where (x,y,z) are the Cartesian coordinates of a 3D point in the environment.

Moreover, since the image plane is an unfolded sphere, each pixel, (u,v), can be represented in the same spherical coordinates, giving a direct relationship between the image plane and the ray that comes from the environment. The conversion of spherical coordinates to pixels is:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} W\dfrac{\theta-\theta_{init}}{\theta_{end}-\theta_{init}} \\ H\dfrac{\varphi-\varphi_{init}}{\varphi_{end}-\varphi_{init}} \end{pmatrix} \tag{2.2}$$

where (u,v) are pixel coordinates, (W,H) is the image resolution and $((W_{init},W_{end}),(H_{init},H_{end}))$ are the initial and final angle in the horizontal and vertical direction respectively (i.e. typically, these angle limits are defined as $((-\pi,\pi),(-\pi/2,\pi/2))$ to cover the whole environment around the optical center).

To compute the back-projection model, we can obtain the spherical coordinates from the image pixel coordinates as:

$$\begin{pmatrix} \theta \\ \varphi \end{pmatrix} = \left(\frac{u}{W}(\theta_{end}-\theta_{init})+\theta_{init}\,\frac{v}{H}(\varphi_{end}-\varphi_{init})+\varphi_{init}\right), \tag{2.3}$$

and then the 3D projecting ray, $\mathbf{v}$, as:

$$\mathbf{v} = \begin{pmatrix} \cos\theta\cos\varphi \\ \sin\theta\cos\varphi \\ \sin\varphi \end{pmatrix} \tag{2.4}$$

Fisheye cameras (Fig. 2.3) are constructed as a conventional camera with a set of lenses that increases the information obtained from the environment, obtaining a wider field of view Schneider et al. (2009). The projection model for this camera system has been obtained from Bermudez-Cameo et al. (2015), where a unified model for revolution symmetry cameras is defined (See Fig. 2.4). This

(a) Acquisition device

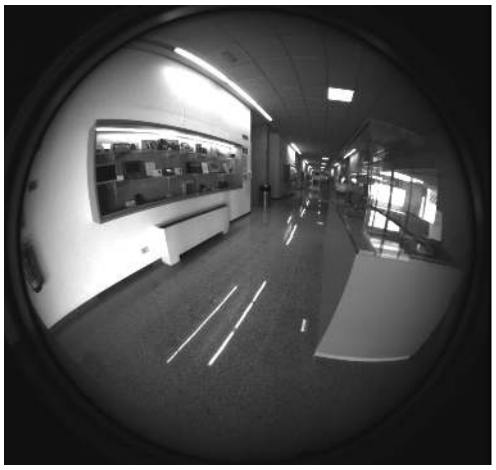

(b) Acquired image

Figure 2.3: Acquisition device and obtained fisheye image.

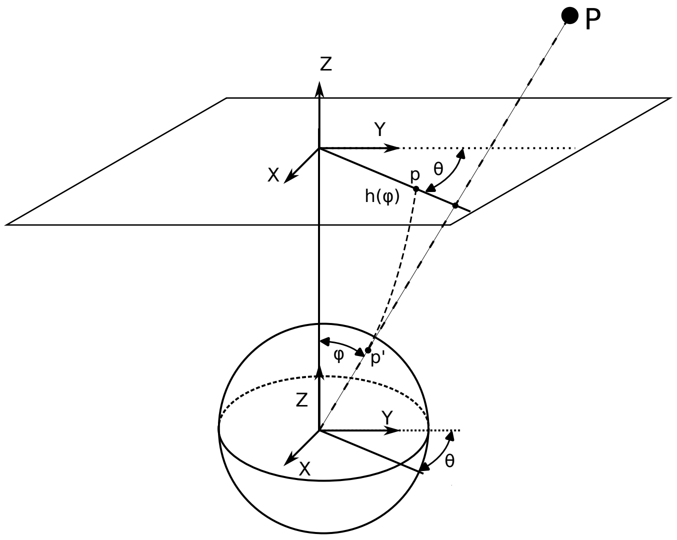


Figure 2.4: Unified model for revolution symmetry cameras from Bermudez-Cameo et al. (2015). From a 3D point **p**, the model defines a point $p = (u_p, v_p)^T$ in the image coordinate system.

method consists of the projection of the environment rays into a unit radius sphere. The intersection between the sphere and the ray is projected into the image plane through a non-linear function $\hat{r} = h(\varphi)$, which depends on the angle of the incoming ray and the modelled fisheye lens. Defining the incoming ray in spherical coordinates as in Eq. (2.1), we obtain the forward projection model, in polar coordinates, from Table 2.1, where $h(\varphi)$ is defined for the lenses implemented in this simulator.

Then, the relationship between the polar coordinates and pixel coordinates is computed as:

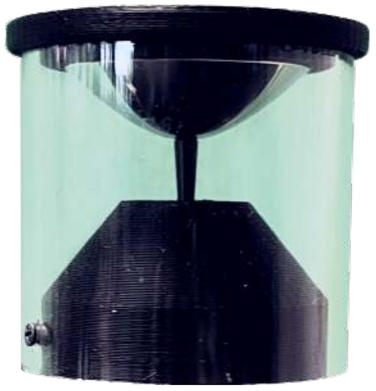
$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} + \hat{r} \cdot \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}, \tag{2.5}$$

Table 2.1: Definition of $h(\varphi)=\hat{r}$ for the fisheye forward projection model.

Equi-angular	Stereographic	Orthogonal	Equi-solid angle
$f\varphi$	$2f\tan(\varphi/2)$	$f\sin\varphi$	$f\sin(\varphi/2)$

Table 2.2: Definition of φ from the fisheye backward projection model.

Equi-angular	Stereographic	Orthogonal	Equi-Solid Angle
$\hat{r}/f$	$2\arctan(\hat{r}/2f)$	$\arcsin(\hat{r}/f)$	$2\arcsin(\hat{r}/f)$


(a) Acquisition device

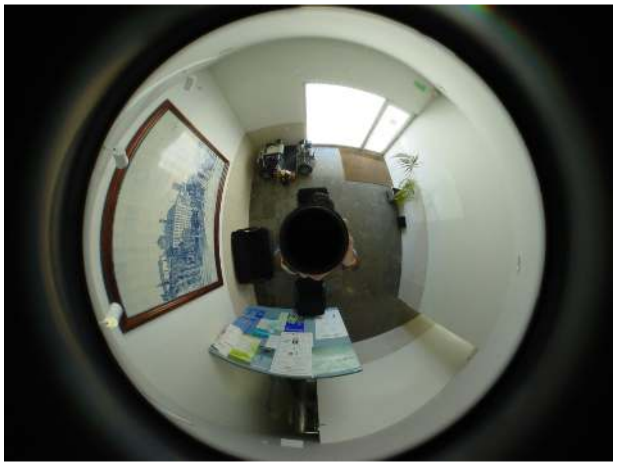

(b) Acquired image

Figure 2.5: Acquisition device and obtained catadioptric image.

where (u_0,v_0) are the pixel coordinates of the center point of the fisheye camera.

Due to the symmetry of revolution of this kind of cameras, we use polar coordinates. For the backward projection model, we first transform the pixel coordinates into polar coordinates as:

$$(\hat{r},\theta)=\left(\sqrt{(u-u_0)^2+(v-v_0)^2},\arctan((u_0-u)/(v-v_0))\right) \tag{2.6}$$

The relation of the polar, representing the pixels, and spherical coordinates, representing the 3D projecting ray, is the inverse of the forward projection model and is summarized in Table 2.2 where f is the focal length of the fisheye camera.

A different approach of omnidirectional cameras are the **catadioptric systems**, Fig 2.5. These systems are the combination of a conventional camera with specific mirrors creates, obtaining a field of view up to 360º around the camera. To model these systems, we use the sphere model presented in Baker and Nayar (1999) (See Fig. 2.6). As in fisheye cameras, we define the forward projection model as the intersection of the 3D ray with a unit radius sphere. Then, through a non-linear function, we project the intersected point into a normalized plane. The non-linear function, $h(x)$, depends on the

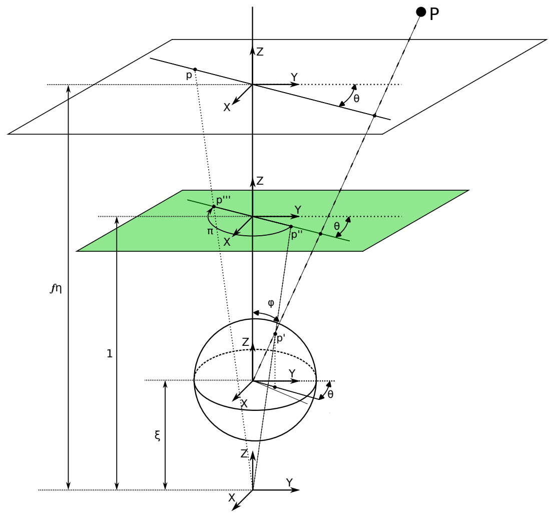


Figure 2.6: Sphere model for catadioptric systems from Baker and Nayar (1999). From a 3D point **p**, the model defines a point $p = (u_p, v_p)^T$ in the image coordinate system.

Table 2.3: Definition of ξ and η for central mirrors.

Catadioptric System	ξ	η
Para-catadioptric	1	-2p
Hyper-catadioptric	$\frac{d}{\sqrt{d^2+4p^2}}$	$\frac{-2p}{\sqrt{d^2+4p^2}}$

mirror we are modelling. The final step of this model projects the point in the normalized plane into the image plane with the calibration matrix H_c, defined as $\mathsf{H}_c = \mathsf{K}_c \mathsf{R}_c \mathsf{M}_c$, where K_c is the calibration matrix of the perspective camera, R_c is the rotation matrix of the catadioptric system and M_c defines the behaviour of the mirror (see Equation (2.7)).

$$\mathsf{K}_c = \begin{pmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{pmatrix}; \mathsf{M}_c = \begin{pmatrix} -\eta & 0 & 0 \\ 0 & \eta & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{2.7}$$

where (f_x, f_y) are the focal length of the camera, and (u_0, v_0) the coordinates of the optical center in the image, the parameters ξ and η represent the geometry of the mirror and are defined in Table 2.3, d is the distance between the camera and the mirror and $2p$ is the semi-latus rectum of the mirror.

The back-projection model for these catadioptric systems is described in Puig et al. (2012). First, we project the pixel coordinates into a 2D projective space: $p = (u, v)^T \rightarrow \hat{\mathbf{v}} = (\hat{x}, \hat{y}, \hat{z})^T$. Then, we re-project the point into the normalized plane with the inverse calibration matrix of the catadioptric system as $\bar{\mathbf{v}} = \mathsf{H}_c^{-1} \hat{\mathbf{v}}$. Finally, through the inverse of the non-linear function $h(x)$, shown in the equation (2.8), we can obtain the coordinates of the ray that goes from the catadioptric system to the environment.

$$\mathbf{v}=h^{-1}(\bar{\mathbf{v}})=\begin{pmatrix}x\\y\\z\end{pmatrix}=\begin{pmatrix}\bar{x}\frac{\bar{z}\xi+\sqrt{\bar{z}^2+(1-\xi^2)(\bar{x}^2+\bar{y}^2)}}{\bar{x}^2+\bar{y}^2+\bar{z}^2}\\ \bar{y}\frac{\bar{z}\xi+\sqrt{\bar{z}^2+(1-\xi^2)(\bar{x}^2+\bar{y}^2)}}{\bar{x}^2+\bar{y}^2+\bar{z}^2}\\ \bar{z}\frac{\bar{z}\xi+\sqrt{\bar{z}^2+(1-\xi^2)(\bar{x}^2+\bar{y}^2)}}{\bar{x}^2+\bar{y}^2+\bar{z}^2}-\xi\end{pmatrix} \tag{2.8}$$

Due to the imperfections in the lenses and mirrors produced by the manufacturing process, many commercial devices are calibrated through **empirical models** that take into account these imperfections in the projection model. In the literature we can find two models for revolution symmetry projection models that rise above the others: **Scaramuzza's** Scaramuzza et al. (2006a) and **Kannala–Brandt's** Kannala and Brandt (2006) projection models. These empiric models represent the projection of a 3D point into the image plane through non-lineal functions which parameters can be adjusted through calibration processes.

In the Kannala–Brandt's camera model Kannala and Brandt (2006), the forward projection of the reduced model (i.e. without taking into account small radial or tangential distortions) is represented as:

$$\begin{pmatrix}u\\v\end{pmatrix}=d(\theta)\cdot\begin{pmatrix}f_x\cos\varphi\\f_y\sin\varphi\end{pmatrix}+\begin{pmatrix}c_x\\c_y\end{pmatrix}, \tag{2.9}$$

where (θ,φ) are the spherical coordinates of the incoming ray, (f_x,f_y) are the pixels per unit distance in the horizontal and vertical directions, (c_x,c_y) are the pixel coordinates of the optical center and $d(\theta)$ is the non-linear function which models the camera distortion. This non-linear function is defined as $d(\theta)=k_0\theta+k_1\theta^3+k_2\theta^5+k_3\theta^7+k_4\theta^9$ where $[k_0,k_1,k_2,k_3,k_4]$ are the parameters that characterize the modelled camera.

The back-projection of the Kannala-Brandt's reduced model Kannala and Brandt (2006) can be obtained computing the real roots of the ninth order polynomial $d(\theta)$. Since this functions if monotonically increasing in the interval of the camera values, we can get a solution for any case. Thus, the back-projection model is defined as $\mathbf{X}=(\cos\varphi\cdot\sin\theta,\sin\varphi\cdot\sin\theta,\cos\theta)^T$, where:

$$\varphi=\arctan\left(\frac{v-c_y}{f_y}/\frac{u-c_x}{f_x}\right) \tag{2.10}$$

and $\theta=d^{-1}(r_0)$ where:

$$r_0=\sqrt{\frac{u-c_x}{f_x}^2+\frac{v-c_y}{f_y}^2} \tag{2.11}$$

The Scaramuzza's model Scaramuzza et al. (2006a) is a back-projection model where a non-linear function is used to model the image distortion. Given the image coordinates $(u,v)^T$, we can obtain the direction of a 3D point as:

$$\mathsf{P}\mathcal{X} = \lambda \begin{pmatrix} u' \\ v' \\ w' \end{pmatrix} = \lambda g(\mathsf{A}\mathbf{u} + t) = \lambda \begin{pmatrix} u \\ v \\ f(u',v') \end{pmatrix}, \lambda > 0, \quad (2.12)$$

where P is the perspective projection matrix, $\mathcal{X}$ is a 3D point in the environment, $\mathbf{u'} = (u',v',w')^T$ the projecting ray of the 3D point from the sensor plane, A, t is an affine transformation between the sensor plane and image plane, $\mathbf{u'} = (u,v,f(u',v')^T$ the projection of the 3D point in the image plane and $f(u',v')$ is the non-linear function that models the camera distortion. The non-lineal function is defined by the sensor plane coordinates and a n-grade polynomial function:

$$f(u',v') = a_0 + a_1\rho + a_2\rho^2 + \cdots + a_N\rho^N, \quad (2.13)$$

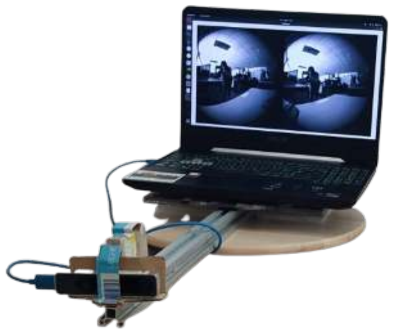
where ρ is defined as the distance of the pixel to the optical center in the image plane, and $[a_0,a_1,...,a_N]$ are calibration parameters of the modeled camera.

2.4 Non-central-projection cameras

Central projection systems are those acquisition systems where all projecting rays that form the image intersect at a single point, called optical center. The pinhole camera model or the spherical panorama are examples of central projection systems. By contrast, non-central projection systems do not have a unique optical center, that means, the rays that form the images do not pass through a unique point. This nature leads to a harder modelling of the projection and management of the information. Nevertheless, this characteristic allows to obtain more geometric information from the image than from central projection systems. In particular, we can extract the 3D information of lines directly from a single image.

The **non-central panorama** (Fig. 2.7) is a projection model, presented in Shum and Szeliski (1999), with symmetry of revolution in which each projecting ray intersects both an axis **n** and a circle of radius R_c (see Fig 2.8). This system can be modeled as the projection of the environment into a toroidal surface (Fig. 2.8). The optical center is distributed in a circle of radius R_c, centred in an axis **n** (dashed line of Fig. 2.8). For each optical center, we have a region where the projection is locally central, which correspond with one column in the panoramic image. We use Plücker coordinates Pottmann and Wallner (2001) to define the forward and backward projection function of the system as well as the 3D lines in the environment.

$$\theta = atan2(y,x); \ \varphi = atan\left(\frac{z}{\sqrt{x^2+y^2} - wR_c}\right) \quad (2.14)$$


(a) Acquisition device

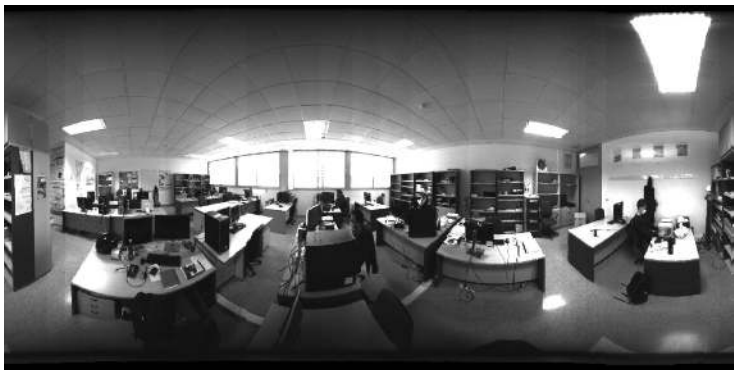

(b) Acquired image

Figure 2.7: Acquisition device and obtained non-cenral panoramic image.

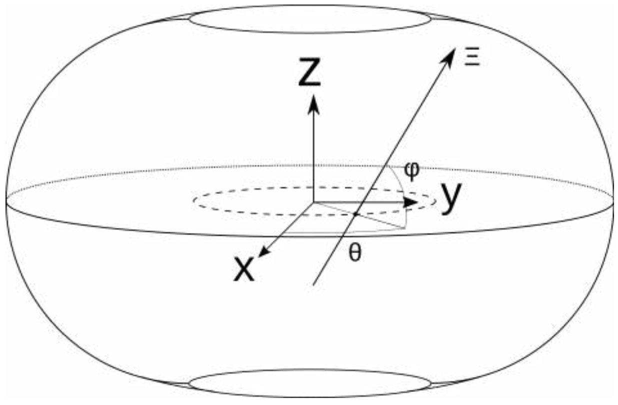


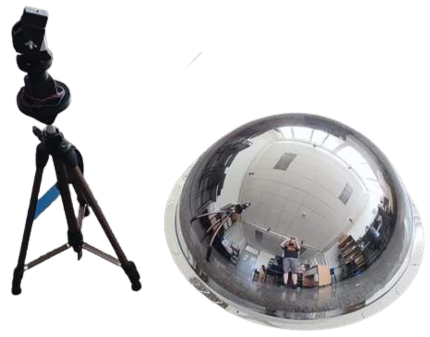
Figure 2.8: Non-central Panorama scheme.

$$j = n_{columns} \frac{\theta - \theta ini}{\theta end - \theta ini};\; i = m_{rows} \frac{\varphi - \varphi_{ini}}{\varphi_{end} - \varphi_{ini}} \tag{2.15}$$

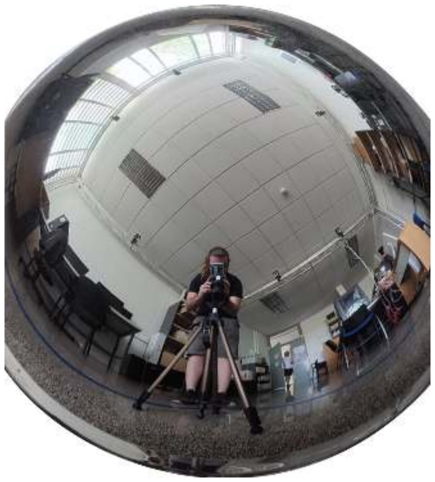
The forward projection provides the pixel coordinates (i,j) for each 3D point $(x,y,z,w)^T$ defined in homogeneous coordinates. Each point is defined by the spherical coordinates (φ,θ) from its corresponding optical center as defined in equation (2.14). The angles are transformed into pixel coordinates (2.15) taking into account the image resolution $(m_{rows}, n_{columns})$ and the horizontal $(\theta_{ini},\theta_{end})$ and vertical $(\varphi_{ini},\varphi_{end})$ fields of view.

$$\Xi = \begin{pmatrix} \xi & \bar{\xi} \end{pmatrix}^T = (\sin\varphi\cos\theta, \sin\varphi\sin\theta, \cos\varphi, R_c\sin\varphi\sin\theta, -R_c\sin\varphi\cos\theta, 0)^T \tag{2.16}$$

The backward projection model provides the projecting rays in Plücker coordinates from each pixel in the non-central panorama. The pixel coordinates are transformed back into spherical coordinates as in spherical central panoramas (see Eq. 2.3), taking into account the image resolution and field of views. Then, the projecting ray (2.16) is computed considering the radius R_c of the non-central acquisition system.



(a) Acquisition device



(b) Acquired image

Figure 2.9: Acquisition device and obtained non-central catadioptric image with spherical mirror.

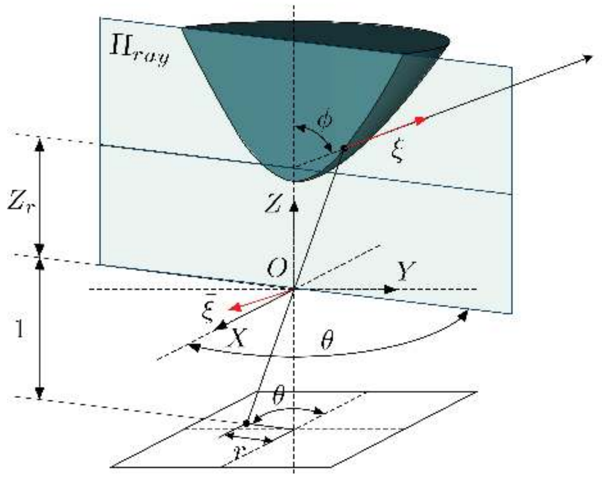


Figure 2.10: Non-central Catadioptric system scheme.

We also find **non-central catadioptric systems**, Fig. 2.9. Systems with spherical and conical mirrors are also described by non-central-projection models. Even though the basis of non-central and central catadioptric systems is the same, we take a picture of a mirror from a perspective camera, the mathematical projection model is quite different. As in non-central panoramas, for spherical and conical mirror we also use the Plücker coordinates to define projection rays; see Figure 2.10. For conical catadioptric systems, we define $Z_r = Z_c + R_c \cot\phi$, where Z_c and R_c are geometrical parameters and $\cot\phi = (z + r\tan 2\tau)/(z\tan 2\tau - r)$, where τ is the aperture angle of the cone. When these parameters are known, the 3D ray in Plücker coordinates is defined by:

$$\Xi = \begin{pmatrix} \xi & \bar{\xi} \end{pmatrix}^T = \begin{pmatrix} \sin\varphi\cos\theta, \sin\varphi\sin\theta, \cos\varphi, -Z_r\sin\varphi\sin\theta, Z_r\sin\varphi\cos\theta, 0 \end{pmatrix}^T \tag{2.17}$$

For the spherical catadioptric system, we define the geometric parameters as $Z_s = Z_m + R_s$, $Z_{rel} = Z_s/R_s$, $r^2 = x^2 + y^2$ and $\rho^2 = x^2 + y^2 + z^2$. Given the coordinates at a point of the image plane, the Equation (2.18) defines the ray that reflects on the mirror:

$$\Xi = \begin{pmatrix} \xi & \bar{\xi} \end{pmatrix}^T = \begin{pmatrix} -x\delta, y\delta, -\zeta, \varepsilon y Z_s, \varepsilon x Z_s, 0 \end{pmatrix}^T, \tag{2.18}$$

where

$$\begin{aligned}
&\delta = 2r^2 Z_{rel}^4 - 2z\sqrt{\gamma} Z_{rel}^2 - 3\rho^2 Z_{rel}^2 + \rho^2 \\
&\varepsilon = (-r^2 + z^2) Z_{rel}^2 + 2\sqrt{\gamma} z + \rho^2 \\
&\zeta = 2r^2 z Z_{rel}^4 - z\rho^2 Z_{rel}^2 - 2\sqrt{\gamma}(-r^2 Z_{rel}^2 + \rho^2) - z\rho^2 \\
&\gamma = (-r^2 Z_{rel}^2 + \rho^2) Z_{rel}^2.
\end{aligned}$$

Chapter 3

Simulator of omnidirectional cameras

Omnidirectional images are been more and more used in the computer vision and deep learning research communities. However, a high number of images is essential for the correct training of computer vision algorithms based on learning. In this chapter, we present a tool for generating datasets of omnidirectional images with semantic and depth information. These images are synthesized from a set of captures that are acquired in a realistic virtual environment for Unreal Engine 4. We gather a set of omnidirectional systems such as central and non-central panoramas, fisheye cameras and catadioptric systems. Besides, since the omnidirectional images are made virtually, we provide pixel-wise information about semantics and depth as well as perfect knowledge of the calibration parameters of the cameras. This allows the creation of ground-truth information with pixel precision for training learning algorithms and testing 3D vision approaches.

3.1 Introduction

The great amount of information that can be obtained from omnidirectional and 360º images makes them very useful. Being able to obtain information from an environment using only one shot makes these kinds of images a good asset for computer vision algorithms. However, due to the distortions they present, it is necessary to adapt or create special algorithms to work with them. New computer vision and deep-learning-based algorithms have appeared to take advantage of the unique properties of omnidirectional images. Nevertheless, for a proper training of deep-learning algorithms, big datasets are needed. Existing datasets are quite limited in size due to the manual acquisition, labelling and post-processing of the omnidirectional images. To make faster and bigger datasets, previous works such as Xiao et al. (2012); Song et al. (2015); Dai et al. (2017a); Armeni et al. (2017); Straub et al. (2019) use special equipment to obtain images, camera pose, and depth maps simultaneously from indoor scenes. These kinds of datasets are built from real environments, but need post-processing of the images to obtain semantic information or depth information. Tools like *LabelMe* Russell et al. (2008) and new neural networks such as *SegNet* Badrinarayanan et al. (2017) can be used to obtain automatic semantic segmentation from the real images obtained in the previously mentioned datasets, yet without pixel precision. Data-sets such as Zhang et al. (2010); Geiger et al. (2013) use video sequences from outdoor scenes to obtain depth information for autonomous driving algorithms. In addition, for these outdoor datasets, neural networks are used to obtain semantic information from video sequences Cordts et al. (2015, 2016), in order to speed up and enlarge the few datasets available.

Due to the fast development of graphic engines such as *Unreal Engine* EpicGames (2020)[1], virtual environments with realistic quality have appeared. To take advantage of this interesting property, simulators such as *SYNTHIA* Ros et al. (2016) and *CARLA* Dosovitskiy et al. (2017) recreate outdoor scenarios in different weather conditions to create synthetic datasets with labelled information. If we can define all the objects in the virtual environment, it is easier to create a semantic segmentation and object labelling, setting the camera pose through time and computing the depth for each pixel. These virtual realistic environments have helped to create large datasets of images and videos, mainly from outdoor scenarios, dedicated to autonomous driving. Other approaches use photo-realistic video games to generate the datasets. Since these games already have realistic environments designed by professionals, many different scenarios are recreated, with pseudo-realistic behaviours of vehicles and people in the scene. Works such as Doan et al. (2018) use the video game *Grand Theft Auto V (GTA V)* to obtain images from different weather conditions with total knowledge of the camera pose, while Richter et al. (2016, 2017) also obtaining semantic information and object detection for tracking applications. In the same vein, Johnson-Roberson et al. (2016); Angus et al. (2018) obtain video sequences with semantic and depth information for the generation of autonomous driving datasets in different weather conditions and through different scenarios, from rural roads to city streets. More recent approaches such as the *OmniScape* dataset Sekkat et al. (2020) uses virtual environments such as *CARLA* or *GTA V* to obtain omnidirectional images with semantic and depth information in order to create datasets for autonomous driving.

However, most of the existing datasets have only outdoors images. There are very few synthetic indoor datasets McCormac et al. (2017) and most of them only have perspective images or equirectangular panoramas. Fast development of computer vision algorithms demands ever more omnidirectional images and that is the gap between the resources that we want to fill. In this chapter, we present a tool to generate image datasets from a huge diversity of omnidirectional projection models. We focus not only on panoramas, but also on other central projections, such as fisheye lenses Kingslake (1989); Schneider et al. (2009), catadioptric systems Baker and Nayar (1999) and empiric models such as Scaramuzza's Scaramuzza et al. (2006a,b) and Kannala–Brandt's Kannala and Brandt (2006). Additionally, we include other omnidirectional systems such as non-central panoramas Menem and Pajdla (2004) or spherical Agrawal and Ramalingam (2013) and conical Lin and Bajcsy (2006) catadioptric systems.

[1] *Unreal Engine 4* was released at the beginning of my research career and just before the initial development of *OmniSCV*. At the day of writing this thesis, *Unreal Engine 5* has been released, but the developed algorithm hasn't been updated since this procedure introduces many technical issues, which are out of the current research line.

The composition of the images is made in a virtual environment from *Unreal Engine 4*, making camera calibration and image labelling easier. Moreover, we implement several tools to obtain ground-truth information for deep-learning applications, for example layout recovery or object segmentation.

This chapter is dedicated to:

- Integrate in a single framework several projection systems to obtain photo-realistic images from virtual environments generated with *Unreal Engine 4*.
- How is created *OmniSCV*, a framework that automatically generates labelled image data for several computer vision tasks, such as depth estimation and semantic segmentation, from several omnidirectional cameras.

3.2 OmniSCV framework

The objective of this section is explaining the development of a tool to create omnidirectional images enlarging existing datasets or making new ones to be exploited by computer vision algorithms under development. For this purpose, we use virtual environments, Unreal Engine 4 EpicGames (2020), from where we can get perspective images to compose 360º and several omnidirectional projections. In these environments, we can define the camera (pose, orientation and calibration), the layout, and the objects arranged in the scene, making it easier to obtain ground-truth information.

The proposed tool includes the acquisition of images from a virtual environment created with Unreal Engine 4 and the composition of omnidirectional images. Moreover, we can acquire photo-realistic images, semantic segmentation on the objects of the scene or depth information from each camera proposed. Furthermore, given that we can select the pose and orientation of the camera, we have enough information for 3D-reconstruction methods.

3.2.1 Virtual environments

Virtual environments present a new asset in the computer vision field. These environments allow the generation of customized scenes for specific purposes. The development of computer graphics has increased the quality and quantity of graphic software, obtaining even realistic renderings. A complex modelling of the light transport and its interaction with objects is essential to obtain realistic images. Virtual environments such as POV-Ray POV-Ray (2020) and Unity Unity (2020) allow the generation of customized virtual environments and obtain images from them. However, they do not have the realism or flexibility in the acquisition of images we are looking for. A comparative of images obtained from acquisitions in POV-Ray and acquisitions in Unreal Engine 4 is presented in Figure 3.1.

The virtual environment we have chosen is Unreal Engine 4[2], (UE4), which is a graphic engine developed by EpicGames (2020). Being an open-source code has allowed the development of a

[2]As mentioned before, this graphic engine was state of the art at the time we developed our tool. Many upgrades and a new and better graphic engine has appeared since, but our software has not been updated.

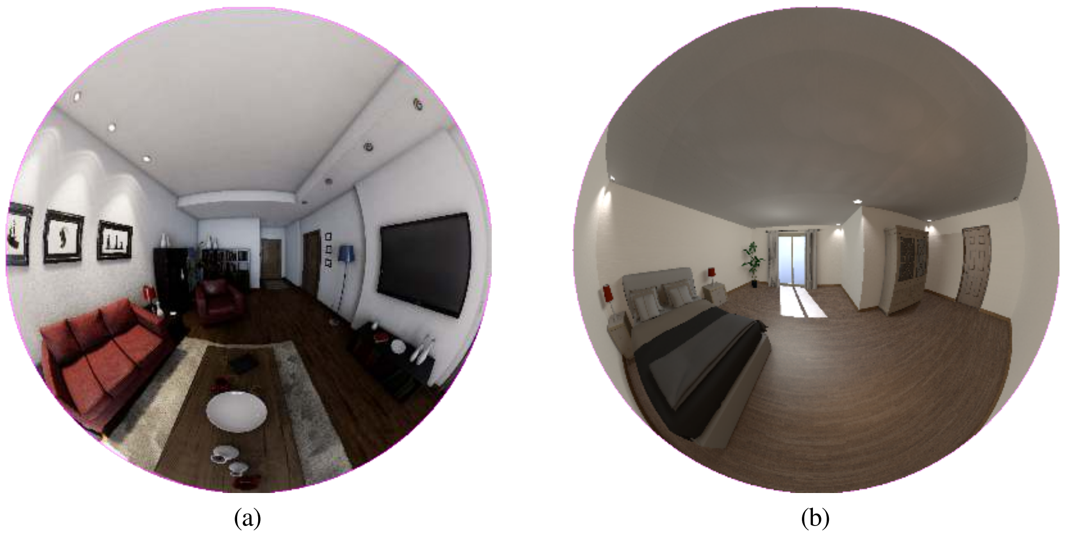


Figure 3.1: a): Kannala–Brandt projection obtained from Unreal Engine 4. b): Kannala–Brandt projection obtained from POV-Ray.

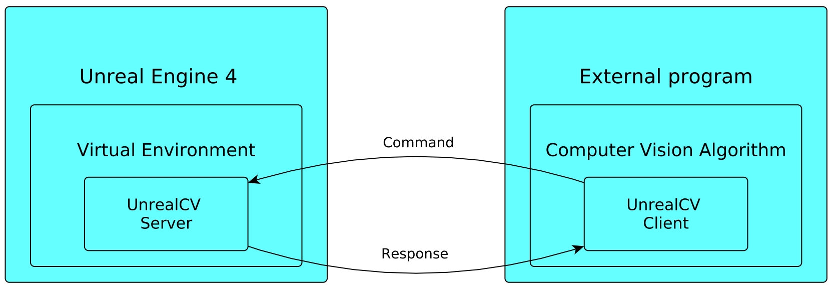


Figure 3.2: Client–server communication between Unreal Engine 4 and an external program via UnrealCV.

great variety of plugins for specific purposes. Furthermore, realistic graphics in real time allows the creation of simulations and generation of synthetic image datasets that can be used in computer vision algorithms. Working on virtual environments makes easier and faster the data acquisition than working on the field.

In our proposal for image generation, we use UE4 with UnrealCV Qiu et al. (2017), which is a plugin designed for computer vision purposes. This plugin allows client-server communication with UE4 from external Python scripts (see Figure 3.2) which is used to automatically obtain images. The set of available functions includes commands for defining and operating virtual cameras, i.e. set the position and orientation of the cameras and acquiring images. As can be seen in Figure 3.3, the acquisition obtains different kinds of information from the environment (i.e. RGB, semantic, depth or normals).

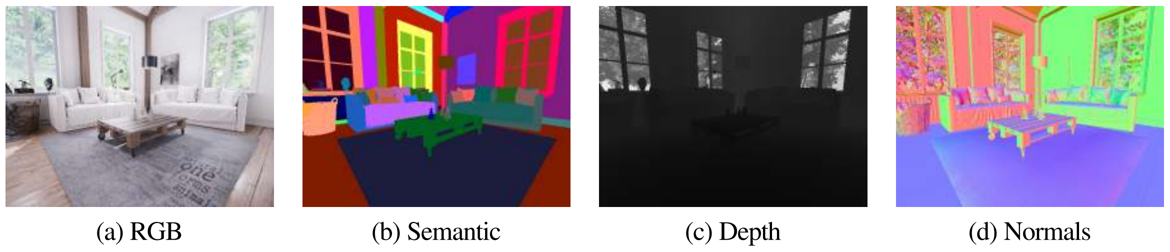


Figure 3.3: Capture options available in UnrealCV.

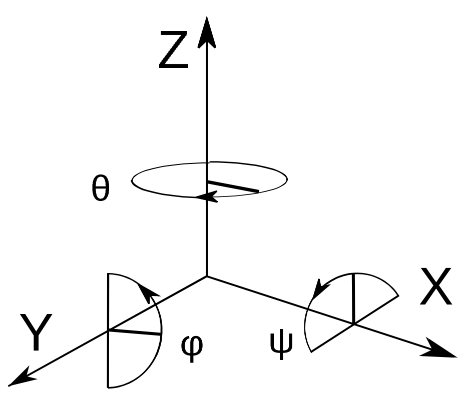
The combination of UE4+UnrealCV only allows perspective images, so it is necessary to find a way to obtain enough information of the environment to obtain omnidirectional images and in particular to build non-central images. For central omnidirectional images, the classical adopted solution is the creation of a cube map Greene (1986). This proposal consists of taking 6 perspective images, with a field of view of 90°, from one position so we can capture the whole environment around that point. We show that this solution only works for central projections, where we have a single optical center that matches with the point where the cube map has been taken. Due to the characteristics of non-central-projection systems, we have to make several acquisitions in different locations, which depend on the projection model, to compose the final image. The method to obtain omnidirectional images can be summarized in two steps:

- **Image acquisition**: the interaction with UE4 through UnrealCV to obtain the perspective images from the virtual environment.

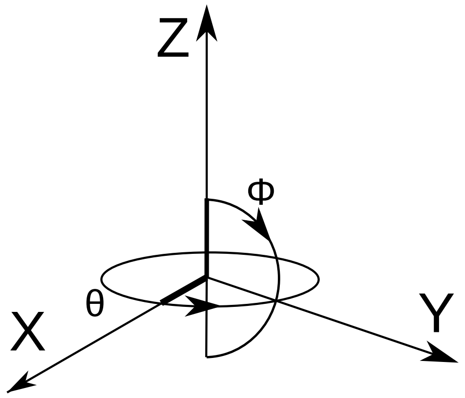
- **Image composition**: the creation of the final image. In this step we apply the projection models to select the acquired information of the environment.

For central-projection cameras, the two steps are independent from each other. Once we have a cube map, we can build any central-projection image from that cube map. However, for non-central-projection cameras, the two steps are mixed. We need to compute where the optical center is for each pixel (or cluster of pixels) and make the acquisition at that location.

One feature to take into account is the coordinate system of UE4, which is a left-handed coordinate system and the rotations are game-oriented (see Figure 3.4a). On the other hand, the projection models implemented in our simulator are defined as right-handed coordinate system and spherical coordinates, as shown in Figure 3.4b. The change of coordinate systems is computed internally in our tool to keep mathematical consistency between the projection models and the virtual environment.



(a) UE4 coordinate system



(b) OmniSCV coordinate system

Figure 3.4: (**a**): Coordinate system used in graphic engines focused on first-person video games; (**b**): Coordinate system of our image simulator.

3.2.2 Image acquisition

The image acquisition is the first step to build omnidirectional images. In this step we interact with UE4 through UnrealCV using Python scripts. Camera pose and orientation, camera field of view and mode of acquisition are the main parameters that we must define in the scripts to give the commands to UnrealCV.

We call cube map to the set of six perspective images that models the full 360º projection of the environment around a point; concept introduced in Greene (1986). Notice that 360º information around a point can be projected into a sphere centred in this point. Composing a sphere from perspective images requires a lot of time and memory. Simplifying the sphere into a cube, as seen in Figure 3.5, we have a good approximation of the environment without losing information. We can make this affirmation since the defined cube is a smooth atlas of the spherical manifold $\mathbb{S}^2$ embedded in $\mathbb{R}^3$ Lee and Lee (2012).

To create **central-projection systems**, the acquisition of each cube map must be done from a single location. Each cube map is the representation of the environment from one position, the optical center of the omnidirectional camera. That is why we use UE4 with UnrealCV, where we can define the camera pose easily. Moreover, the real-time renderings of the realistic environments allows fast acquisitions of the perspective images to build the cube maps. Also, other virtual environments can be used to create central-projection systems whenever the cube map can be built with these specifications.

On the other hand, **non-central projection systems** do not have a unique optical center, so the cube acquisition is not fit as a solution. Besides, since each pixel of a non-central system can have a different optical center, acquiring a cube map for each pixel is also a bad idea, since significantly increases the memory and computation time required. Looking for an efficient implementation, we group the pixels sharing the same optical center, reducing the number of acquisitions needed to build it. Then, instead of storing all the acquisitions, the image composition is made simultaneously, taking only

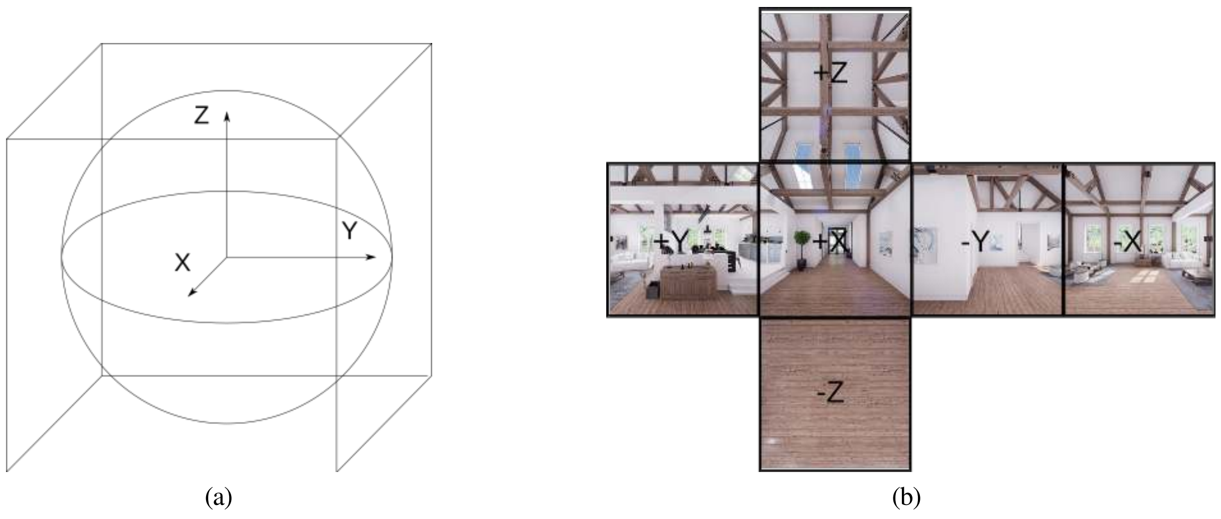


Figure 3.5: (**a**): Simplification of the sphere into the cube map; (**b**): Unfolded cube map from a scene.

the relevant information needed for each optical center. The non-central systems considered group the pixel in different ways, so the solutions are slightly different.

From the projection model of non-central panoramas, we get that pixels with the same *u* coordinate share the same optical center. For each coordinate *u* of the image, the position for the image acquisition is computed. For a given center $(X_c,Y_c,Z_c)^T$ and radius R_c of the non-central system, we have to compute the optical center of each *u* coordinate.

$$(x_{oc},y_{oc},z_{oc})^T = (X_c,Y_c,Z_c)^T + R_c(\cos\theta\cos\phi, \sin\theta, \cos\theta\sin\phi)^T \tag{3.1}$$

$$\theta = \left(\frac{2u}{u_{max}} - 1\right)\pi \tag{3.2}$$

To obtain each optical center, we use equation (3.1), where θ is computed according to equation (3.2) and ϕ is the pitch angle of the non-central system. Once we have obtained the optical center, we make the acquisition in that location, obtaining the faces of the cube that see ”out” of the non-central system (i.e. the sides facing the axis of revolution or which are crossed by the circle of optical centres are not acquired, reducing the computation time).

In non-central catadioptric systems, pixels sharing the same optical center are grouped in concentric circles. That means, we go through the final image from the center to the edge, increasing the radius pixel by pixel. For each radius, we compute the parameters Z_r and $\cot\phi$ as in Table 3.1, which depend on the mirror we want to model (see section 2.1). The parameter Z_r allows us to compute the optical center from where we acquire a cube map of the environment. In this case, we acquire the whole cube map since we do not know exactly where is the limit of the mirror, so the environment behind the mirror can also be acquired.

$$(u,v) = (u_{max}/2 + r\cos\theta, v_{max}/2 - r\sin\theta) \tag{3.3}$$

Table 3.1: Definition of $\cot\phi$ and Z_r for different mirrors.

Mirror	Z_r	$\cot\phi$
Conical	$Z_c + R_c\cot\phi$	$(1+r\tan(2\tau))/(\tan(2\tau)-r)$
Spherical	$-\xi/\delta$	$Z_s(\delta+\varepsilon)/\delta$

Going back to UnrealCV, the plugin gives us different kinds of capture modes. For our simulator, we have taken 3 of these modes: *lit* (a.k.a. RGB image) *object mask* (a.k.a. semantic segmentation) and *depth*. In the *lit* mode, UnrealCV gives a photorealistic RGB image of the virtual environment. The degree of realism must be created by the designer of the scenario. The second is the *object mask* mode. This mode gives us semantic information of the environment. The images obtained have a coloured code to identify the different objects into the scene. The main advantage of this mode is the pixel precision for the semantic information, avoiding the human error in manual labelling. Moreover, from this capture mode, we can obtain ground-truth information of the scene and create specific functions to obtain ground-truth data for computer vision algorithms, as layout recovery or object detection. The third mode is *depth*. This mode gives a data file where we have depth information for each pixel of the image. This data is defined as the Euclidean distance between the optical center and the 3D point.

3.2.3 Image composition

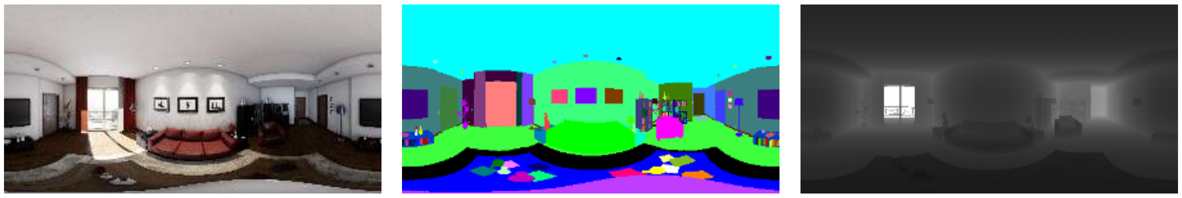
Images from **central-projection cameras** are composed from the cube map acquired in the scene. The composition is made offline and the image mapping depends on the projection model of the camera, but they follow the same pattern. Initially, we get the pixel coordinates from the image that we want to build. We compute the spherical coordinates for each pixel through the projection model of the camera and compute the projecting rays that goes from the optical center of the camera to the environment. Then, we rotate this rays to match the orientation of the camera in the environment. The intersection with the cube map gives the information of the pixel (color, semantic or depth).

The composition of images from **non-central projection cameras** is made online, that means, at the same time of the acquisition. Following the same principle as in central systems, we compute the spherical coordinates to cast the ray from the current optical center into the acquired images. In this process, we select the pixels that cast the projecting rays according to the optical center in which the acquisition has been made. Once we have the projecting rays, the intersection with the acquired images gives the information for the pixels (color, semantic or depth). This process of acquisition-composition continues until all the pixels of the image have their corresponding information.

3.2.4 Let's watch distortion: omnidirectional images

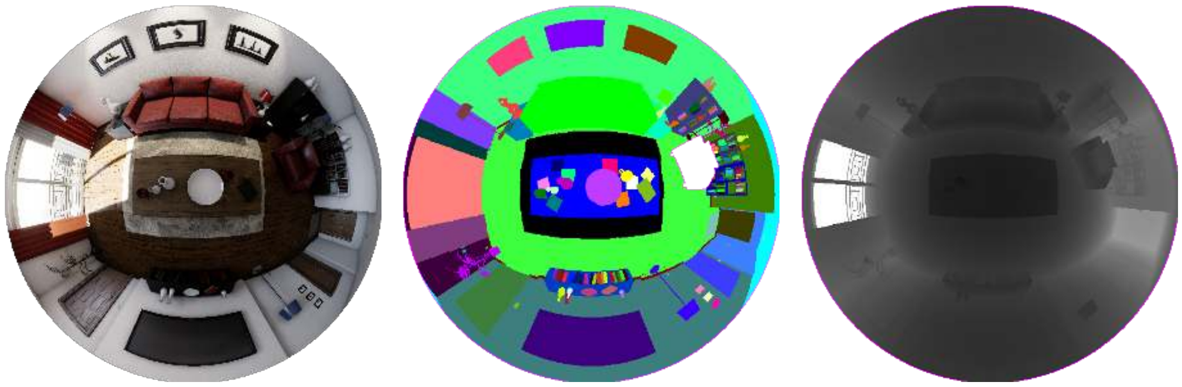
To this point, we have talked a lot about omnidirectional cameras, projection models, virtual environments... but no images have been shown. As a short break before the experimentation, here we show an example of color images taken from real cameras and with our framework OmniSCV. The reader will be able to see the difference between the images and also how these cameras look like in the real world.

Starting with central projection systems, we can find several companies that sell panoramic cameras and fisheye lenses. Besides, it is fairly easy to build catadioptric systems. On the other hand, non-central projection systems are harder to find. Catadioptric systems can be easily build as in the central case, but we cannot find a panoramic device to generate non-central panoramas. For that purpose, we have created a homemade device that allows us to obtain this kind of images.



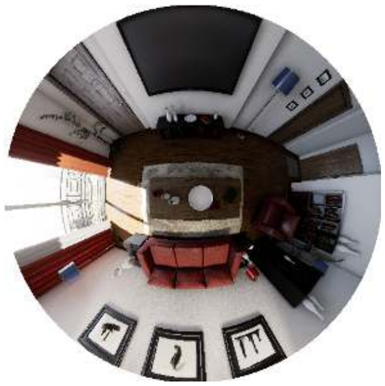
(a) Color image (b) Semantic map (c) Depth map

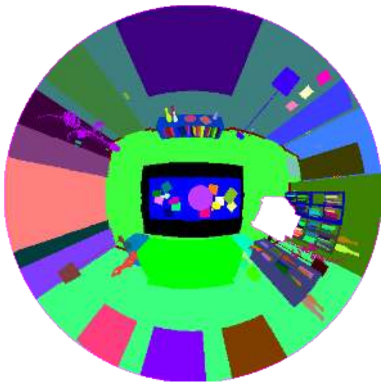
Figure 3.6: Equirectangular panoramic image example obtained with OmniSCV.



(a) Color image (b) Semantic map (c) Depth map

Figure 3.7: Fisheye image example obtained with OmniSCV under the equi-angular projection model.


(a) Color image


(b) Semantic map

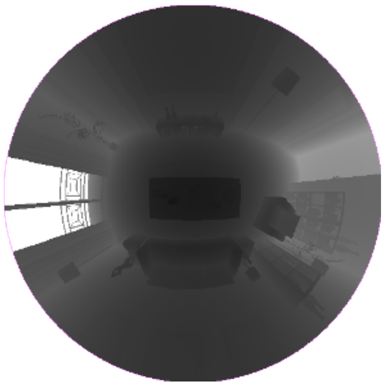
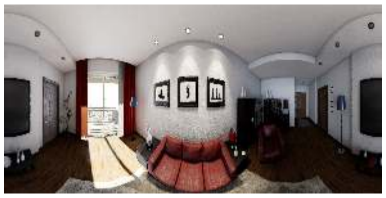
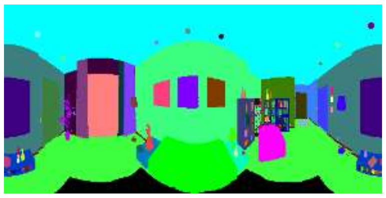

(c) Depth map

Figure 3.8: Catadioptric image example obtained with OmniSCV with an hyperbolic mirror.


(a) Color image


(b) Semantic map

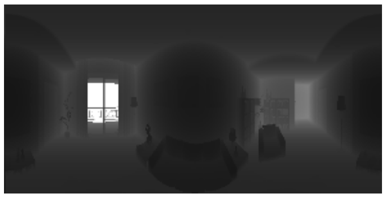
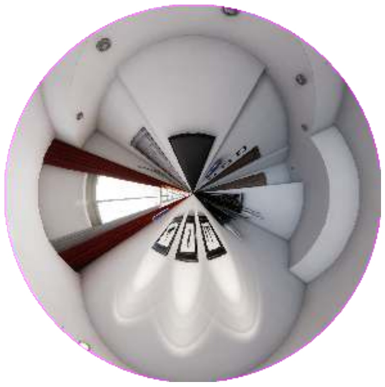
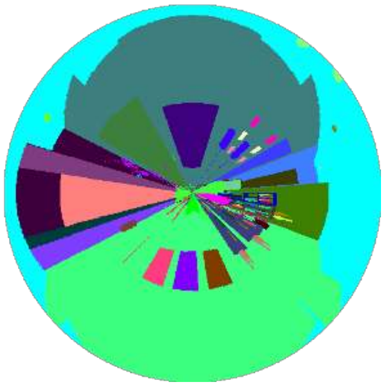

(c) Depth map

Figure 3.9: Non central panoramic image example obtained with OmniSCV.


(a) Color image


(b) Semantic map

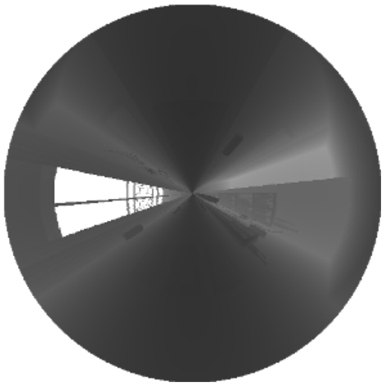

(c) Depth map

Figure 3.10: Non-central catadioptric image example obtained with OmniSCV with a conic mirror.

3.3 Examples and applications of OmniSCV

Working on environments of Unreal Engine 4 EpicGames (2020) and with our simulator, we have obtained a variety of photorealistic omnidirectional images from different systems. To evaluate if our synthetic images can be used in computer vision algorithms, we compare the evaluation of three algorithms with the synthetic images and real ones. The algorithms selected are:

- **Corners For Layouts**: CFL Fernandez-Labrador et al. (2020) is a neural network that recovers the 3D layout of a room from an equirectangular panorama. We have used a pre-trained network to evaluate our images.
- **OpenVSLAM**: a virtual Simultaneous Location and Mapping framework, presented in Sumikura et al. (2019), which allows use of omnidirectional central image sequences.
- **Uncalibtoolbox**: the algorithm presented in Bermudez-Cameo et al. (2015) is a toolbox for line extraction and camera calibration for different fisheye and central catadioptric systems. We compare the calibration results from different images.
- **3D line Reconstruction from single non-central image** which was presented in Bermudez-Cameo et al. (2017, 2016) using non-central panoramas.

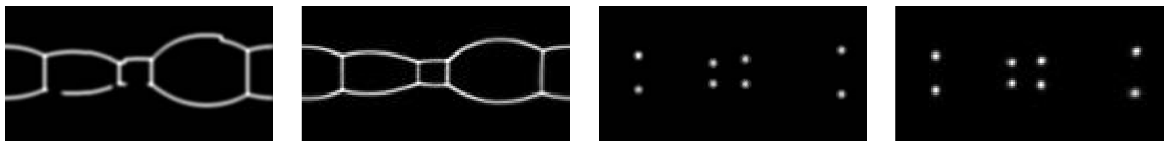
3.3.1 3D layout recovery: CFL

Corners For layouts (CFL) is a neural network that recovers the layout of a room from one equirectangular panorama. This neural network provides two outputs: one is the intersection of walls (or edges) in the room and the second is the corners of the room. With those representations, we can build a 3D reconstruction of the layout of the room using Manhattan world assumptions.

For our evaluation, we have used a pre-trained CFL[3] network with Equirectangular Convolutions (*Equiconv*). The training dataset was composed by equirectangular panoramas built from real images and the ground truth was made manually, which increases the probability of mistakes due to human error. The rooms that compose this dataset are 4-wall rooms (96%) and 6-wall rooms (6%).

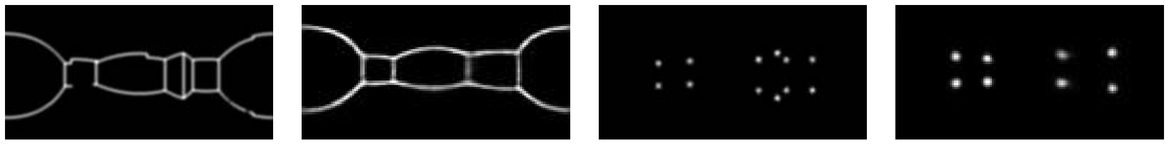
To compare the performance of CFL, the test images are divided into real images and synthetic images. In the set of real images, we have used the test images from the datasets Stanford2D3D Armeni et al. (2017), composed of 113 equirectangular panoramas of 4-wall rooms, and the SUN360 Xiao et al. (2012), composed of 69 equirectangular panoramas of 4-wall rooms and 3 panoramas of 6-wall rooms. The set of synthetic images are divided in panoramas from 4-wall rooms and from 6-walls rooms. Both sets are composed by 10 images taken on 5 different places in the environment in 2 orientations. Moreover, for the synthetic sets, the ground-truth information of the layout has been obtained automatically with pixel precision.

[3] https://github.com/cfernandezlab/CFL



(a) (b) (c) (d)

Figure 3.11: Results of CFL using a synthetic image from a 4-wall environment. (**a**): Edges ground truth; (**b**): Edges output from CFL; (**c**): Corners ground truth; (**d**): Corners output from CFL.



(a) (b) (c) (d)

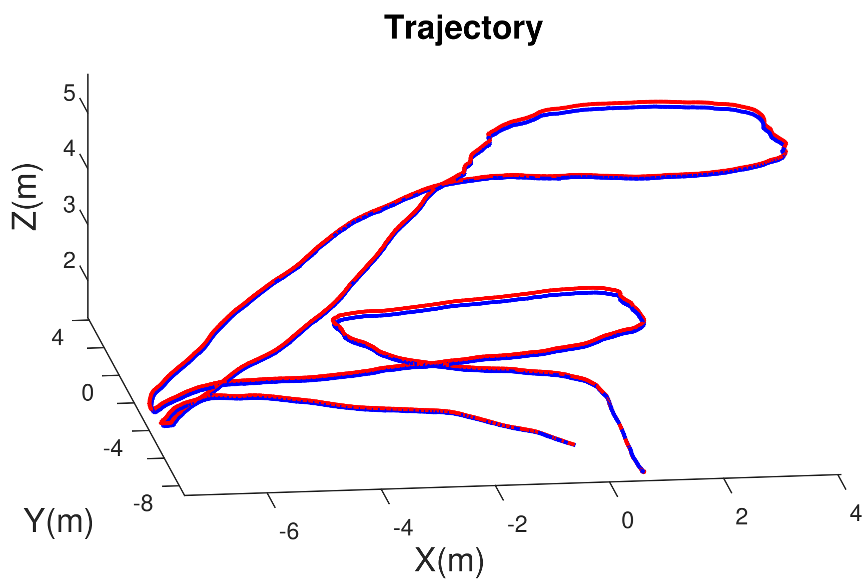
Figure 3.12: Results of CFL using a synthetic image from a 6-wall environment. (**a**): Edges ground truth; (**b**): Edges output from CFL; (**c**): Corners ground truth; (**d**): Corners output from CFL.

Table 3.2: Comparative of results of images from different datasets. *OmniSCV* contains the images created with our tool on a 6-wall room and a 4-wall room. The real images have been obtained from the test dataset of Armeni et al. (2017) and Song et al. (2015).

Test images	Edges					Corners				
	IoU	Acc	P	R	F_1	IoU	Acc	P	R	F_1
6-wall	0.424	0.881	0.694	0.519	0.592	0.370	0.974	0.547	0.526	0.534
4-wall	0.474	0.901	0.766	0.554	0.642	0.506	0.985	0.643	0.698	0.669
Real images	0.452	0.894	0.633	0.588	0.607	0.382	0.967	0.758	0.425	0.544

The goal of these experiments is testing the performance of the neural network in the different situations and evaluate the results using our synthetic images and comparing with those obtained with the real ones. In the figures above, the ground-truth generation can be seen in Figures 3.11a 3.11c 3.12a 3.12c, and the output of CFL in Figures 3.11b 3.11d 3.12b 3.12d, for an equirectangular panorama in the virtual environments recreated. On the 4-wall layout environment we can observe that the output of CFL is similar to the ground truth. This seems logical since most of the images from the training dataset have the same layout. On the other hand, the 6-wall layout environment presents poorer results. The output from CFL in this environment only fits four walls of the layout, probably due to the gaps in the training data.

To quantitatively compare the results of CFL, in table 3.2 we present the results using real images from existing datasets with the results using our synthetic images. We compare five standard metrics: Intersection over union of predicted corner/edge pixels (IoU), accuracy (Acc), precision (P), Recall (R) and F1 Score F_1.



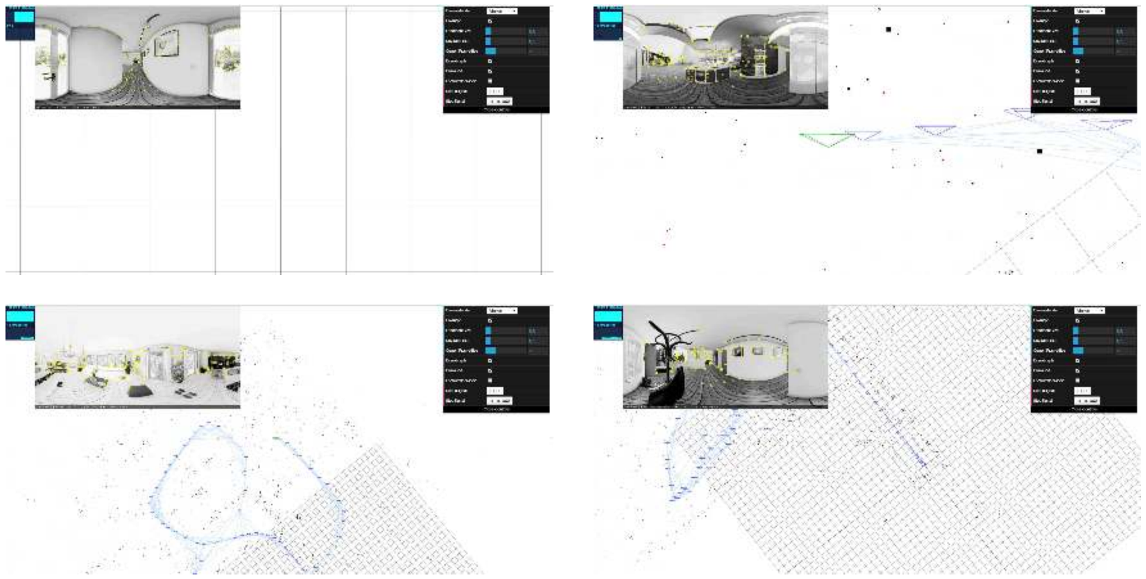
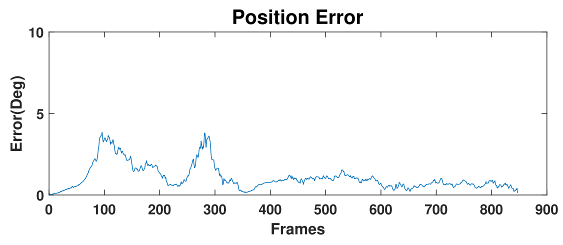


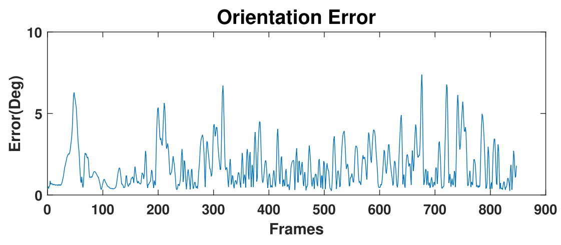
Figure 3.13: Visual odometry from SLAM algorithm. The red line is the ground-truth trajectory while the blue line is the scaled trajectory of the SLAM algorithm. Below we present several frames of the sequence with the OpenVSLAM user interface.

3.3.2 OpenVSLAM

The algorithm presented in Sumikura et al. (2019) allows the obtaining of a SLAM for different cameras, from perspective to omnidirectional central systems. This open-source algorithm is based on an indirect SLAM algorithm, such as ORB-SLAM Mur-Artal et al. (2015) and ProSLAM Schlegel et al. (2018). The main difference with other SLAM approaches is that the proposed framework allows definition of various types of central cameras other than perspective, such as fisheye or equirectangular cameras.



(a)



(b)

Figure 3.14: (**a**): Position error of the SLAM reconstruction. (**b**): Orientation error of the SLAM reconstruction. Both errors are measured in degrees.

To evaluate the synthetic images generated with OmniSCV, we create a sequence in a virtual environment simulating the flight of a drone. Once the trajectory is defined, we generate the images where we have the ground truth of the pose of the drone camera.

The evaluation has been made with equirectangular panoramas of 1920×960 pixels through a sequence of 28 seconds and 30 frames per second. The ground-truth trajectory as well as the scaled SLAM trajectory and several frames of the sequence[4] can be seen in Figure 3.13.

We evaluate quantitatively the precision of the SLAM algorithm computing the position and orientation error of each frame respect to the ground-truth trajectory. We consider error in rotation, ε_θ, and in translation, ε_t, as follows:

$$\varepsilon_t = \arccos\left(\frac{t_{gt}^T \cdot t_{est}}{|t_{gt}||t_{est}|}\right); \varepsilon_\theta = \arccos\left(\frac{Tr\left(R_{gt}{R_{est}}^T\right) - 1}{2}\right), \tag{3.4}$$

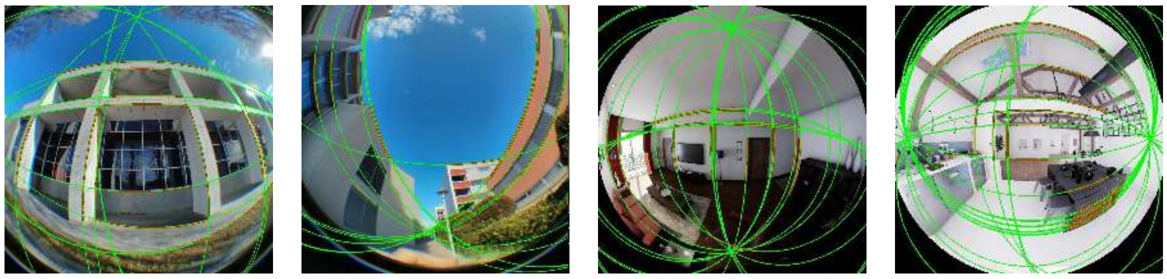
where t_{gt}, R_{gt} are the position and rotation matrix of a frame in the ground-truth trajectory and t_{est}, R_{est} are the estimated position up to scale and rotation matrix of the SLAM algorithm in the same frame. The results of these errors are shown in Figure 3.14

3.3.3 Uncalibtoolbox

The uncalibtoolbox is a toolbox where we can use a line extractor and compute the calibration on fisheye lenses and catadioptric systems. This toolbox makes the line extraction from the image and computes the calibration parameters from the distortion on these lines. The more distortion the lines present in the image, the more accurate the calibration parameters are computed. Presented in Bermudez-Cameo et al. (2015), this toolbox considers the projection models to obtain the main calibration parameter $\hat{r}_{vl}$ of the projection system. This $\hat{r}_{vl}$ parameter encapsulates the distortion of each projection and is related with the field of view of the camera.

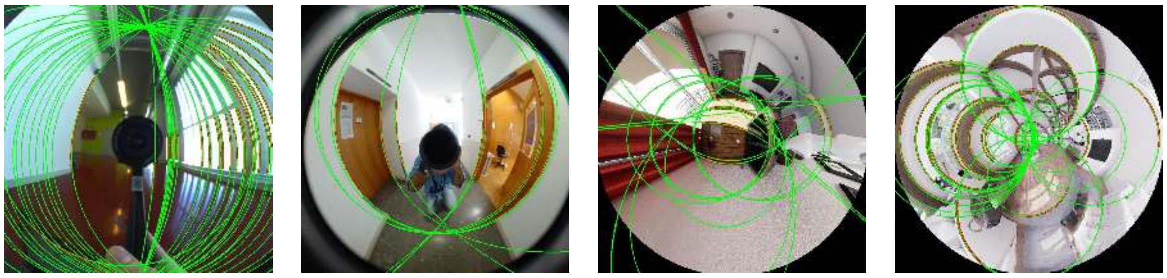
On this evaluation we want to know if our synthetic images can be processed as real images on computer vision algorithms. We take several dioptric and catadioptric images generated by our

[4]The full sequence can be found in `https://github.com/Sbrunoberenguel/OmniSCV`.



(a) (b) (c) (d)

Figure 3.15: Line extraction on fisheye camera. (**a,b**): Real fisheye images from examples of Bermudez-Cameo et al. (2015); (**c,d**): Synthetic images from our simulator.



(a) Hyper-catadioptric (b) Para-catadioptric (c) Hyper-catadioptric (d) Para-catadioptric

Figure 3.16: Line extraction on catadioptric systems. (**a,b**): Real catadioptric images from examples of Bermudez-Cameo et al. (2015); (**c,d**): Synthetic images from our simulator.

simulator and perform the line extraction on them. To compare the results of the line extraction, we compare with real images from Bermudez-Cameo et al. (2015).

In the previous figures, we presented four equi-angular fisheye, Figure 3.15, and two catadioptric systems, Figure 3.16. The behaviour of the line extraction algorithm of uncalibtoolbox for the synthetic images of our simulator is similar to the real images presented in Bermudez-Cameo et al. (2015). That encourages our work, because we have made photorealistic images that can be used as real ones for computer vision algorithms.

On the other hand, we compare the accuracy of the calibration process between the results presented in Bermudez-Cameo et al. (2015) and the obtained with our synthetic images. The calibration parameter has been obtained testing 5 images for each r_{vl} and taking the mean value. Since we impose the calibration parameters in our simulator, we have selected 10 values of the parameter r_{vl}, in the range $0.5 < r_{vl} < 1.5$, in order to compare our images with the results of Bermudez-Cameo et al. (2015). The calibration results are presented in the Figure 3.17.

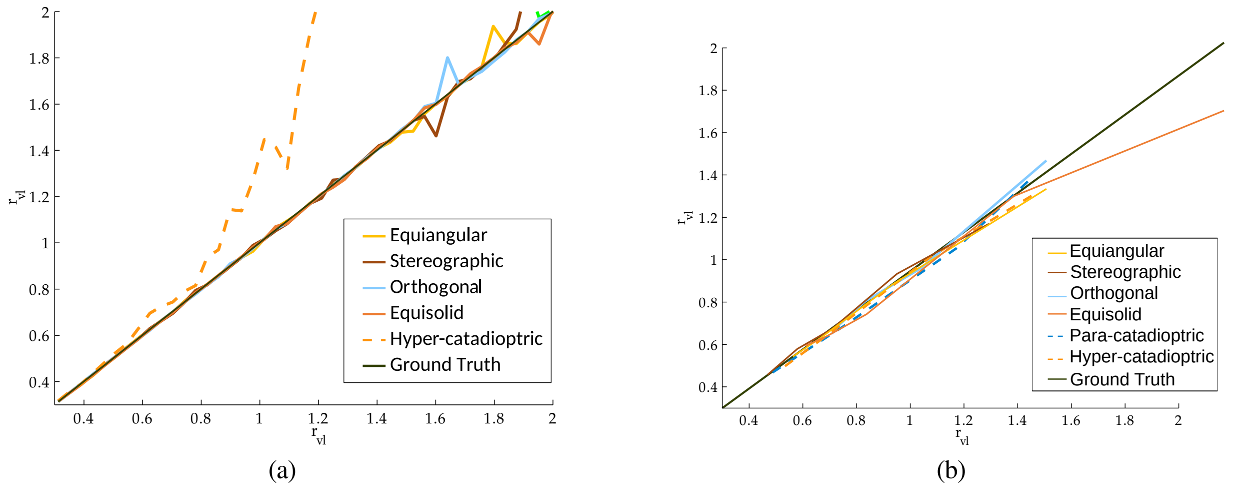


Figure 3.17: Normalized result for the calibration parameters using different omnidirectional cameras. (**a**): Calibration results from Bermudez-Cameo et al. (2015); (**b**): Calibration results using images from our simulator.

3.3.4 3D line reconstruction from single non-central image

One of the particularities of non-central images is that line projections contain more geometric information. In particular, the entire 3D information of the line is mapped on the line projection Teller and Hohmeyer (1999); Gasparini and Caglioti (2011).

To evaluate if our synthetic non-central images generated by our tool conserve this property, we test the proposal presented in Bermudez-Cameo et al. (2016). This approach assumes that the direction of the gravity is known (this information could be recovered from an inertial measurement unit (IMU)) and lines are arranged in vertical and horizontal lines. Horizontal lines can follow any direction contained in any horizontal plane (soft-Manhattan constraint). The non-central camera captures a non-central panorama of 2048×1024 pixels with a radius of $R_c = 1m$ and an inclination of 10 degrees from the vertical direction.

A non-central-depth synthetic image has been used as ground truth of the reconstructed points (see Figure 3.18b). In Figure 3.18a we show the extracted line projections and segments on the non-central panoramic image; meanwhile, Figure 3.19 presents the reconstructed 3D line segments. We have omitted the segments with low effective baseline in the 3D representation for visualization purposes.

3.4 Discussion and conclusions

In the evaluation with CFL we have obtained results comparable to datasets with real images. This behaviour shows that the synthetic images generated with our tool work as well as with real images from the existing datasets. On the other hand, we have made different tests changing the layout of our scene, something that cannot be done in real scenarios. After these changes, we realized that CFL

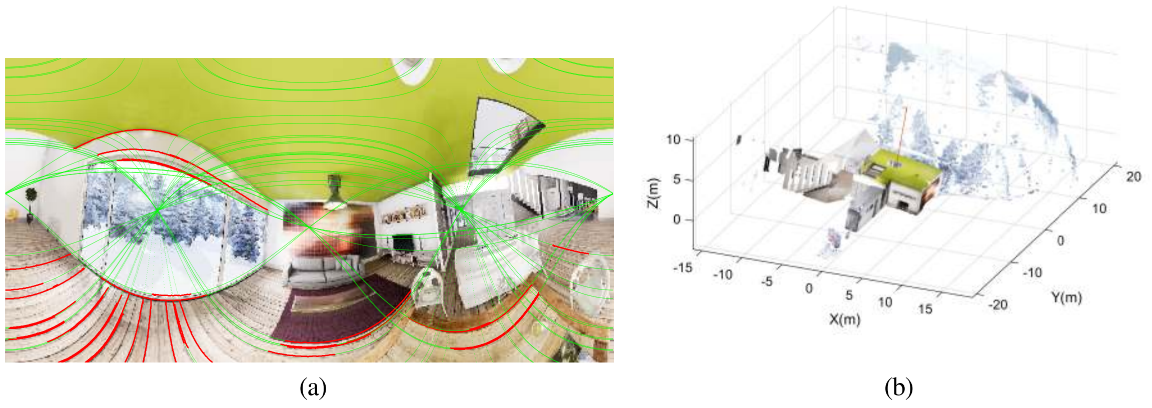


Figure 3.18: (**a**) Extracted line projections and segments on the non-central panorama. (**b**) Ground-truth point-cloud obtained from depth-map.

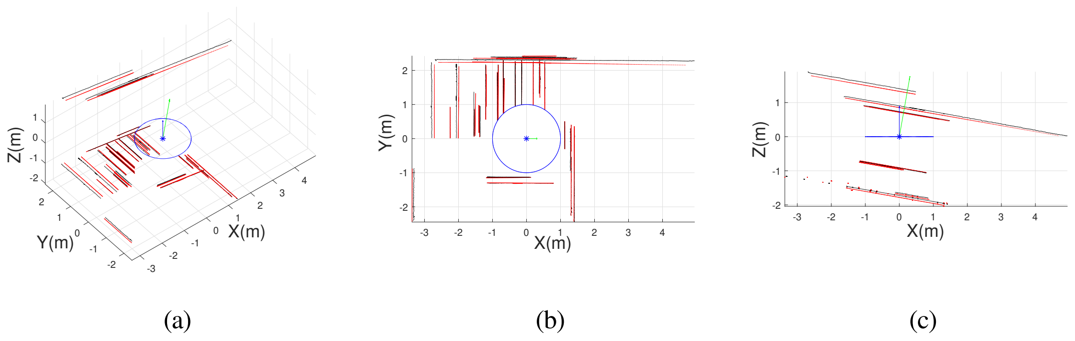


Figure 3.19: 3D line segments reconstructed from line extraction in non-central panorama. In red the reconstructed 3D line segments. In black the ground truth. In blue the circular location of the optical center and the Z axis. In green the axis of the vertical direction. (**a**) Orthographic view. (**b**) Top view. (**c**) Lateral view.

does not work properly with some layouts. This happens because existing datasets have mainly 4-wall rooms to use as training data and the panoramas have been taken in the middle of the room Armeni et al. (2017); Song et al. (2015). This makes it hard for the neural network to generalize for rooms with more than 4 walls or with panoramas that have been taken in different places inside the room. Our tool can aid in solving this training problem. Since we can obtain images from every place in the room and we can change the layout, we can fill the gaps of the training dataset. With bigger and richer datasets for training, neural networks can improve their performance and make better generalizations.

In the evaluation of the SLAM algorithm, we test if the synthetic images generated with our tool can be used in computer vision algorithms for tracking and mapping. If we compare the results obtained from the OpenVSLAM algorithm Sumikura et al. (2019), with the ground-truth information that provides our tool, we can conclude that the synthetic images generated with OmniSCV can be used for SLAM applications. The position error is computed in degrees due to the lack of scale in the SLAM algorithm. Moreover, we observe the little position and orientation error of the camera along the sequence (see Figure 3.14), keeping the estimated trajectory close to the real one. Both errors are less than 8 degrees and decrease along the trajectory. This correction of the position is the effect of the loop closure of the SLAM algorithm. On the other hand, we obtain ground-truth information of the camera pose for every frame. This behaviour encourages the assumptions we have been referring to in this section: that synthetic images generated from our tool can be used as real ones in computer vision algorithms, obtaining more accurate ground-truth information too.

In the evaluation with *uncalibtoolbox*, we have tested catadioptric systems and fisheye lenses. We have compared the precision of the toolbox for real and synthetic images. In the line extraction, the toolbox has no problems nor makes any distinction from one kind of images or the other. That encourages our assumptions that our synthetic images are photorealistic enough to be used as real images. When we compare the calibration results, we can see that the results of the images obtained from Bermudez-Cameo et al. (2015) and the results from our synthetic images are comparable. There are no big differences in precision. The only different values observed are in hyper-catadioptric systems. For the hyper-catadioptric systems presented in Bermudez-Cameo et al. (2015), the computed calibration parameters differ from the real ones while in the synthetic hyper-catadioptric systems, we have more accurate parameters. A possible conclusion of this effect is the absence of the reflection of the camera in the synthetic images. For those, we have more information of the environment in the synthetic images than in real ones, helping the toolbox to obtain better results for the calibration parameters. From the results shown, we can conclude that our tool can help to develop and test future calibration tools. Since we are the ones that set the calibration of the system with the simulator, we have perfect knowledge of the calibration parameters of the image. However, real systems need to be calibrated a priori or we must trust the calibration parameters that the supplier of the system gives us.

Finally, in the evaluation of the non-central 3D line fitting from single view we can see how the non-central images generated with our tool conserve the full projection of the 3D lines of the scene. It is possible to recover the metric 3D reconstruction of the points composing these lines. As presented in Bermudez-Cameo et al. (2017) this is only possible when the set of projecting skew rays composing the projection surface of the segment have enough effective baseline.

In this chapter we have presented a set of omnidirectional projection models and we have devised a tool to create a great variety of omnidirectional images, outnumbering the state of the art. We do not only get photorealistic images but also labelled information. We obtain semantic and depth information for each of the omnidirectional systems proposed with pixel precision and can build specific functions to obtain ground truth for computer vision algorithms. The evaluations of our images show that we can use synthetic and real images equally.

Chapter 4

Semantic segmentation and depth estimation from equirectangular panoramas

Equirectangular panoramas reveal advantages when addressing the understanding of the environment due to the 360-degree contextual information. However, the inherent characteristics of the omnidirectional images add additional problems to obtain an accurate detection and segmentation of objects or a good depth estimation. To overcome these problems, we exploit convolutions in the frequential domain, obtaining a wider receptive field in each convolutional layer, and convolutions in the equirectangular projection, to cope with the panoramic image distortion. Both of these convolutions allow to leverage the whole context information from omnidirectional images. In this chapter we present FreDSNet, a neural network for joint depth estimation and semantic segmentation from single panoramas.

4.1 Introduction

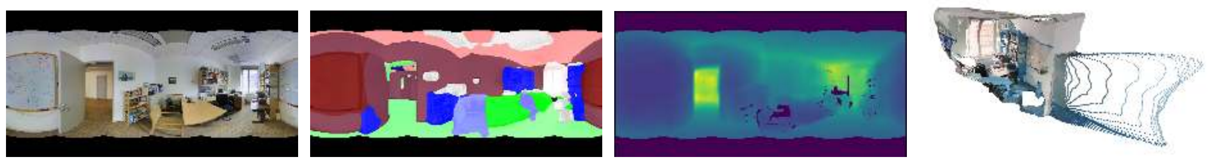


Figure 4.1: Overview of our proposal. From a single RGB panorama (a) we estimate a semantic segmentation (b) and a depth map (c) of an indoor environment. With this information we are able to reconstruct in 3D the whole environment (d).

In recent years, the computer vision and robotics research communities have put a great deal of effort into the 3D scene understanding problem Naseer et al. (2018)Zou et al. (2021). For autonomous vehicles or in the guidance of visually impaired people, it is essential to understand the surroundings in order to interact or manipulate different objects or navigate through obstacles Berenguel-Baeta et al. (2020b)Sanchez-Garcia et al. (2019). Among the different branches in the scene understanding field, the object detection and segmentation has great interest for interaction while depth estimation is more interesting for 3D mapping and navigation.

Previous approaches on scene understanding rely on using conventional cameras with classic algorithms (i.e. ORB-SLAM Mur-Artal et al. (2015) for 3D mapping and localization). With the rise of deep learning approaches, there are also many tools for depth Heo et al. (2018)Ranftl et al. (2016) and semantic segmentation He et al. (2017)Hermans et al. (2014) from perspective images. Even though these solutions provide good results, the conventional camera still lack one key feature in navigation: awareness of the surrounding. Conventional cameras provide a limited field of view which make difficult to obtain a fast knowledge of the environment.

Omnidirectional cameras, as fish eye or panoramic cameras, provide a better understanding of the environments in a single image. With a field of view up to 360 degrees, these cameras encode the whole environment's information into a single image. However, we find some challenges that we do not find on conventional cameras. The first challenge is the image distortion of these omnidirectional cameras. Due to the wide field of view, the projection model of these cameras carry heavy distortion in some areas of the image (i.e. in equirectangular panoramas, the top and bottom part of the images are heavily distorted since, in the extreme, one pixel is stretched in the whole image width). The second challenge is the few number of labelled datasets for general or specific purposes. Since these images have only being used in the last decade and it is difficult to manually annotate the images due to the distortion, there are few datasets and the ones that can be found are not very large (i.e. the Matterport dataset Chang et al. (2017) has around 10k panoramic images while the NYU-Depth v2 dataset Silberman et al. (2012) counts with more than 400k perspective images).

In this chapter, we aim to cope with these challenges. On the first one, we introduce FreDSNet, which jointly provides semantic segmentation and depth estimation from a single equirectangular panorama (see Fig. 4.1). We propose the use of the Fast Fourier convolution (FFC) Chi et al. (2020) to leverage the wider receptive field of these convolutions. We also evaluate the feasibility of convolutions adapted to the equirectangular distortion Fernandez-Labrador et al. (2020) to take advantage of the wide field of view of 360 panoramas. Besides, we propose a joint training of semantic segmentation and depth, where each task can benefit from the other. We further explore the use of different loss functions for semantic segmentation and optimization processes to compute the hyper-parameters for the global loss function. On the second challenge, we present a recollection of semantic segmentation maps in equirectangular projection from the Matterport3D dataset Chang et al. (2017). Obtaining the object information from the 3D meshes of the dataset, we have generated labelled equirectangular panoramas for the joint depth and semantic segmentation task.

Summarizing the contributions presented in this chapter:

- We present the first neural network for joint depth estimation and semantic segmentation for equirectangular panoramas.

- We evaluate convolutions adapted to the equirectangular projection in parallel with the FFC for a better distortion management in the equirectangular panorama.
- We evaluate the robustness and adaptability of several learning-based solutions for monocular depth estimation and semantic segmentation from equirectangular panoramas on more general situations.
- We extend the Matterport3D dataset Chang et al. (2017) with semantic segmentation maps in the equirectangular projection.

4.2 Indoor scene understanding: what can be found in the literature?

Among the 3D scene understanding topics, the semantic segmentation and monocular depth estimation problems have attracted the attention of researchers in the last years. Besides, with the use of omnidirectional images, we can find several approaches to cope with the distortion and leverage context information.

Semantic segmentation The semantic segmentation on perspective images is a well-studied field. We can find many works on object detection Russakovsky et al. (2015), semantic segmentation He et al. (2017)Zheng et al. (2014) or both tasks Dvornik et al. (2017)Girshick et al. (2014) from perspective cameras. These approaches provide good results but lack in context information. On the other hand, the use of omnidirectional images provides a better understanding of the surroundings and new approaches are appearing. We can find works on object detection and segmentation Guerrero-Viu et al. (2020) or only on semantic segmentation Eder et al. (2020) which use convolutional neural network, attention modules Sun et al. (2021) or transformers Zhang et al. (2022). Since omnidirectional images present heavy distortions (e.g. in spherical projections, like equirectangular images, this distortion is more accentuated in the mapping of the poles) these kinds of images are difficult to manually annotate. Nevertheless, due to the wide field of view of these images (e.g. in the spherical projection, we can see all the surroundings in a single image), the use of omnidirectional images in semantic segmentation is an active field of study since we can obtain a complete semantic understanding of the environment from a single image.

Depth estimation Monocular depth estimation is a research topic that has been on the spotlight in recent years. Recent approaches rely on deep learning methods, such as Facil et al. (2019), which proposes convolutions adapted to the camera calibration, Lee and Kim (2019) propose relative depth maps and Ranftl et al. (2016) the use of optical flow, all in perspective cameras. On omnidirectional cameras, first approaches used convolutional networks Zioulis et al. (2018). New ones combine convolutional networks with: recurrent networks Pintore et al. (2021a), attention modules Sun et al. (2021) or transformers Li et al. (2022). The use of twin networks has also been studied in Wang et al. (2020)Wang et al. (2022). Each work presents particular approaches for monocular depth estimation,

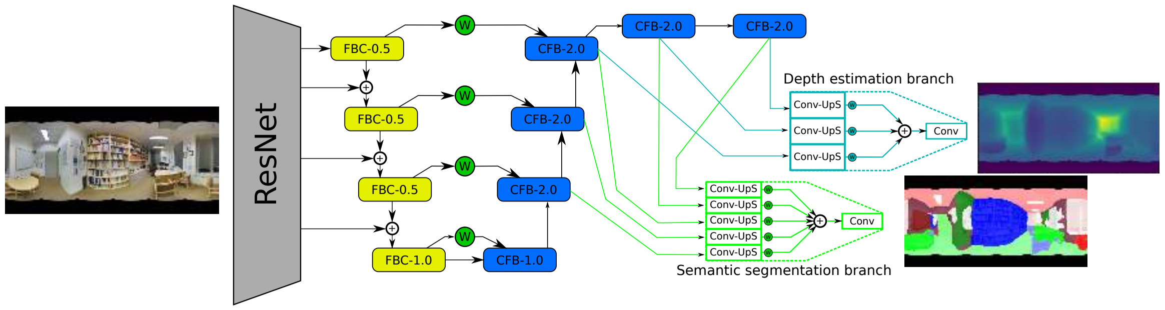


Figure 4.2: Architecture of FreDSNet. The encoder part is formed by a feature extractor (ResNet) and four encoder blocks. The decoder part is formed by six decoding blocks and two branches that predict depth and semantic segmentation.

being an open field of study with great interest and many applications.

Network architecture Previous works on semantic segmentation or depth estimation rely on convolutional encoder-decoder architectures with some recurrent Pintore et al. (2021a) or attention mechanism Sun et al. (2021) as hidden representation of the environment. These approaches of encoder-decoder architecture which rely on standard convolutions Zioulis et al. (2018) suffer from slow growth of the receptive field of the convolutions, losing general context information. More recent approaches overcome this problem decomposing the panoramic image into several gnomic projections to take advantage of standard convolutions Eder et al. (2020) or visual transformers Li et al. (2022), using an spherical geometric positional encoding for feature fusion.

Our method is inspired by works that try to adapt the network to the particular distortion of equirectangular panoramas Zhang et al. (2022). In particular, we propose an encoder-decoder architecture with convolutions adapted to the spherical distortion, EquiConvs Fernandez-Labrador et al. (2020), as well as convolutions in the frequential domain with a higher receptive field, Fast Fourier convolutions (FFC) Chi et al. (2020). With distortion-aware convolutions and a higher receptive field, we are able to obtain more context information of the environment directly from a single panorama. We believe that learning directly from the spherical model provides continuous context information, which is lost in the decomposition of the panorama by patches, and adapting the network to the distortion allow a more robust behaviour against more general conditions (i.e. panoramas under different orientations).

4.3 FreDSNet: monocular depth and semantic segmentation

Our network follows an encoder-decoder architecture, resembling U-net Ronneberger et al. (2015), with Resnet He et al. (2016) as initial feature extractor and two separated branches for depth estimation and semantic segmentation. (see Fig.4.2). It is inspired by BlitzNet Dvornik et al. (2017) and PanoBlitzNet Guerrero-Viu et al. (2020), using multi-resolution encoding and decoding, in order to obtain a multi-scale representation of the scene, and the use of skip connections, which makes the training process more stable. Each branch takes intermediate feature maps from the decoder part to

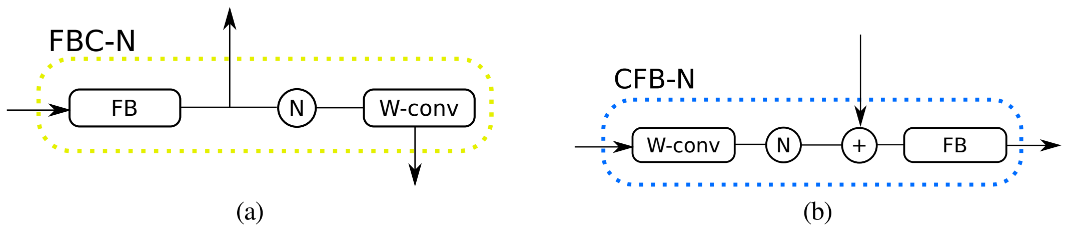


Figure 4.3: a) FBC-N: encoder block composed by a Fourier Block (FB), skip connection for the Decoder part, Down-Scaling (N) of scale N and a W-Conv (C). b) CFB-N: decoder block composed by a W-Conv (C), Up-Scaling (N) of scale N, addition of a skip connection from the Encoder part and a Fourier Block (FB).

provide an output from the multi-scale decoded information. What differentiates our architecture is how the blocks of encoder and decoder are composed and interconnected and the convolutions used, such as the frequential convolutions and the distortion-aware convolutions.

4.3.1 Architecture

The encoder blocks (FBC-N) are formed by a Fourier Block (FB) followed by a down-scaling (N) and a set of convolutions (C) as shown in Fig.4.3a. The Fourier Block has the same structure as the FFC implemented in Suvorov et al. (2022), however we differ in the use of the activation function (AF) and the 3x3 convolutions. In the original work, they use a *ReLu* AF in the FFC (they propose an in-painting method). However, recent works as Pintore et al. (2021a) found that *ReLu* function is not really suited for depth estimation, since it is prone to make gradients vanish. Instead, we use *PReLu* as AF, which is more stable for monocular depth estimation training Pintore et al. (2021a). The same AF change has been made in the *Spectral block*Chi et al. (2020) from the FB in order to homogenize the behaviour of the network. The 3x3 convolutions also change when we use convolutions adapted to the equirectangular panorama Fernandez-Labrador et al. (2020).The output of the FB is the information that we use as skip connection for the decoder part.

After the FB, we re-scale the feature map by a factor of N, followed by a set of convolutions defined as W-conv. The W-conv block is defined as 3 consecutive Convolution-Batch normalization-Activation function blocks with circular padding, taking into account the continuity of panoramic images and their features. This W-conv block follows the ResNetHe et al. (2016) *bottleneck* structure for architecture homogeneity. As defined before, we use *PReLu* as AF. The output of the W-conv is then added, and not concatenated, to the next scale-level output of the feature extractor, as seen in Fig. 4.2.

The decoder part follows the same principle as the encoder but in the inverse order, as shown in Fig 4.3b. The output of the encoder is fed to the decoder blocks (CFB-N) where the feature maps go through a W-conv block and then are up-scaled. The scaled features are added with the skip connection, weighted with a learned parameter, and then go through a FB. The output of each decoder block is then fed to the next decoder block until we recover the full resolution of the network input.

4.3.2 Semantic segmentation branch

We use the last five sets of feature maps from the decoder which contain enough relevant information for the semantic segmentation task. We convolve and upscale each set of features into an intermediate representation. However, instead of concatenating the feature maps, as done in previous works as Dvornik et al. (2017), we make a learned-weighted sum of them to keep a more compact intermediate representation. Finally, we use one more convolution to obtain the semantic segmentation map. In this branch we switch to a *ReLu* activation function instead of the *PReLu* used in the main body of the architecture.

4.3.3 Depth estimation branch

For the depth estimation, we have another branch that takes the last three blocks of feature maps from the decoder part. As done in the semantic segmentation, we convolve and upscale the features to an intermediate representation, where we add them as a learned-weighted sum. Finally, we convolve again the intermediate representation to make the depth estimation. In this branch, we keep the *PReLu* activation function in the convolution from the feature maps to the intermediate representation. However, we switch to a *ReLu* function in the last convolution since depth cannot be a negative value. We tried different output functions, such as the *PReLu* activation function or not using any, but the *ReLu* function provided the best performance.

4.3.4 Equirectangular convolutions

Kernels are the keystone of convolutional neural networks (CNN). Conventional kernels are square filters that go through images or feature maps obtaining new feature maps. Previous works addressed the problem: what happens if we change the shape of the kernel? The work Dai et al. (2017b) was the first to present a learned deformable kernel, improving the performance of CNN on several tasks. In this work, we also propose the use deformable kernels, but in our case we are not learning them. We use the equirectangular projection to compute the kernel deformation adapting our network to the distortion of equirectangular panoramas. Figure 4.4 shown the differences among standard, deformable and equirectangular kernels.

The convolutional kernels that we use in this work are called *EquiConvs* and were first presented by Fernandez-Labrador et al. (2020). Taking advantage of the closed form of the equirectangular projection, we can deform the kernel to adapt its geometry to the equirectangular distortion. This deformation comes from computing a set of offsets for the kernel in each pixel of the image or feature map.

In a similar manner as Fernandez-Labrador et al. (2020), we define a kernel of resolution $r_w \times r_h$ as: $\hat{\mathbf{k_{ij}}} = [i, j, d]$, where i, j are in the range $\left[-\frac{r-1}{2}, \frac{r-1}{2}\right]$ and d is the distance from the center of the sphere to the kernel (in our case, $d = 1$).

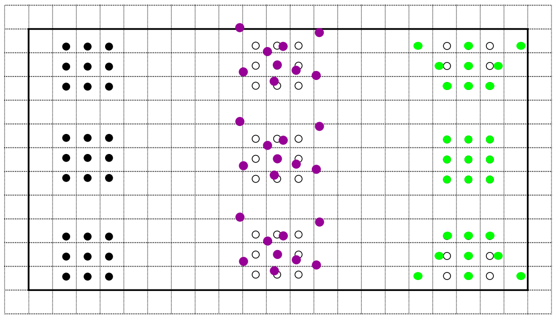


Figure 4.4: Equirectangular panorama with padding and convolutions on different places. From left to right: Standard convolution, learned deformable convolution Dai et al. (2017b), Equirectangular convolution Fernandez-Labrador et al. (2020). In empty black circles are where the standard convolutions would be located.

Defined the kernel as a plane tangent to the sphere, we rotate the kernel to match its center with the pixel coordinates we want to apply the convolution on as: $\mathbf{k_{ij}} = [x_{ij}, y_{ij}, z_{ij}] = \mathcal{R}_{ot}\, \hat{\mathbf{k_{ij}}}/|\hat{\mathbf{k_{ij}}}|$, where $\mathcal{R}_{ot}$ is the rotation matrix to the pixel where we apply the convolution. Once defined the kernel in the correct position, we use the back projection model to define the kernel back into the equirectangular image domain.

$$\phi_{ij} = \arctan\left(\frac{x_{ij}}{z_{ij}}\right);\ \theta_{ij} = \arcsin\left(y_{ij}\right) \tag{4.1}$$

First we compute the spherical coordinates of the kernel's projecting rays (see eq. 4.1) and then we compute the pixel coordinates as:

$$u_{ij} = \left(\frac{\phi_{ij}}{2\pi} + 0.5\right)W;\ v_{ij} = \left(-\frac{\theta_{ij}}{\pi} + 0.5\right)H, \tag{4.2}$$

where (W, H) is the panorama width and height and (u_{ij}, v_{ij}) are the pixel coordinates of the kernel in the image.

4.3.5 Loss functions

Semantic segmentation and depth estimation have common characteristics that can benefit from each other Zhang et al. (2018)(i.e. object edges are clear on the semantic segmentation map and define a discontinuity in the depth map; room layouts define 3D planes in the depth map and can aid the detection of the structural elements in the segmentation map). We want to take advantage of these similarities making a joint training where the semantic segmentation and the depth estimation can be jointly predicted.

For the semantic segmentation loss L_{Seg} , we use the standard *Cross Entropy Loss* and weights for the classes, as a solution for the class imbalance in the dataset. Besides, we also compare this solution with the proposed in Tian et al. (2022) called *Recall Loss*. This loss is computed as:

$$L_{Recall} = -\sum_{c=0}^{C} \frac{FN_c}{FN_c + TP_c} N_c \log(P^c), \tag{4.3}$$

using the same notation as in the original paper where FN_c is false negatives of class c, TP_c the true positives, N_c the count of c class pixels and P^c is the geometric mean confidence of the class.

For the depth estimation loss L_{Dep} we keep using the *Adaptive Reverse Huber Loss* (eq. 4.4), defined as:

$$B_c(e) \begin{cases} |e| & |e| \leq c \\ \frac{e^2+c^2}{2c} & |e| > c \end{cases}, \tag{4.4}$$

where $e = Prediction - GroundTruth$ and c is defined as the 20% of the maximum absolute error for each training batch. Following the same idea as Pintore et al. (2021a), we also define the loss function as the sum of the *Adaptive Reverse Huber Loss* on the depth map as well as the gradients (approximated as Sobel Filters). The final L_{Dep} is computed as:

$$L_{Dep} = B_{c_1}(e) + B_{c_2}(\nabla_x) + B_{c_2}(\nabla_y), \tag{4.5}$$

where e defines the absolute depth error between the prediction and ground truth, ∇_x,∇_y define absolute error between the x, y gradients of the prediction and ground truth respectively, c_1 is the threshold in eq. 4.4 for the absolute depth map and c_2 is the threshold in eq. 4.4 for the gradients.

We add another two losses to help in the joint training process. The first one is the margin loss, in order to force the depth estimation branch to fill the depth range between the closest and farthest pixels. We compute the mean square difference between the minimum and maximum values of prediction and ground truth in each batch as:

$$L_{mar} = \frac{\left(y_{gt}^{max} - y_{pred}^{max}\right)^2 + \left(y_{gt}^{min} - y_{pred}^{min}\right)^2}{2}, \tag{4.6}$$

where y_{gt}^{max}, y_{pred}^{max}, y_{gt}^{min} and y_{pred}^{min} are the maximum and minimum values of the ground truth and predicted depth maps respectively. The second one is an object oriented loss L_{obj}. This loss helps the network to better define the objects boundaries as well as create the depth discontinuities that appear in these boundaries. To compute the loss, we first compute per-class depth maps from the network prediction and ground truth. Then we compute the mean of the *mean squared error* (MSE) of each class depth map to obtain the final L_{obj} as:

$$L_{obj} = \frac{1}{C} \sum_{i=0}^{C} ({y_{gt}}^i - {y_{pred}}^i)^2, \tag{4.7}$$

where C is the number of classes and ${y_{gt}}^i$,${y_{pred}}^i$ are the ground truth and predicted class depth maps for the class i respectively.

Our final training loss is the combination of the previous losses. This joint loss function is computed as:

$$L_{total} = \alpha_1 \cdot L_{Seg} + \alpha_2 \cdot L_{Dep} + \alpha_3 \cdot L_{mar} + \alpha_4 \cdot L_{obj}, \tag{4.8}$$

where α_i are hyper-parameters to weight the relevance of each individual loss in the final joint loss function. At first, we set these hyper-parameters to $\alpha = [8.0, 12.0, 0.001, 4.0]$. After a first training, we make a Bayesian optimization with the tool *Adaptive Experimentation Platform*[1] to optimize these hyper-parameters to obtain the best validation metric for our network.

4.4 Panoramic Matterport3DS dataset

The Matterport3D dataset Chang et al. (2017) is a great recollection of perspective images taken in indoor environments with an array of cameras to cover 360 degrees. The dataset provides the color images and depth scans as perspective images. Furthermore, they also provide a 3D reconstruction of the different environments as 3D meshes with semantic information in the nodes of the meshes. In order to obtain semantic and depth information in the equirectangular projection, we generate the panoramic images from the raw data provided in the original dataset, building the *Panoramic Matterport3DS* dataset.

To obtain the color panoramas, we blend the perspective images in the equirectangular projection using Gaussian Pyramids Burt and Adelson (1983), reducing the artefacts in the intersection of the perspective images. The blending also takes into account the brightness of the scene, adapting the brightness of the perspective images to avoid different illumination conditions in the same panorama. Depth maps are build blending the perspective scans into the equirectangular projection and computing the average depth in overlapping areas. Also, depth measures are corrected with the extrinsic calibration of the array of cameras to compute the distance from a single point on each panorama (central projection system).

Semantic segmentation maps are retrieved from the 3D meshes of each environment. We build a pseudo-virtual environment with the 3D mesh and fill each triangle of the mesh with the label of the nodes. 3 cases can appear: if the three nodes have the same label, the triangle takes that semantic label; if one node has a different label than the other two, the triangle takes the semantic label of the two nodes; if each node has a different label, the triangle takes no semantic label, that means, that triangle get the ”*unknown*” label. Once build the pseudo-virtual environment, the image acquisition follows the same pipeline as OmniSCV Berenguel-Baeta et al. (2020a). We obtain 6 perspective images (a.k.a. cube map) in the same locations as the color images and project these images into the equirectangular projection.

[1] github.com/facebook/Ax

4.5 Experiments

In this section we present a set of experiments to validate several architectures and training methods for FreDSNet. We also make a comparison with a state of the art task-specific architecture for depth estimation and semantic segmentation. Besides, we make a study-comparison of our proposal with state-of-the-art network in no-gravity-oriented panoramas.

To evaluate our work and compare it with the state of the art, we use the following metrics. For the depth estimation task, we use the metrics introduced by Zioulis et al. (2018), used by most of recent works on monocular depth estimation. We use the Mean Relative Error (*MRE*), Mean Absolute Error (*MAE*), Root-Mean Square Error of linear (*RMSE*) and logarithmic (*RMSElog*) measures, and three relative accuracy measures defined as the fraction of pixels where the relative error is within a threshold of 1.25^n for $n = 1,2,3$ ($\delta^1, \delta^2, \delta^3$). On the other hand, for the semantic segmentation task, we use standard metrics as the mean Intersection over Union (*mIoU*), computed as the average *IoU* for each class except the *Unknown* class; and the mean Accuracy (*mAcc*), computed as the average accuracy for each class except the *Unknown* class.

4.5.1 Ablation study

In this section we evaluate different training methods on several network configurations. For a better understanding, we split the results obtained in 7 experiments. In the first 2 experiments we only use FreDSNet with standard convolutions while in the last 5 experiments, we will compare FreDSNet with standard convolutions or with *EquiConvs*. We use 2 metrics for each task: *MRE* and *MAE* for the training process of depth estimation, *MRE* and δ^1 for the optimization and architecture evaluation of depth estimation and *mIoU* and *mAcc* for all the experiments of semantic segmentation. Training is performed in one GPU NVIDIA GeForce RTX3090-Ti with an initial learning rate of $1e-5$. We use the first folder split from Stanford2D3DS Armeni et al. (2017) for training, leaving Area 5 only for test. We make an inner split in the train set (all areas except Area 5) of $80\% - 20\%$ as train-validation sets. We use the validation set for comparison in this ablation study and finally we use the test set to select the network that better generalizes to unknown environments.

We name each network with a four-component code such as **F-Std-CE-X**[2] where: the first component describes if we use an spectral block (**F**) or if we use a convolutional layer in the Fourier Block (**X**); the second component describes if we use standard convolutions (**Std**) or Equiconvs (**Equi**); the third component describes the semantic segmentation loss used during training as (**CE**) for the Cross Entropy Loss or (**RL**) for the Recall Loss; and the last component describes if we use the Bayesian optimization (**B**) for the tune of hyper-parameters or if we don't (**X**). Furthermore, we add a fifth component as (**r**) only if the network has been trained on rotated panoramas.

[2]This example describes a network with FFC and Standard convolutions, trained with the Cross Entropy loss function and with hand-set hyper-parameters on gravity-oriented panoramas.

Table 4.1: Ablation study of the combination of Loss functions. The comparison is made on the validation set.

Network	L_{Seg}	L_{Dep}	L_{mar}	L_{obj}	MRE ↓	MAE ↓	mIoU ↑	mAcc ↑
F-Std-CE-X	✓	✓	×	×	0.0613	0.0950	60.3	81.9
F-Std-CE-X	✓	✓	×	✓	0.0583	0.0855	61.5	82.5
F-Std-CE-X	✓	✓	✓	×	0.0560	0.0898	60.6	81.6
F-Std-CE-X	✓	✓	✓	✓	0.0553	0.0827	62.7	84.2

Table 4.2: Ablation study of joint training. Best validation weights are used in each of the cases for the evaluation in the test set of the dataset.

Network	Training	MRE ↓	MAE ↓	mIoU ↑	mAcc ↑
F-Std-CE-X	Depth	0.1401	0.1773	-	-
F-Std-CE-X	Semantic	-	-	40.3	55.1
F-Std-CE-X	Joint	0.0952	0.1327	46.1	63.1

Table 4.3: Ablation study. Semantic Loss functions comparison evaluated on the validation set.

Network	MRE ↓	δ^1 ↑	mIoU ↑	mAcc ↑
F-Std-CE-X	0.0589	0.9059	62.40	84.47
F-Std-RL-X	0.0545	0.9172	62.69	79.59
F-Equi-CE-X	0.0626	0.9009	62.26	84.14
F-Equi-RL-X	0.0576	0.9145	62.26	78.96

Table 4.4: Ablation study. Bayesian optimization evaluated on the validation set.

Network	MRE ↓	δ^1 ↑	mIoU ↑	mAcc ↑
F-Std-CE-B	0.0412	0.9404	71.12	89.76
F-Std-RL-B	0.0408	0.9409	75.61	88.73
F-Equi-CE-B	0.0461	0.9378	71.45	89.15
F-Equi-RL-B	0.0510	0.9258	72.56	86.42

Table 4.5: Ablation study. Convolution influence evaluated on the validation set.

Network	MRE ↓	δ^1 ↑	mIoU ↑	mAcc ↑
F-Std-CE-X	0.0589	0.9059	62.40	84.47
X-Std-CE-X	0.0629	0.8998	62.27	84.29
F-Equi-CE-X	0.0626	0.9009	62.26	84.14
X-Equi-CE-X	0.0666	0.8862	58.43	83.13

Table 4.6: Ablation Study. Performance on unknown environments. We evaluate different training methods and proposed architectures on the test set. Selected network are in bold.

Network	MRE ↓	δ^1 ↑	mIoU ↑	mAcc ↑
F-Std-CE-X	0.0961	0.8313	44.85	64.45
F-Std-RL-X	0.0888	0.8416	42.61	58.63
F-Std-CE-B	0.0968	0.8481	45.64	63.76
F-Std-RL-B	0.0964	0.8483	46.31	58.79
X-Std-CE-X	0.0979	0.8147	44.31	64.01
F-Equi-CE-X	0.0993	0.8270	43.29	63.27
F-Equi-RL-X	0.0945	0.8414	42.01	56.31
F-Equi-CE-B	0.1010	0.8533	45.32	62.24
F-Equi-RL-B	0.1046	0.8357	44.82	56.56
X-Equi-CE-X	0.0999	0.7927	42.35	62.90

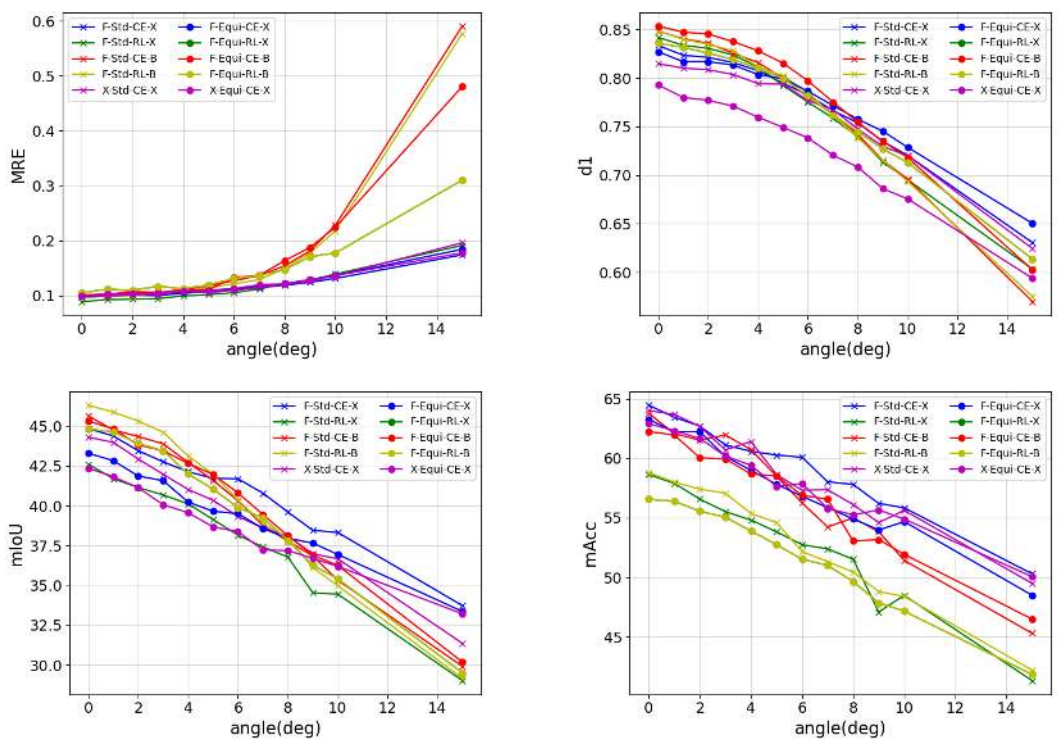


Figure 4.5: Ablation Study of the different training and architectures under different rotation angles. First row are depth metrics while second row are semantic segmentation metrics.

Combined loss function. We evaluate how the overall performance of our network is affected by each loss function. For that purpose, we perform the same joint training using the task-specific losses and adding sequentially the several losses that we propose in Section 4.3.5. Results are shown in Table 4.1.

Joint training. On a second experiment, we train our network for task-specific purposes, that means with only one of the branches at a time, and in the proposed joint training, with both branches. Notice that only half of the metrics are used for the task-specific training, since the other half correspond to a different task. In the task-specific training, we only use the specific loss function for each task, i.e. we only use L_{Dep} for the specific depth estimation training and L_{Seg} for the specific semantic segmentation training. The results are presented on Table 4.2.

Semantic loss function. In this experiment we compare the performance of the network with the two proposed loss functions for semantic segmentation: *Cross Entropy* and *Recall Loss*. The training of these networks has been done from scratch and with the hyper-parameters set by hand, that is, $\alpha = [8.0, 12.0, 0.001, 4.0]$. Results are shown in Table 4.3.

Hyper-parameter optimization. After the first training of the networks, we perform a Bayesian optimization in the hyper-parameters in order to improve their performance. The evaluation results are shown in Table 4.4.

Convolutions. Additionally, we perform another experiment to evaluate the influence of each type of convolution in the final performance of our network. In this experiment we compare FreDSNet with the same architecture but changing the *Spectral block* from the *Fourier Block* by a convolutional layer, standard or equirectangular. The results are shown in Table 4.5.

Performance on unknown environments. Finally, we evaluate the generalization to unknown environments of the several training configurations proposed using the test split of Stanford2D3DS (Area 5 of the dataset). From this evaluation we will select the two best networks for the following experiments: we take the best network with standard convolutions and the best with EquiConvs. The results are shown in Table 4.6.

Non gravity-oriented panoramas. Indoor datasets of panoramic images are often aligned with the gravity direction. This means that the distribution of the environment is almost always the same: floor in the bottom of the image, ceiling on top and walls along the middle. However, for indoor navigation systems, this distribution can change (i.e. while flying a drone). In this experiment, we evaluate the robustness of different networks to no-gravity-oriented panoramas. All networks have been trained on gravity oriented panoramas in the Stanford dataset and evaluated in the *Area 5* rotating the panoramas around an axis in the $X-Y$ plane and a fixed angle. Results of this experiment are shown in Figure 4.5.

Table 4.7: Quantitative comparison for Depth Estimation on gravity-oriented panoramas. Evaluation made in the Stanford2D3DS dataset (S2D3DS) Armeni et al. (2017) and Matterport3D dataset (M3D) Chang et al. (2017). In **bold** are the two best metrics on each dataset.

Dataset	Network	MRE ↓	MAE ↓	RMSE ↓	RMSElog ↓	δ^1 ↑	δ^2 ↑	δ^3 ↑
S2D3DS	HohoNet	**0.0812**	**0.1162**	0.3137	**0.0367**	**0.8838**	**0.9674**	**0.9880**
	OmniFusion	**0.0798**	0.1566	**0.1443**	0.0480	**0.8619**	**0.9654**	0.9771
	F-Std-CE-B	0.0968	0.1320	0.2738	**0.0430**	0.8481	0.9589	**0.9863**
	F-Equi-CE-B	0.1010	**0.1305**	**0.2638**	0.0439	0.8533	0.9579	0.9848
M3D	HohoNet	0.8847	0.4170	**0.5610**	**0.0576**	**0.8409**	**0.9483**	**0.9781**
	SliceNet	0.8434	0.4398	**0.4836**	0.1036	**0.7974**	**0.9231**	0.9671
	F-Std-CE-B	**0.7220**	**0.3815**	0.7036	**0.0782**	0.7517	0.9201	**0.9695**
	F-Equi-CE-B	**0.7022**	**0.3883**	0.6635	0.0846	0.7366	0.9125	0.9658

Table 4.8: Quantitative comparison for Semantic Segmentation on gravity-oriented panoramas. Evaluation made in the Stanford2D3DS dataset (S2D3DS) Armeni et al. (2017) and Matterport3D dataset (M3D) Chang et al. (2017). Greater metrics is better. In **bold** are the two best metrics on each dataset.

Dataset	Network	mIoU	mAcc
S2D3DS	HohoNet	35.6	45.7
	Trans4PASS	26.5	38.7
	F-Std-CE-B	**45.6**	**63.8**
	F-Equi-CE-B	**45.3**	**62.2**
M3D	HohoNet	-	-
	Trans4PASS	-	-
	F-Std-CE-B	**35.4**	**57.4**
	F-Equi-CE-B	**37.3**	**52.5**

4.5.2 State of the art comparison

We compare the networks selected in section 4.5.1 with task-specific state-of-the-art methods for semantic segmentation and depth estimation. For the semantic segmentation task, we compare our network with HohoNetSun et al. (2021) and Trans4PASSZhang et al. (2022). We evaluate and compare all networks in the *Area 5* of the Stanford2D3DS dataset (S2D3DS) Armeni et al. (2017), following the first folder split. We also provide quantitative results of our network in the Matterport3D dataset (M3D) Chang et al. (2017) where we obtain the labels for semantic segmentation from the 3D meshes of the dataset and stitch the panoramas with the same labeling as the Stanford2D3DS dataset. Results of other networks are not presented since there is no other work that evaluates semantic segmentation on the Matterport3D dataset with equirectangular panoramas.

For the depth estimation task, we compare our work with HohoNet again, SliceNetPintore et al. (2021a) and OmniFusionLi et al. (2022). We also evaluate the networks on the Stanford2D3DS dataset (S2D3DS), first folder split, and Matterport3D dataset (M3D), for the available weights of each method.

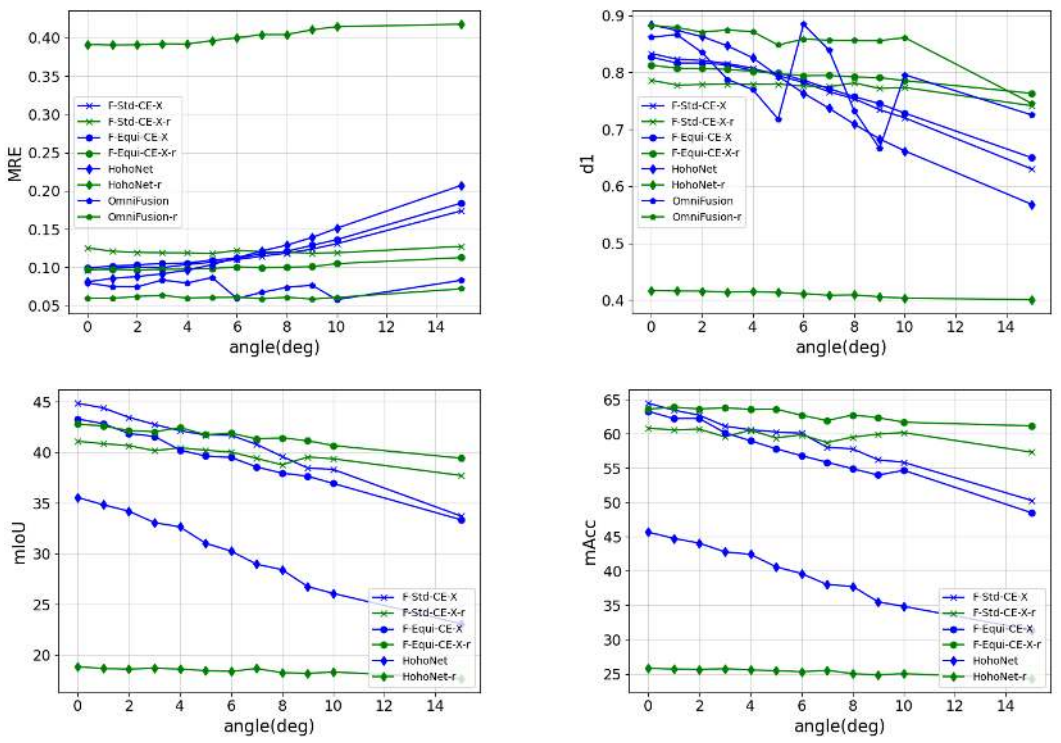


Figure 4.6: Comparison of the different state-of-the-art methods under different rotation angles. The methods which name finish as ”-r” have been trained on rotated panoramas. First row are depth metrics while second row are semantic segmentation metrics.

Notice that HohoNetSun et al. (2021) appears in both comparisons. This is the only other method that provides both semantic segmentation and depth estimation from a similar network architecture. However, since HohoNet uses ground truth depth as well as RGB data as input for semantic segmentation, for a fair comparison, we use the depth output of HohoNet as input for the semantic segmentation task (*this configuration is different from their original experiments*).

We also evaluate how well different networks generalize to no-gravity-oriented panoramas. For that purpose, we evaluate the networks in a similar way as the experiment of the ablation study: Non Gravity-oriented panoramas. Besides, we also train the networks on no-gravity-oriented panoramas to evaluate their behaviour when they have learned under these circumstances. We use a similar denomination for state-of-the-art networks as for our own, including at the end of their name $-r$ to identify that are trained on no-gravity-oriented images. The training is made rotating the panoramas around an axis in the $X-Y$ plane and an angle sampled from a uniform distribution $\mathcal{U}(-10^{\circ}, 10^{\circ})$. The evaluation is made rotating the panoramas around an axis in the $X-Y$ plane and a fix angle. Results of this experiment are show in Figure 4.6.

The quantitative results of the evaluation of depth estimation and semantic segmentation are presented in Table 4.7 and Table 4.8 respectively. We also present a qualitative comparison for depth estimation and semantic segmentation on Figure 4.7 and Figure 4.8 respectively.

4.6 Experiment's results and discussion

Ablation study: architecture and training. In section 4.5.1 we have presented several network configurations and training methods. From the results shown in table 4.1 we observe that the use of several losses increase the performance of our network. We have a greater improvement with the object oriented loss L_{obj} in the semantic segmentation. This loss uses depth and semantic information, providing an improvement in both branches as well as the shared encoder-decoder. In addition with the margin loss L_{mar}, which helps to improve the depth estimation, we observe that each task effectively benefit from each other. In the experiment for the joint training, Table 4.2, we confirm our first intuition, which confirm the results of Zhang et al. (2018). The joint training of semantic segmentation and depth estimation increases the performance of the network for both tasks. From the results of Table 4.3 we infer that the Cross Entropy and Recall Loss have similar performance, being better the depth estimation with the Recall loss and better the semantic segmentation with the Cross Entropy (taking into account both semantic metrics). This behaviour may be due to the balance between both branches of the network, which may fight each other for a better performance. When tuning the hyper-parameters with the Bayesian optimization, Table 4.4, the metrics obtained during training increase significantly. This can mean two things: the network generalizes better or the fine tune over fits to the validation set. The solution can be found in Table 4.6, where we can see that the Bayesian optimization does not provide a great improvement in the performance of the network, which may lead that this fine tuning over fits the network to the validation set. Nevertheless, these networks are the ones that provide the best performance on gravity-oriented panoramas. In Table 4.5 we compare the performance of our network with and without the *Spectral block*, obtaining better results when we do use it. This strengthen the approach of FreDSNet in the use of convolutions in the frequential domain.

Ablation study: no-gravity-oriented panoramas. In the last experiment from section 4.5.1, we evaluate the robustness of several configurations of our networks and we obtain interesting results. In the comparison with networks trained on gravity-oriented panoramas (Fig. 4.5) we observe that, on small angles, networks with the Bayesian optimization provide the best performance. However, as we increase the angle of rotation, these networks decrease their performance faster than any other, ending as the worst solution. Our intuition is that Bayesian optimized networks indeed over fit the validation data, generalizing worse on unknown environments. In the comparison of convolutions, on depth estimation we can observe that, for all the angles, the networks with equirectangular convolutions provide the best performance, at first *F-Equi-CE-B* and for greater angles *F-Equi-CE-X*. However, for semantic segmentation this behaviour does not apply and *F-Std-CE-X* prevails as the best option for almost all angles. These results show that equirectangular convolutions may not provide the best performance for gravity-oriented equirectangular panoramas. Nevertheless, we do see a trend where Equiconvs are more robust than standard convolutions on non gravity-oriented panoramas presenting less performance decay.

On the other hand, when we train the networks on non-gravity-oriented panoramas (Fig. 4.6), we observe that the initial performance is worse that those trained on gravity-oriented, as can be expected. However, as we increase the rotation angle, those trained on non-gravity-oriented panoramas are significantly more robust. We also observe that the network with the Equiconvs performs better that the network with standard convolutions, validating their use on more general conditions.

State of the art comparison. The experiments from section 4.5.2 show that our method has similar performance on depth estimation than the other method that provides both semantic segmentation and depth estimation and close performance task-specific methods and better performance on semantic segmentation. The qualitative comparison on depth estimation (Fig. 4.7), the results may be different, being more visible the image patches on the transformer approach while convolutional methods provide smoother maps. However, the qualitative comparison on semantic segmentation (Fig. 4.8) shows that all methods provide smooth results, being more accurate the convolutional methods. When we apply rotations to the panoramas, the behaviour changes significantly. From the depth results shown in Fig. 4.6, we observe an uneven behaviour in the performance of OmniFusionLi et al. (2022). Our intuition is that the division of the panorama into smaller patches disrupts the positional encoding of the network making it less robust than the other methods. Also, even if HohoNetSun et al. (2021) provides better results than the others on small angles, the performance decreases really fast for bigger angles. A similar behaviour can be seen with the semantic results, where the performance of HohoNet decreases significantly in comparison with our approach. Our intuition is that the 1D representation of the environment of HohoNet and the patch division of OmniFusion does not allow the network to generalize to different orientations of the panorama while the fast Fourier convolutions and EquiConvs are more suited for more general conditions.

With the networks trained on rotated panoramas, the performance of HohoNet is greatly worsened. Even if the performance is more constant than the original training of HohoNet, the quantitative results are worse than other methods. We theorize that the 1D representation of this particular architecture cannot handle different orientations during training, which leads to worse performances on evaluation. On the other hand, OmniFusion performance increases and is more robust to angles that has seen while training (i.e. in our experiment setup, angles $\leqq 10°$). However, it suffers a bigger performance decay for new angles than our proposed methods. We believe that OmniFusion learns the rotated patch embedding while training but cannot generalize outside training data. In this aspect, FreDSNet is able to better adapt to a more general problem, obtaining more robust and stable results in unknown conditions.

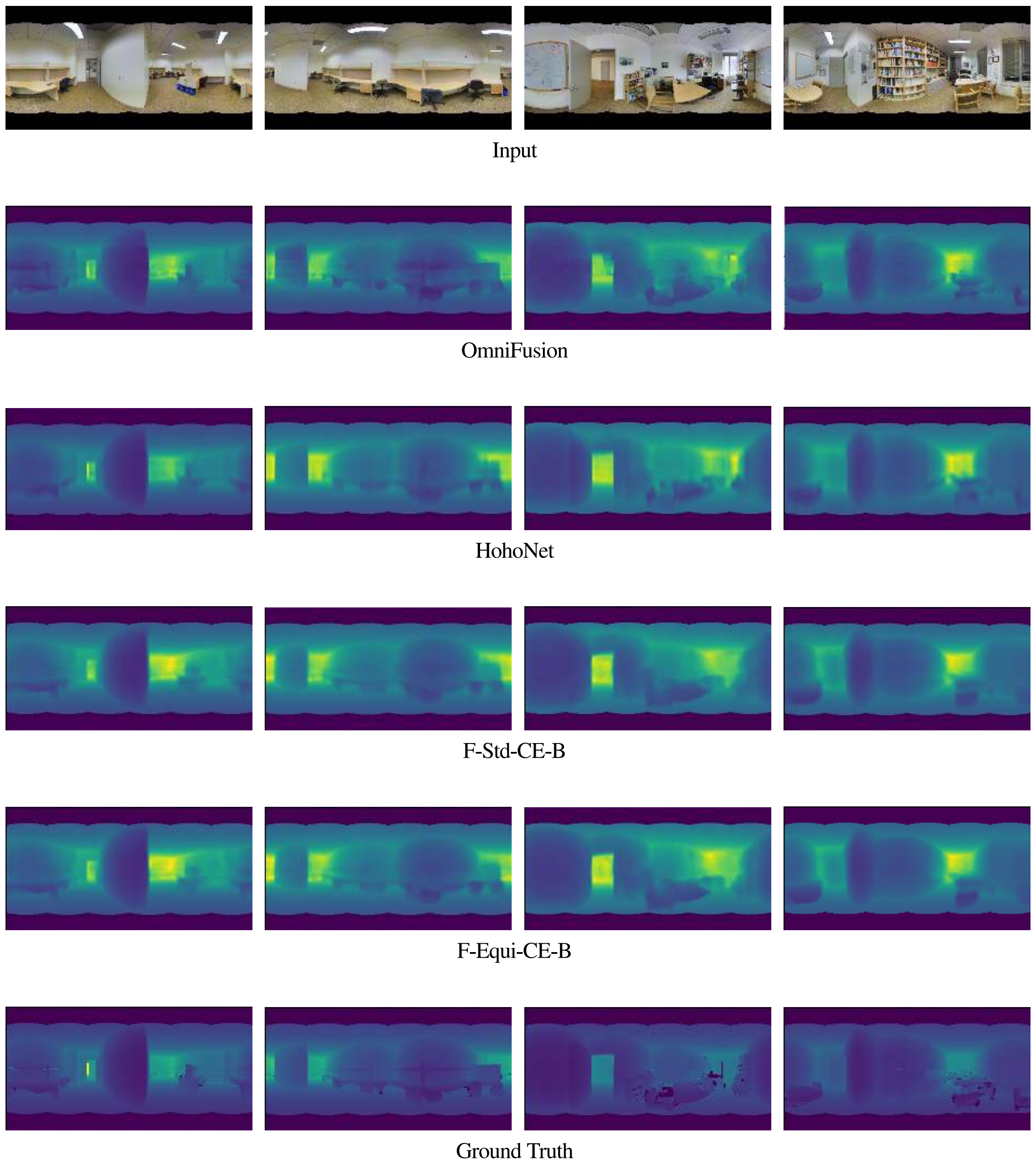


Figure 4.7: Qualitative comparison of state of the art methods for depth estimation on gravity-oriented panoramas in the Stanford2D3DS dataset Armeni et al. (2017).

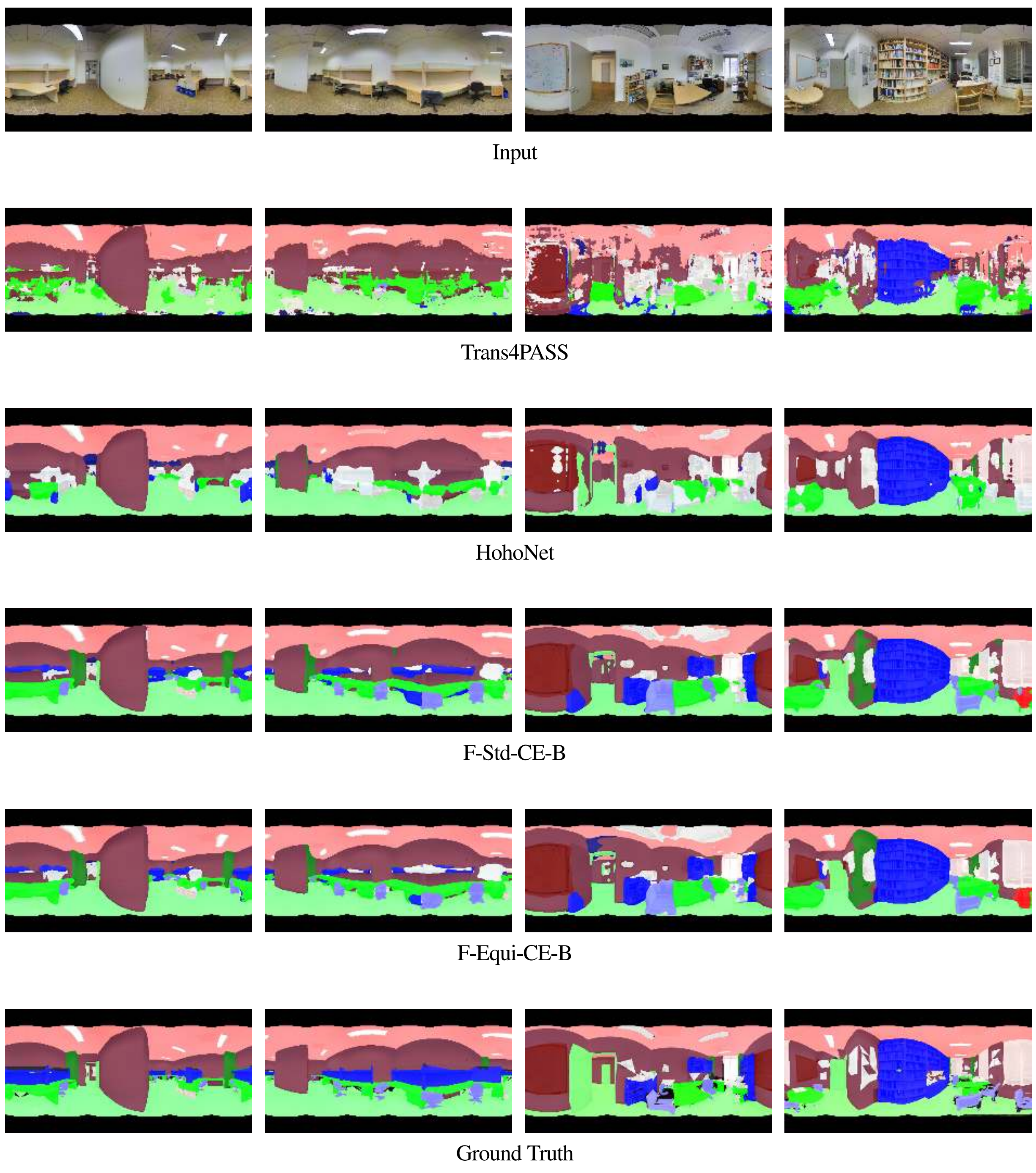


Figure 4.8: Qualitative comparison of state of the art methods for semantic segmentation on gravity-oriented panoramas in the Stanford2D3DS dataset Armeni et al. (2017).

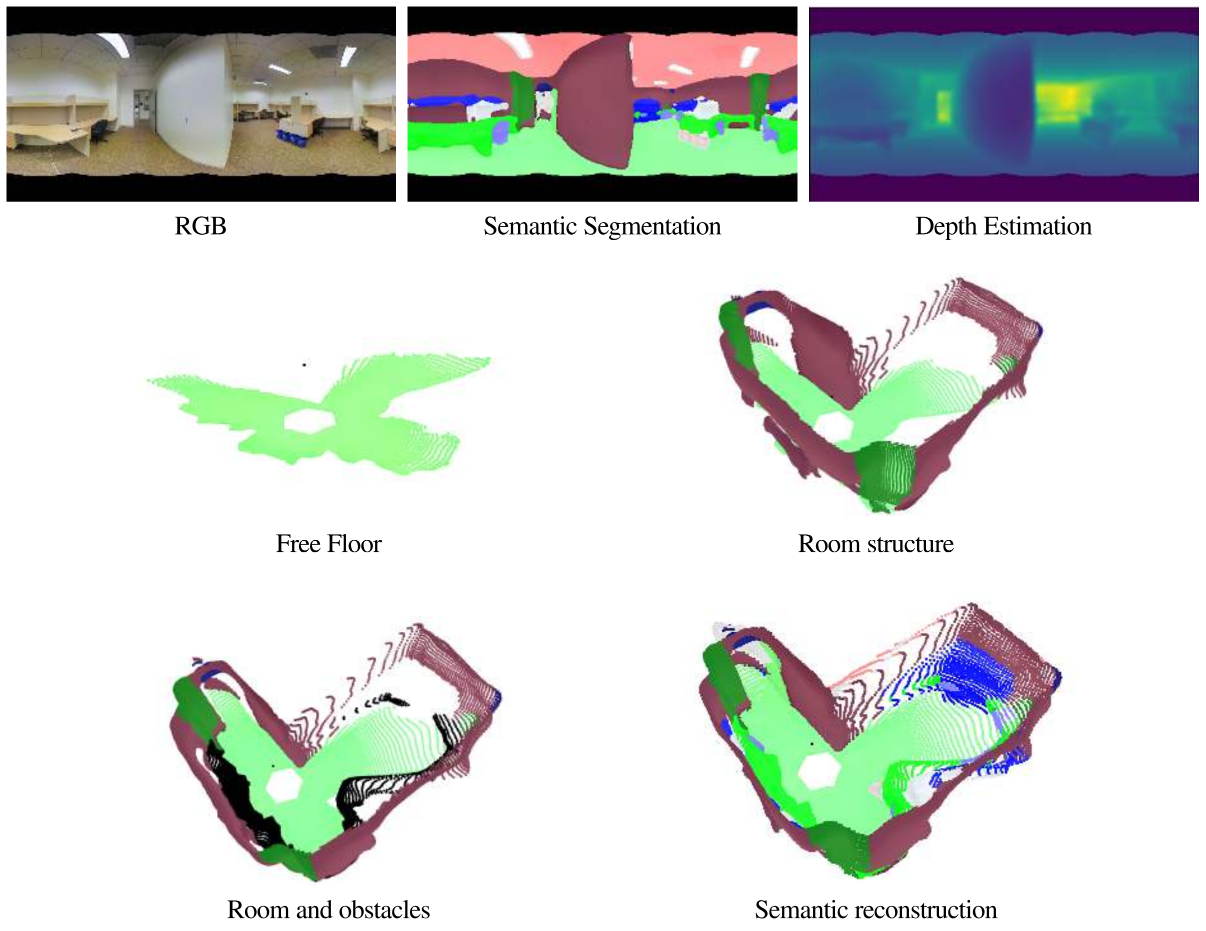


Figure 4.9: In the first row: RGB is the input of our network which outputs the Semantic Segmentation and Depth estimation. In the second and third rows: different useful environment representations that can be obtained from the output information provided by FreDSNet. (For a better representation, the ceiling has been removed from all visualizations)

4.7 Scene understanding for navigation: an example of application

We present different results from our network and ideas of applications. With the combination of semantics and depth, we can extract the free space for navigation (extracting the floor) or compute where the obstacles are located (computing the position of the different segmented objects).

Navigation algorithms for mobile robots require to detect the obstacles and the free space around the vehicle. With an omnidirectional camera we can obtain RGB information of the surroundings in one shot. Then, FreDSNet can simultaneously obtain a semantic and depth maps of the environment. With this information, we can obtain different useful representations of the scene, allowing a better interaction of a mobile vehicle with the environment allowing a robot to be autonomous in unknown environments. Also, in the case of autonomous vehicles, it is important to be able to work in real time. We have evaluated the feasibility of our network for such a task obtaining that it can work at 33 frames per second with panoramas of 512×256 pixels of resolution (average speed computed with the test set of the dataset).

As an example of the information that can be obtained from our network, in Figure 4.9 we present several useful environment representations from a single equirectangular panorama. From left to right in the Figure, we show: free floor reconstruction, thought for terrestrial robot navigation; Room structure reconstruction, defines the maximum space that a vehicle can move; Room and obstacles, includes the room structure and the obstacles (in black) of the room; and the semantic reconstruction, which defines the environment and the different objects with which an autonomous vehicle can interact in the environment.

4.8 Conclusions

In this chapter we have presented FreDSNet. We have conducted an extended set of experiments and evaluated several configurations of the network. In the training process, we determine that the *Recall Loss* does not really improve the performance of our network, obtaining better results with the classical *Cross Entropy*. Besides, the joint training and combined loss function really improves the performance of our network, benefiting each task from the other.

With respect to the use of different convolutions, we observe that convolutions adapted to the equirectangular projection can improve the robustness of our proposal against no-gravity-oriented panoramas, providing a better generalization on more general and challenging conditions. Besides, we also prove that the fast Fourier convolutions can adapt to the image distortion of equirectangular panoramas, since we obtain better results when we use them than on networks with only standard or equirectangular convolutions.

Our experiments show that FreDSNet has slightly better performance than the sole state-of-the-art method that obtain both tasks, depth estimation and semantic segmentation in equirectangular images. However, task specific methods can achieve better performance on each individual task. Nevertheless, while other methods obtain depth and semantic maps separately, FreDSNet is able to obtain both tasks simultaneously and from a single RGB panorama. Besides, experiments on more general and challenging conditions show that FreDSNet is able to generalize better than other state-of-the-art methods, providing more stable and robust results in unknown conditions.

Chapter 5

Scaled layout recovery from non-central panoramas

In this chapter we present a novel approach for 3D layout recovery of indoor environments using non-central panoramic images. From a single image of a non-central system, full and scaled 3D lines can be independently recovered by geometry reasoning without additional nor scale assumptions. However, their sensitivity to noise and complex geometric modelling has led these panoramas and required algorithms being little investigated. Our new pipeline aims to extract, from a single non-central panorama, the boundaries of the structural lines of an indoor environment with a neural network and exploit the properties of non-central projection systems in a new geometrical processing to recover scaled 3D layouts. We completely solve the problem both in Manhattan and Atlanta environments, handling occlusions and retrieving the metric scale of the room without extra measurements.

5.1 Introduction to 3D layout estimation

Layout recovery and 3D understanding of indoor environments is a hot topic in computer vision research Naseer et al. (2018); Zou et al. (2021). Recovering the information of an environment from a single view is an attractive tool for different applications such as virtual or augmented reality Karsch et al. (2011) and human pose estimation Fouhey et al. (2014); dos Reis et al. (2021). Previous works for layout recovery relied on pure geometrical processing Wei and Wang (2018). Those methods usually required hard layout assumptions and iterative processes in order to obtain proper results. Besides, since many hypotheses and verifications should be made, these approaches derive in very slow implementations, not suitable for real time applications. The development of neural networks made the problem of layout recovery more accurate, efficient and faster. The high and low level features obtained by deep learning architectures have proven to be useful for structural recovery of indoor environments.

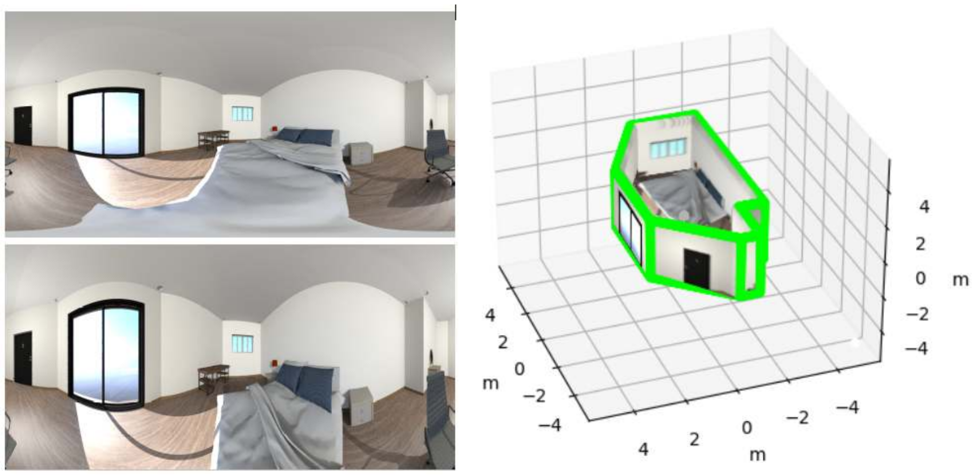


Figure 5.1: Central (top-left) and non-central (bottom-left) panoramas from the same virtual environment taken in the same position. Both panoramas have similar appearance but there are subtle differences in favour of the second if we want to obtain 3D information. On the right, the scaled layout from a single non-central panorama in Atlanta world. The green wireframe shows the real 3D layout of the virtual environment.

Through the development of algorithms for layout recovery, different kinds of acquisition systems have been used in order to obtain more information of the environment with as less images as possible. An example is the evolution from perspective images to equirectangular panoramas, which acquire much more information of the environment in a single image. With this extra information, better reconstructions from more complex environments could be made. We propose to go a step further and evolve taking as acquisition system the non-central circular panoramas proposed in Li et al. (2004); Menem and Pajdla (2004) (for simplicity, we will call these as non-central panoramas). These non-central panoramas provide 360 information of the environment and the image distortion of the non-central acquisition systems includes subtle differences allowing geometric 3D reasoning. In particular, the distortion of the curves fitting the projections of lines encodes the full 3D description of the line. This characteristic of non-central projection systems is a clear advantage for environment reconstruction, since it allows to recover the scale of the environment directly from the image, without measurement assumptions (e.g. camera position or room height). However, due to their sensitivity to noise and complex geometric reasoning, non-central panoramas have been little investigated. Figure 5.1 shows two panoramas, one central and the other non-central, in the same environment and from the same position. From the non-central panorama we can recover a more accurate layout, including the real scale.

In this chapter we present the first proposal of layout recovery with single non-central panoramas and the first deep learning approach for this kind of images. We propose to adapt the neural network architecture of HorizonNet Sun et al. (2019) to non-central circular panoramas for the extraction of the boundaries of the structural lines from indoor environments. As done by Sun et al. (2019), we assume that our panoramas are horizontally oriented and that the layout share the ceiling and floor heights. These restrictions will give strong priors in the geometrical processing. Taking advantage of the omnidirectional view of non-central panoramas and the unique properties of the non-central projection systems, we extract the 3D information of the structural lines provided by the network. The experiments performed show that our pipeline outperforms the state of the art in layout recovery by a margin. The main contributions are:

- Two new geometrical solvers to obtain the layout of an environment in a Manhattan or Atlanta world assumption for non-central projection systems.
- First work that uses deep learning with non-central projection systems.
- First work of scaled layout recovery without extra measurements from Manhattan and Atlanta world assumptions, handling occlusions, from a single non-central panorama.

5.2 Layouts and non-central systems: what did we have?

Non-central projection systems have been extensively studied from their different acquisition systems Agrawal et al. (2010). Different works have set the fundamentals on catadioptric systems Lopez-Nicolas and Sagues (2014) based on conical Bermudez-Cameo et al. (2018) or spherical Agrawal et al. (2011) mirrors. Other non-central images come from moving cameras, as the push-broom camera Gupta and Hartley (1997) or the non-central panorama Li et al. (2004); Menem and Pajdla (2004). These non-central projection systems present geometrical properties that allow to recover 3D information from single images with geometric reasoning. In particular, several works exploit that a 3D line can be recovered with scale from a single non-central projection Gasparini and Caglioti (2011). The fundamentals for this approach consist in computing the intersection of a line by four generic rays Teller and Hohmeyer (1999). However, although it is theoretically possible to recover the scaled, full 3D reconstruction of a line from a single non-central projection, in practice the results are so sensitive to noise and therefore it is not possible to directly use these approaches with current non-central systems. For this reason, more recent works aim to improve the accuracy of 3D lines fitting by imposing structural constraints. As example, in Bermudez-Cameo et al. (2016) the line extraction is constrained to lines parallel to a known plane, which can be used for extracting horizontal lines from a non-oriented camera by using the gravity direction, for example from an IMU, as prior. Other constraints such as parallel lines or intersecting orthogonal lines are well studied and solved in Bermudez-Cameo et al. (2014), where they present a minimal solution to be included in a robust approach.

When the geometric constraints of structural lines are globally considered, lines and their intersections are enclosed in the concept of layout. The layout of indoor environments provides a strong prior for many computer vision tasks. Several works on virtual or augmented reality Karsch et al. (2011), object recognitionBao et al. (2011); Song and Xiao (2016), segmentation Wang et al. (2021); Zhou et al. (2022) and human pose estimation Fouhey et al. (2014); dos Reis et al. (2021) rely on information of the environment, which is more easily obtainable once the layout is known. Many different methods have been developed in order to recover the layout of a room from different central cameras Jung and Kim (2012). Particularly, in recent years, the use of omnidirectional central images is on the rise, since a single image can provide enough information to make an estimation for a whole room Fukano et al. (2016); Rao et al. (2021); Pintore et al. (2021b). One of the first attempts for layout estimation is proposed by Zhang et al. (2014), which presents an implementation where many 3D layout hypotheses are generated and then ranked by a Support Vector Machine. Then the best ranked hypotheses are selected and compared with the input image to test its validity. More recent approaches take advantage of neural networks to estimate the layouts in a more efficient way. Corners for Layouts (CFL) Fernandez-Labrador et al. (2020) uses an encoder-decoder architecture with convolutions adapted to the spherical distortion of the equirectangular panorama. The output of the network are two heat-maps for the corners and edges that compose the structure of rooms. With a post-processing of this information, an up-to-scale reconstruction of Manhattan environments can be obtained. Other recent approaches combine the convolutional networks with recurrent neural networks, which allow to obtain dependencies along the image, extracting the boundaries of the structural lines. Relying on different geometry constraints, HorizonNet Sun et al. (2019) and AtlantaNet Pintore et al. (2020) obtain a 1D representation of corners, as a probability of having a wall-wall intersection at a certain image column, and other 1D representation of the ceiling-wall and floor-wall intersections which form the structural lines of the room. This minimal representation allows to obtain a more precise approximation of the layout of the room. As in other works, after a post-processing, an up-to-scale layout estimation can be obtained.

In our proposal, we overcome the problem of structural lines extraction in non-central panoramas adapting the neural network of HorizonNet Sun et al. (2019). Then, we propose a new geometrical processing, which takes advantage of two new solvers that fit Manhattan and Atlanta layouts, to recover the 3D layout of indoor environments. Besides, exploiting the properties of non-central projection systems in our geometrical processing, we are able to recover the scale of the 3D layout without extra measurements, which no state-of-the-art method is able to do.

5.3 Scaled layout estimation proposal

Our proposal for layout estimation is a new pipeline composed by two main blocks (see Fig. 5.2). In a first block, we use a neural network to obtain the boundaries of the structural lines of an indoor environment from an image. On the second block, we geometrically process the information provided by the network, exploiting the properties of non-central projection systems and recovering the scaled layout.

With respect to the first block, Bermudez-Cameo et al. (2016) and Zhang et al. (2014) propose geometrical methods based on hypothesis generation-verification to extract lines and layouts respectively while Sun et al. (2019) and Pintore et al. (2020) rely on the use of neural networks for layout recovery. In our proposal we combine both solutions in order to obtain the scaled layout from a single non-central panorama. The use of a neural network allows to obtain the structural lines of an environment faster than with classical approaches of hypothesis generation-verification. In the next section we define

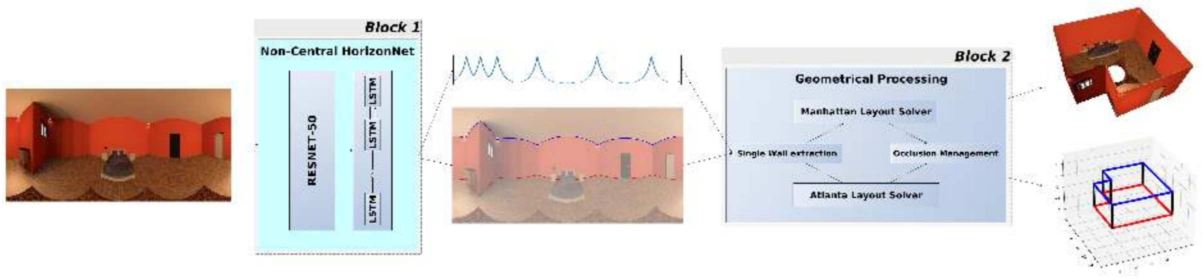


Figure 5.2: Pipeline of our proposal. In a first stage, the neural network extracts the boundaries of the structural lines of the room as well as a probability of corner positions from the non-central panorama. On a second stage, our proposed geometrical processing exploits the properties of non-central projection systems to recover the 3D of the layout from the information provided by the network.

in more detail the network architecture proposed and its advantages and disadvantages. On the other hand, we aim to exploit the geometrical properties of non-central projection systems. In section 5.4 we present in detail the different geometrical solutions proposed to solve the layout recovery problem and the geometrical pipeline proposed to obtain scaled layouts from single panoramas.

5.3.1 Non-central HorizonNet

We propose to adapt an existing neural network in order to obtain the structural lines of indoor environments from non-central circular panoramas (see Fig. 5.3). This architecture is divided in two parts: a convolutional part, formed by the first layers of ResNet-50 He et al. (2016) and a set of convolutions; and a recurrent part, formed by a set of bi-directional LSTMs Schuster and Paliwal (1997). From this architecture, we obtain three 1D arrays with the boundaries information from the image. One of the arrays contains the probability of finding a wall-wall intersection in each column of the image. The other two 1D arrays provide the pixel of the intersection between the ceiling or the floor with the walls. From these three 1D arrays we obtain the boundaries of each of the structural lines that form the layout of the room.

The advantage of this architecture when dealing with non-central panoramas resides in how the network extracts the ceiling-wall and floor-wall intersections: column by column. Due to the bi-directional LSTMs, each column of the image is treated separately in order to recover the structural lines of the room. In our case, where non-central panoramas are used, this property is very interesting since, due to the distributed optical center of this projection system, each column of the image is locally central, allowing the network to work in a central projection system for each separate column. Thus, this architecture proposed for central projection systems fits perfectly and is very suitable to extract the structural lines of a room from a non-central panorama.

HorizonNet imposes some restrictions required to adapt it for non-central panoramas. The main restriction is that the image must be oriented with the vertical direction. It means, that the wall-wall intersections form a straight vertical line in the image. Assuming this restriction, non-central panoramas

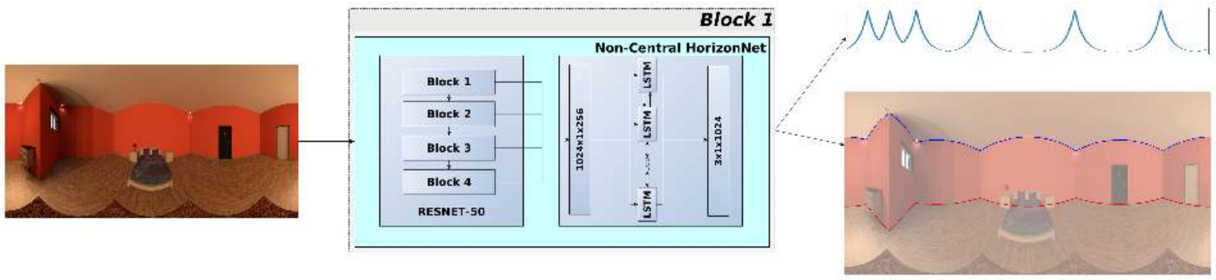


Figure 5.3: The non-central panorama is processed by Non-central HorizonNet, which is an adaptation of the work Sun et al. (2019). First, it goes through a ResNet50, where high and low-level features are extracted. After a set of convolutions, the result is concatenated and fed to an array of bidirectional LSTMs. The network provides the boundaries of the structural lines of ceiling and floor, as well as the corners of the room as three separate 1D arrays.

must be acquired with the revolution axis of the system aligned with the gravity direction. This configuration introduces some disadvantages, since the depth and direction of lines parallel to the axis, the wall-wall intersection in this case, cannot be directly estimated (are one of the degenerated cases mentioned in section 2.2.2). However, since we know the gravity direction and that the structural lines will be perpendicular to it, we can turn the disadvantage into advantage, exploiting this constraint in the geometrical processing to estimate the 3D lines that form the layout.

The original network architecture is trained in PanoContext Zhang et al. (2014) and Stanford 2D-3D Armeni et al. (2017). These datasets are formed by equirectangular panoramas obtained from indoor environments. In our proposal, we start with the network trained on these datasets, since the distortion of equirectangular panoramas and non-central panoramas are similar. Afterwards we add a fine tuning to learn the particular distortion of the non-central panorama. For that purpose, we train the network, starting with the weights presented in Sun et al. (2019), with a dataset formed of non-central panoramas and 3D information of the environment. However, since non-central projection systems are little used, there is no public dataset available. To solve this problem, we have generated and used a dataset of non-central panoramas from synthetic environments to fill this gap in the resources (more details in section 5.5). Once the network has been fine-tuned, it has learned the subtle different distortion of the non-central panoramas, providing more accurate information of the boundaries of the structural lines of the different environments.

5.4 Geometrical processing

The next step in our proposal is to take advantage of the geometrical properties of the non-central panorama in order to recover the 3D layout and the scale of the environment. To do that, we propose a geometrical pipeline, which includes different linear solvers, that takes as input the information provided by the network and outputs the 3D corners of the room. In this section, we present the geometrical problem, defined as a plane extraction problem, and we provide two different solutions

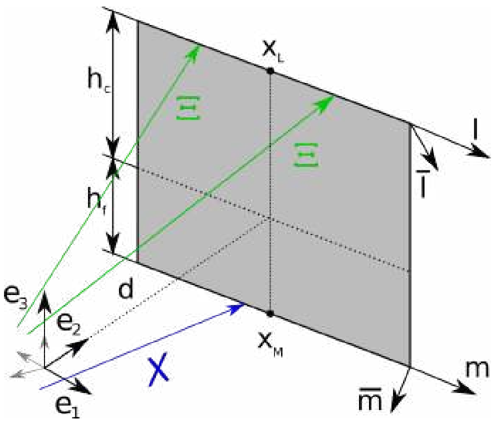


Figure 5.4: Rays and wall parameter definition. The wall reference system is defined as $\{\mathbf{e}_1,\mathbf{e}_2,\mathbf{e}_3\}$; Ξ and X are the projecting rays; $(\mathbf{l},\bar{\mathbf{l}})$ and $(\mathbf{m},\bar{\mathbf{m}})$ are the ceiling and floor lines that define the wall; $\mathbf{x_L}$,$\mathbf{x_M}$ are the closest points of the lines to the reference system; h_c, h_f and d are the distance from the reference system to the ceiling, floor and wall planes respectively.

to jointly obtain the whole layout of a room under different world assumptions. One of the solutions is for Atlanta environments, which are more general and challenging. The second solution is for Manhattan environments, which can be seen as a special case of Atlanta, where we have more geometric restrictions. After these solutions, we present our detailed geometrical pipeline, that includes several steps to make more robust our implementation as well as handling occluded walls in the environment.

5.4.1 Vertical wall extraction

Man-made environments are usually built by vertical walls that intersect the ceiling and the floor in two horizontal parallel straight lines. Thus, we define a wall as a set of two horizontal parallel lines $(\mathbf{L},\mathbf{M})$ contained in a vertical plane (see Fig. 5.4). We define the ceiling line $\mathbf{L} = \left(\mathbf{l}^T,\bar{\mathbf{l}}^T\right)^T$ and the floor line $\mathbf{M} = \left(\mathbf{m}^T,\bar{\mathbf{m}}^T\right)^T$ in Plücker coordinates. We also define an orthonormal reference system placed in the origin and oriented with the vertical wall as $\{\mathbf{e}_1,\mathbf{e}_2,\mathbf{e}_3\}$. From the wall definition shown in Fig. 5.4, the lines direction coincide with the first component of the reference system $\mathbf{l} = \mathbf{m} = \mathbf{e}_1$. To define the momentum vector of the lines with respect to the reference system, we compute the cross product between the closest point of the line to the origin and the line direction as: $\bar{\mathbf{l}} = \mathbf{x_L} \times \mathbf{l}$ and $\bar{\mathbf{m}} = \mathbf{x_M} \times \mathbf{m}$, where $\mathbf{x_L}$ and $\mathbf{x_M}$ are the closest points of the lines to the origin. These points are defined as $\mathbf{x_L} = d\mathbf{e}_2 + h_c\mathbf{e}_3$ and $\mathbf{x_M} = d\mathbf{e}_2 + h_f\mathbf{e}_3$, where h_c, h_f and d refer to the distance from the reference system to the ceiling, floor and wall planes respectively.

$$side(\Xi,\mathbf{L}) = \xi^T\bar{\mathbf{l}} + \bar{\xi}^T\mathbf{l} = \xi^T(h_c\mathbf{e}_2 - d\mathbf{e}_3) + \bar{\xi}^T\mathbf{e}_1 = 0 \tag{5.1}$$

$$side(X,\mathbf{M}) = \chi^T\bar{\mathbf{m}} + \bar{\chi}^T\mathbf{m} = \chi^T\left(h_f\mathbf{e}_2 - d\mathbf{e}_3\right) + \bar{\chi}^T\mathbf{e}_1 = 0 \tag{5.2}$$

To compute the lines that define a wall from an image, we need the projecting rays of these lines.

In our proposal, the neural network provides the pixel information of the boundaries of the projection of structural lines in the environment. From this pixel information, we can compute the projecting rays to the ceiling $\Xi = \left(\xi^T, \bar{\xi}^T\right)^T$ and floor $X = \left(\chi^T, \bar{\chi}^T\right)^T$ lines from the backprojection model seen in Section 2.4. Known the projecting rays, we aim to obtain the 3D lines that define each wall. The relation among the projecting rays and the wall lines is given by their intersection, defined with the side operator in Plücker coordinates (see Section 2.2.2). From our definition of the wall, equations (5.1) and (5.2) define the intersection of the projecting rays and the walls of the layout. Trying to solve directly these equations may be difficult since, in general, is a non-linear problem. Instead, we propose a DLT-like Abdel-Aziz et al. (2015) approach where we compute the solution as a linear problem.

In a first approach, we aim to extract each wall of the layout independently. Let the main direction of a wall be horizontal and described by the vector $\mathbf{u} = (u_x, u_y)^T$ such that $\mathbf{l} = \mathbf{m} = (u_x, u_y, 0)^T$. Then, the orthonormal reference system oriented with the wall can be re-defined as: $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\} = \{(u_x, u_y, 0)^T, (-u_y, u_x, 0)^T, (0,0,1)^T\}$. From this parametrization, equations (5.1) and (5.2) can be written as:

$$\bar{\xi}_1 u_x + \bar{\xi}_2 u_y + (\xi_2 u_x - \xi_1 u_y) h_c - d\xi_3 = 0; \; \bar{\chi}_1 u_x + \bar{\chi}_2 u_y + (\chi_2 u_x - \chi_1 u_y) h_f - d\chi_3 = 0 \tag{5.3}$$

These equations are non-linear since $\mathbf{u}$, h_c and h_f are coupled. At this point, we define the new variables $\mathbf{v} = h_c \mathbf{u}$ and $\mathbf{w} = h_f \mathbf{u}$. From this new variable definition, equations (5.3) become linear, obtaining the following equations:

$$\bar{\xi}_1 u_x + \bar{\xi}_2 u_y - \xi_1 v_y + \xi_2 v_x - d\xi_3 = 0; \; \bar{\chi}_1 u_x - \bar{\chi}_2 u_y - \chi_1 w_y + \chi_2 w_x - d\chi_3 = 0 \tag{5.4}$$

Now, we can build a linear system $\mathsf{A}\mathcal{W} = 0$, where the matrix A is full-filed with relations (5.4) and $\mathcal{W} = (\mathbf{u}^T, \mathbf{v}^T, \mathbf{w}^T, d)^T$ is the unknown wall homogeneous vector. Notice that $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$ are independent variables which can be non-parallel. Since we have defined these vectors as proportional, to impose the parallelism we compute the null space of the system with a Singular Value Decomposition (SVD), obtaining a parametric solution which is a linear combination of singular vectors parametrized with λ_i. Notice that, two horizontal lines contained in a vertical plane have 4 degrees of freedom. At this point we have two options to solve the problem. In one hand, a minimal solution would require 2 projecting rays for each line of the wall, describing the null space with three singular vectors and two parameters λ_1 and λ_2, such as $\mathcal{W} = \mathcal{W}_0 + \lambda_1 \mathcal{W}_1 + \lambda_2 \mathcal{W}_2$. By solving a system of two quadratic equations with action matrices or as a polynomial eigenvalue vector Kukelova et al. (2011), we obtain a set of 4 different solutions which should be discriminated. On the other hand, since the network provides enough robust information of the structural lines, we propose to solve the over-determined case, taking at least 3 rays for each line of the wall.

$$\lambda(\mathbf{v_1} - h_c \mathbf{u_1}) = h_c \mathbf{u_0} - \mathbf{v_0}; \; \lambda(\mathbf{w_1} - h_f \mathbf{u_1}) = h_f \mathbf{u_0} - \mathbf{w_0} \tag{5.5}$$

In this over-determined case, the null space is described by a linear combination involving only one parameter λ, such as $\mathcal{W} = \mathcal{W}_0 + \lambda \mathcal{W}_1$. Imposing the parallelism restriction for **u**, **v** and **w** as shown in equations (5.5), we can derive two uncoupled quadratic equations for λ as:

$$(u_{y0}u_{x1} - u_{x0}u_{y1})\lambda^2 + (u_{x0}v_{y1} + v_{x0}u_{y1} - u_{y0}v_{x1} - v_{y0}u_{x1})\lambda + (v_{y0}v_{x1} - v_{x0}v_{y1}) = 0 \tag{5.6}$$

$$(u_{y0}u_{x1} - u_{x0}u_{y1})\lambda^2 + (u_{x0}v_{y1} + w_{x0}u_{y1} - u_{y0}w_{x1} - w_{y0}u_{x1})\lambda + (w_{y0}v_{x1} - w_{x0}w_{y1}) = 0 \tag{5.7}$$

Computing the solution for λ in each equation, we get four solutions. However, the solutions from equation (5.6) and equation (5.7) are paired, which means that effectively we have only two different solutions for λ. The global orientation prior allows to easily discriminate which of the solutions is the correct one. Computing $\mathcal{W}$ for each λ and extracting the ceiling h_c and floor h_f plane distances, we observe that only one of the solutions sets $h_c > h_f$. Taking the correct value of λ, we have defined the wall direction **u**, the ceiling h_c, floor h_f and wall d planes distance to the acquisition system as well as the Plücker coordinates of the ceiling and floor lines that define the wall.

5.4.2 Manhattan layout solver

Notice that in a Manhattan world assumption, there is a set of walls sharing the wall direction $\mathbf{u} = (u_x, u_y)^T$ and the complementary set of walls share the orthogonal direction $\mathbf{u}_\perp = (-u_y, u_x)^T$. Since we assume that the rooms have single ceiling and floor planes, all the walls share the ceiling h_c and floor h_f heights. Defining the projecting rays of the walls with direction $\mathbf{u}_\perp$ as (Z, Ψ), we can redefine the equations (5.9) this set of walls as:

$$\bar{\zeta}_2 u_x - \bar{\zeta}_1 u_y - \zeta_2 v_y - \zeta_1 v_x - d_i \zeta_3 = 0;\ \bar{\psi}_2 u_x - \bar{\psi}_1 u_y - \psi_2 w_y - \psi_1 w_x - d_i \psi_3 = 0 \tag{5.8}$$

Then, we can extend the DLT-like approach to fit all the set of walls computing the null space of $\mathsf{A}\mathcal{L}_M = 0$, where $\mathcal{L}_M = (\mathbf{u}^T, \mathbf{v}^T, \mathbf{w}^T, d_1, \cdots, d_N)^T$, where N is the number of walls, and the matrix A is full-filed with relations (5.4) for a set of walls and (5.8) for the other. This approach allows to jointly obtain the main directions in the Manhattan world assumption, the height of the room and the walls locations.

5.4.3 Atlanta layout solver

In the case of Atlanta world assumption, each wall of the room could have a different horizontal direction, therefore we must find a new approach to obtain the whole layout. Notice that from the previously proposed solver, we can extract each wall independently. However, with this approach we do not impose that the walls share the ceiling and floor heights. Nevertheless, if the direction of each wall is known (e.g. extracting each wall independently), we can derive a new linear solution for the whole layout.

$$\bar{\xi}_1{}' + h_c \xi_2' - d \xi_3' = 0;\ \bar{\chi}_1{}' + h_f \chi_2' - d \chi_3' = 0 \tag{5.9}$$

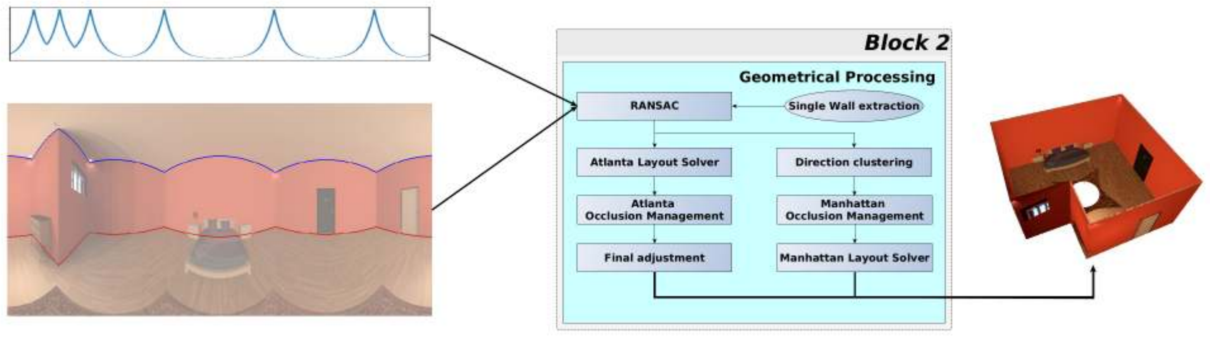


Figure 5.5: Geometrical pipeline. The input is the pixel information provided by the network. First, a RANSAC makes a first wall direction estimation. Then the pipeline branches. For Manhattan world assumption, we cluster the walls direction, then handle the possible occlusions and compute the direction labels of the walls. Finally we compute the Manhattan layout with the proposed solver. For Atlanta world assumption, once defined the walls directions, we implement the proposed layout solver. Then we search for occlusions. As last step, a final adjustment of the corners is made to obtain the final 3D layout.

Making a first independent wall extraction, we obtain the direction for each wall. Changing the reference system of the projecting rays from the acquisition system to each wall local reference system, equations (5.1) and (5.2) become the linear expressions (5.9), where Ξ' and X' are the projecting rays in each wall reference system. Then we can solve the null space of the linear system $\mathsf{A}\mathcal{L}_A = 0$, where A is a matrix full-filed with equations (5.9) and $\mathcal{L}_A = (1, h_c, h_f, d_1, \cdots, d_N)^T$, where N is the number of walls in the environment. In this approach, once known the walls directions, we can simultaneously compute the room height and the walls location.

5.4.4 Detailed geometric pipeline

In order to improve the robustness of our method, we propose a new full geometric pipeline that includes the two new solvers presented before (see Fig. 5.5). This pipeline takes as input the information provided by the network and gives as output the 3D lines and corners that form the layout. This geometric pipeline is divided in two branches, one for Manhattan world assumption and other for Atlanta world. This is due to the different management of the occlusions in each world assumption.

The geometric pipeline starts with a RANSAC algorithm to filter possible spurious data. Here could rise a question: What is the advantage of using these structural deep learning based edges and corners over classic Canny edges if we still have to use a RANSAC approach? The main advantage is the huge reduction of the number of required hypotheses. Consider the number of hypotheses in a RANSAC approach n_{hyp} which is typically estimated by $n_{hyp} = \frac{\log(1-P)}{\log(1-(1-\varepsilon)^k)}$, where P is the probability of not failing in the random search, ε is the rate of outliers and k the number of elements defining the hypothesis. Assuming that we want to extract a wall, with a probability of $P = 99.99\%$ of not failing, we need two lines, defined by 3 points each. We assume a rate of outliers of $\varepsilon = 20\%$

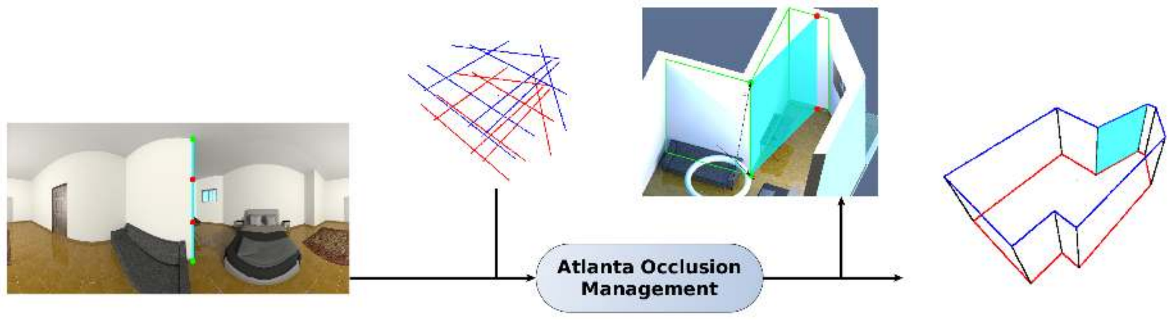


Figure 5.6: Atlanta occlusion management. We generate a new wall (in blue) which is co-planar with the projecting rays Ξ and X corresponding to the visible corner (dots in green). This new wall defines two new corners (dots in red) in a partially occluded wall. The green wireframe is the ground truth layout.

in the input data. With our method, we have well defined which data belong to the ceiling line and which to the floor line and the data of each wall separately. So, we assume that our $\varepsilon = 20\%$ and that we only need $k = 3$ samples to define the two lines, since data is defined by column and we can take one value for each line in each column. This computation leaves that our implementation needs $n_{hyp} = 12{,}84$ hypotheses to define the best wall that fits the data. With state-of-the-art approaches as Bermudez-Cameo et al. (2016, 2017), where an edge detector is used, as Canny, the data is not as well defined and structured. Thus, the probability of a sample been an inlier is reduced by half, since it can be part of the ceiling or the floor. This also means that we need the double of samples, since they are not paired. This assumptions lead to a number of outliers of $\varepsilon = 60\%$ and a number of samples $k = 6$, obtaining $n_{hyp} = 2244$ hypotheses per wall. This difference in the number of hypotheses is reduced by the neural network. Thus, with the solution for a single wall presented in section 5.4.1 as hypothesis in the RANSAC, we get the 3D lines that better fit the information provided by the network. After this step, the pipeline branches depending on the world assumption.

Assuming a Manhattan world. We cluster the extracted walls into two classes corresponding to the main perpendicular directions. In this clustering, we label each wall with the index of one of the clusters corresponding to different perpendicular Manhattan directions. In a second step, we manage the occlusions in the Manhattan environment. Since a Manhattan world only have two main directions, these must be alternating in consecutive walls. If two consecutive walls have the same direction means that an occluded wall is between them. In this case, we add a perpendicular wall between the occluded and occluder walls to keep the alternation in the walls direction. Finally, once we know the number of walls and their Manhattan direction label (defined as $\mathbf{u}$ and $\mathbf{u}_{\perp}$ in Section 5.4.2) we apply the Manhattan layout solver. This solver will provide the walls direction that better fit the whole environment as well as the height of the room and the walls location. Once obtained the lines that define the walls, we can obtain the 3D corners of the room computing the intersection of these lines, which is easy since ceiling lines are co-planar, as well as floor lines.

Assuming an Atlanta world. We do not know the number of dominant directions in an Atlanta environment. Besides, we cannot find an occlusion looking at the walls direction, since two consecutive walls may have a similar orientation. Thus, Atlanta environments must be tackled in a different way than the Manhattan ones.

Since Atlanta environments do not have a defined number of main directions, we cannot cluster the walls extracted in the RANSAC by their direction. By contrast, we consider this first wall direction estimation good enough and use it as initial value in our pipeline. Then, assuming that the walls direction is known, we can apply the solution for Atlanta environments presented in section 5.4.3, where we jointly obtain the room height and the walls locations.

Once defined the lines that form the walls of the environment, we compute the 3D corners. Notice that if we compute the corners as lines intersection, we may make impossible layouts if there is an occlusion in the environment. To avoid this problem, we compute each corner as the point of the computed 3D line that minimizes the Euclidean metric distance in $\mathbb{E}^3$ Bermudez-Cameo et al. (2017) between the 3D line and the projecting ray corresponding to the corner provided by the network. Each projecting ray crosses with two 3D lines corresponding to two consecutive walls where two different cases may appear. When the computed 3D corners coincide in a single point, no occluded wall is detected and this point is a corner of the environment. By contrast, if the computed corners are different 3D points (i.e. the distance between them is higher than a threshold), we have found an occluded wall between the two 3D lines and we insert a wall in the layout model (see Fig. 5.6). Since in an Atlanta world assumption there is no restriction about the walls direction, we assume that the occluded wall and the projecting ray of the corner lie in the same plane.

Finally, we make a final adjustment where we fine tune the 3D lines direction and position. We minimize the least-square reprojecting error of the computed 3D lines with the pixel coordinates of the boundaries Bermudez-Cameo et al. (2017) provided by the network. This final adjustment refines the position of the 3D corners. The movement of the corners out of the plane defined by the projecting rays of the corners is penalized. This extra soft-restriction allows managing the infinite possible solutions caused by occlusions during the optimization step.

5.5 Non-central panorama dataset

Currently, we can find a great amount of image datasets, from perspective images Russakovsky et al. (2015) to omnidirectional panoramas Armeni et al. (2017); Chang et al. (2017). However, non-central projection systems have never been used with deep learning before, thus there is not a dataset of this kind of images. This is a big problem in our case, since we need a large amount of images to train a deep learning architecture. So, we present a new dataset obtained with our synthetic generator of realistic non-central panoramas. It includes semantic, depth and 3D information of the environment. More detailed information of the dataset can also be found in the Appendix A.

We generate random layouts, from 4 to 14 walls, in a Manhattan world assumption. Then, with a probability set by the user, corners of these layouts are clipped and substituted by oblique walls to generate Atlanta world layouts. Once obtained the structure of the room, we compute the free space to place objects in it. We have two kinds of objects: those that are placed next to a wall in a fixed orientation (beds, desks, wardrobes, TVs) and those that are placed in the middle of the room at any orientation (chairs, sofas, carpets). These objects are taken from different pools where one is chosen randomly and placed in the room if has a free space in it. Additionally, the ceiling, floor and walls materials and color are taken from different pools and chosen randomly for each new environment. After the virtual environment is generated, we set the illumination conditions. We have different pre-defined ambient illuminations and we also randomly place spot lights to give the environment a more realistic view.

Once defined the virtual environment, we use POV-Ray POV-Ray (2020) to render RGB and semantic images and MegaPOV MegaPOV (2020) for depth maps. These images are generated by a ray tracing method, which can follow ad-hoc programmable camera projection models. In our case, we use the projection model presented in Section 2.4 for non-central circular panoramas. The center of the acquisition system is placed in a random position for each room generated, obtaining a greater variability in the walls distortion along the dataset. This allows to generate a set of images, from different rooms and in different positions, not only in the center of the room as many existing datasets.

For this work, we have generated a dataset of non-central panoramas to fine-tune the deep learning architecture presented in section 5.3.1. The dataset[1] is formed by more than **2500** images, taken from different positions inside the environments, from around **650** different rooms, from 4 to 14 walls, combining Manhattan and Atlanta environments. We propose a division of the dataset in three blocks: training set, formed by 1677 panoramas; validation set, formed by 399 panoramas; and test set, formed by 499 panoramas. Each set includes Atlanta and Manhattan environments of different number of walls, and we make sure that there are no equal layouts in different sets.

5.6 Experiments

The pipeline of our proposal is divided into two main blocks: a neural network that extracts the boundaries of the structural lines of an indoor environment from a single panorama and a geometrical processing that takes as input the output of the network and recovers the 3D scaled layout. In order to evaluate our proposal, we have performed a set of experiments. We independently evaluate the performance of both main blocks: the proposed neural network and geometric pipeline. Additionally, we make a comparison with state of the art methods for line estimation in non-central panoramas and for layout recovery from single panoramas.

[1] github.com/jesusbermudezcameo/NonCentralIndoorDataset

Table 5.1: Evaluation and comparison of different state-of-the-art methods for layout recovery. We compare HorizonNet Sun et al. (2019) (HN), Corners for layouts Fernandez-Labrador et al. (2020) (CFL) and LayoutNet v2 Zou et al. (2021) (LNv2) with the weights provided in their works (Base Line) and after a fine tuning with non-central panoramas (Fine Tuning). The metrics in bold represent the best result (higher is better). All networks are tested in the test partition of the dataset proposed in section 5.5.

		Base Line			Fine Tuning		
		HN	CFL	LNv2	HN	CFL	LNv2
Corners	P	**0.379**	0.298	0.352	**0.806**	0.644	0.457
	Acc	**0.995**	0.994	0.985	**0.998**	0.997	0.984
	R	0.204	**0.236**	0.160	0.792	0.543	**0.817**
	IoU	**0.154**	0.150	0.133	**0.643**	0.418	0.400
Edges	P	0.068	**0.085**	0.064	**0.476**	0.231	0.115
	Acc	**0.983**	**0.983**	0.945	**0.995**	0.991	0.951
	R	0.153	**0.202**	0.084	**0.544**	0.373	0.158
	IoU	0.047	**0.060**	0.036	**0.382**	0.166	0.068

5.6.1 Ablation study: non-central HorizonNet

In this work, we have adapted an existing neural network to work with non-central panoramas, fine-tuning it with the dataset proposed in the previous section. We take the weights for the network that minimize the validation error during the training to perform the different experiments.

We evaluate and compare different state-of-the-art networks for layout extraction in order to verify our selection. The different networks have been evaluated before and after a fine tuning with our proposed dataset of non-central panoramas. We have considered HorizonNet Sun et al. (2019), Corners for Layouts Fernandez-Labrador et al. (2020) and LayoutNetv2 Zou et al. (2021).

To evaluate the performance of boundary and corner extractors, we use similar metrics as in Fernandez-Labrador et al. (2020); Zou et al. (2021). We compare the probability maps of the output of the network with their respective labels (each network provides different output resolutions and different probability maps). We try to make the comparison of methods as fair as possible, evaluating the outputs in a similar way and with their respective parameters for the label generation. In order to include HorizonNet in this comparison, we generate the probability map of edges and corners from the output of the network (which are 1D arrays with pixel information) in a similar way than Zou et al. (2021); Fernandez-Labrador et al. (2020). The metrics defined for the evaluation are: Precision (P), Recall (R), Accuracy (Acc) and Intersection over Union (IoU). In Table 5.1 we present the results of this comparison.

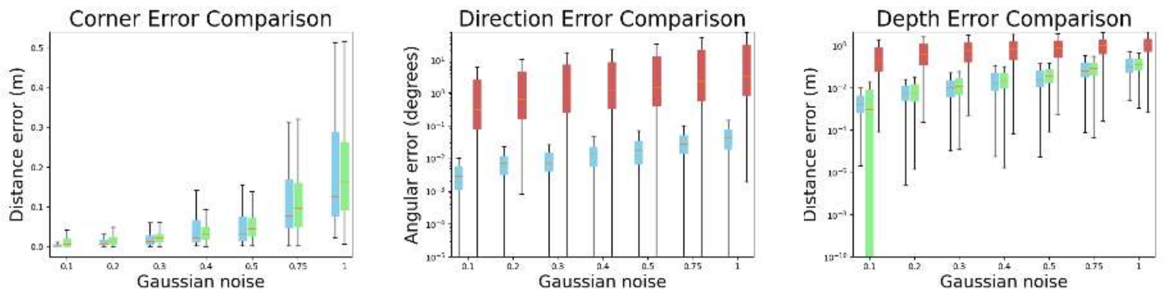


Figure 5.7: Experimental results to evaluate and compare our proposed Manhattan and Atlanta layout solvers against a state-of-the-art method for line extraction Bermudez-Cameo et al. (2016). In blue are the Manhattan solver results. In green are the Atlanta solver results. In red are the state of the art method results. (a) Corner error, in meters, of our proposed solvers. (b) comparison of Direction error, in degrees and logarithmic scale. Notice that the direction error of the Atlanta solver is zero since we provide the lines direction as prior. (c) comparison of Depth error, in meters and logarithmic scale.

5.6.2 Ablation study: geometric solvers

We evaluate the proposed solvers presented in Section 5.4. To do so, we use the ground truth information of the test partition of the dataset to evaluate the sensitivity to noise of the geometric solvers for line extraction. We compare our results with the state of the art method for line extraction in non-central panoramas Bermudez-Cameo et al. (2016). Since we are focusing on the geometric approach, we omit the environments with occlusions.

As input information, we use the projecting rays of the boundaries of the structural lines of the indoor environment, taken with sub-pixel accuracy from the ground truth information of the dataset. To evaluate the sensitivity to noise of the solvers, we add increasing Gaussian noise to the ground truth projecting rays at sub-pixel level. For the evaluation of our Manhattan layout solver, we use the walls direction to label the walls in the two main Manhattan directions. For the Atlanta layout solver, we need these wall directions to compute the projecting rays in the wall reference system. The method proposed in Bermudez-Cameo et al. (2016) compute the lines which are parallel to a known plane, which in our case is the horizontal plane.

To make the evaluation and comparison, we use the same metrics defined in Bermudez-Cameo et al. (2016). Computed a 3D line $\mathbf{L} = (\mathbf{l}, \bar{\mathbf{l}})^T$, we compute the direction error as: $\varepsilon_{dir} = \arccos(\mathbf{l}^T \cdot \mathbf{l}_{gt})$, measured in degrees. We also compute the depth error of the line as: $\varepsilon_{depth} = |||\bar{\mathbf{l}}|| - ||\bar{\mathbf{l}}_{gt}|||$, measured in meters. Additionally, we use a common metric in layout recovery works, the corner error (CE). We compute the corners of the layout as line intersections and compute the L2 distance to the ground truth corners to define the metric.

The evaluation and comparison of our methods with the proposed in Bermudez-Cameo et al. (2016) is shown in Figure 5.7. In blue is shown our Manhattan solver, in green is shown our Atlanta solver while in red is shown the method presented in Bermudez-Cameo et al. (2016).

Table 5.2: Comparison of layout recovery with only the solvers or the full proposed geometric pipeline.

		Network labels		Network predictions	
	World assumption	CE (m)	IoU (%)	CE (m)	IoU (%)
Only Solvers	Manhattan	0.0297	98.0036	0.5569	81.7364
	Atlanta	0.8575	57.3289	1.2860	42.6701
Full Pipeline	Manhattan	0.0218	98.4753	0.2109	86.8104
	Atlanta	0.1391	92.5012	0.4811	76.0218

5.6.3 Ablation study: full pipeline

We analyze the performance of the geometric pipeline proposed in section 5.4.4 against the use of only the geometric solvers from sections 5.4.2 and 5.4.3. The question is: if we have two geometric solvers that obtains the whole layout, why we need a more complex geometric pipeline? The answer comes in two parts. First, the geometrical solvers are not able to handle possible occlusion in the environment. Second, we have observed that the input information to the geometric block of our full pipeline (Fig 5.2) can be noisy and with spurious data. As seen in previous section 5.6.2, the geometrical solvers work well with refined data, however they may lead to impossible layouts if the input is too noisy or if the fitting is corrupted by spurious information.

We test how much the performance of our method improves with and without the proposed geometric pipeline. For that purpose, we compute the corners of the test-set layouts of our dataset in two different cases. In the first case, we use as input of our geometric block the labels of the network ('Network labels' in table 5.2), that means, the best information that the network would be able to provide. On a second case, we use as input of the geometric block the predictions provided by the network ('Network predictions' in table 5.2), which are noisier than the labels. The metrics used to make the comparison are: Corner Error (CE), defined as the L2 distance between the computed corners and the ground truth; and the Intersection over Union (IoU) of the volumes of the computed layout and the ground truth. In this experiment, 'Solvers' refer to the use of only the geometric solvers presented in section 5.4. We use the wall extractor (see section 5.4.1) to compute the wall labels in the Manhattan case and the walls' direction in the Atlanta case, and then the layout solver to obtain the corners of the room. 'Pipeline' refers to the use of the whole geometric pipeline proposed in section 5.4.4. In order to make a more fair comparison, environments with occlusions have not been taken into account to compute the metrics (if so, the performance difference would be much greater). Table 5.2 shows the results of this experiment.

Table 5.3: Comparison of methods: fine tuned HorizonNet Sun et al. (2019) (HorizonNet FT) and our method. Evaluation made in selected Manhattan environments from the test-set of the presented dataset. HorizonNet FT uses the camera height for computing the scale while our proposal does not use extra measurements to compute the 3D layout.

	Scale assumption	CE	IoU (%)
HorizonNet FT	No-scale	0.3380	82.6742
	Metric	0.3504	82.6742
Ours	No-scale	0.2024	87.6208
	Metric	0.2271	87.6208

Table 5.4: Comparison of different state-of-the-art methods for 3D layout recovery. Notice that other methods are evaluated on equirectangular panoramas while our proposal is evaluated with non-central panoramas.

	Manhattan World assumption			
	3D IoU (u2s)	3D IoU	CEN	CE
CFL Fernandez-Labrador et al. (2020)	78.87	-	0.75	-
HorizonNet Sun et al. (2019)	82.66	-	**0.69**	-
AtlantaNet Pintore et al. (2020)	83.94	-	0.71	-
Ours	**93.87**	**86.16**	0.78	**0.223**
	Atlanta World assumption			
HorizonNet Sun et al. (2019)	73.53	-	-	-
AtlantaNet Pintore et al. (2020)	80.01	-	-	-
Ours	**90.46**	**76.02**	**1.5**	**0.481**
	higher is better		*smaller is better*	

Table 5.5: Comparison of different state-of-the-art methods for 3D layout recovery evaluated in the Stanford 2D3DS Armeni et al. (2017) dataset. Notice that other methods are evaluated on equirectangular panoramas while our proposal is evaluated with non-central panoramas.

	3D IoU (u2s)	3D IoU	CEN	CE
CFL	65.23	-	1.44	-
HorizonNet	83.51	-	**0.62**	-
LayoutNet v2	82.66	-	0.66	-
AtlantaNet	83.94	-	0.71	-
Ours	**88.19**	**58.19**	2.02	**0.878**
	higher is better		*smaller is better*	

5.6.4 State of the art comparison

We have performed three different tests to compare our work with the state-of-the-art methods for layout recovery from a single panorama. However, since our method is the first to recover layouts from non-central panoramas, a direct comparison is difficult to make and may not be totally fair. In a first experiment, we have fine-tuned HorizonNet Sun et al. (2019) with equirectangular panoramas of the same virtual environments of our dataset. On a second experiment, we compare the results of different state-of-the-art works with ours in Manhattan and Atlanta environments from different datasets. Finally, on the third experiment we compare our proposal with state-of-the-art methods on the Stanford2D3DS Armeni et al. (2017) dataset, labeled as cuboid rooms. In the Figure 5.8 we present qualitative results of our proposal in the two datasets evaluated.

For our first experiment, we have generated a dual test-set, with equirectangular and non-central panoramas taken in the same position in the same Manhattan environments. We use this set-up to compare two different methods and two different acquisition systems in the same test-set. Then, we recover the layout with HorizonNet, from central panoramas, and with our method, from non-central panoramas, for Manhattan environments. Notice that HorizonNet does not obtain the scale of the layout. Instead, it assumes a camera height and computes the 3D corners with this extra measurement. Our method does extract the scale of the environment, so this measurement is computed and not given. To take this into account, we compare the metrics in two different cases. In a first case (No-scale), we normalize the layouts, predicted and ground truth, with the camera height. In the second case (Metric), we use the real camera height to compute the scale in the prediction from HorizonNet. In the case of our proposal, the scale is computed directly from the image, without any extra measurement. Table 5.3 shows the results of this comparison. The metrics used are the Corner Error (CE) and the Intersection over Union (IoU) defined before.

On a second experiment, we compare our method with other state-of-the-art methods. This second comparison is not as fair as the previous one since each method has been trained and tested on different datasets, so the results can depend on the dataset used and not only on the method. The metrics used for the comparison are: 3D IoU, which refer to the 3D intersection over union of the predicted layout and the ground truth; 3D IoU(u2s), which refer to the up-to-scale intersection over union of the layout; CEN, which refer to the Corner Error Normalized computed as the L2 distance of the corners divided by the diagonal of the layout's bounding box; CE, which refers to the Corner Error computed as the L2 distance of the corners in meters. Table 5.4 shows the results of this experiment.

The third proposed experiment aims to compare the performance of different state-of-the-art methods for layout recovery in the same dataset. In this case, we choose Stanford 2D3DS Armeni et al. (2017), where the layouts are labelled as cuboids. For this experiment, we have generated a set of non-central panoramas from the color and depth information of the dataset. With the new dataset we make a fine tuning of our network and evaluate our proposal. We have observed that some images present glitches and blank spaces that lead our algorithm to failures where no layout can be recovered ($\sim$2.5% of the test-set). We present the results of this experiment in Table 5.5 considering the cases where the layout can be recovered.

5.6.5 Qualitative experiments

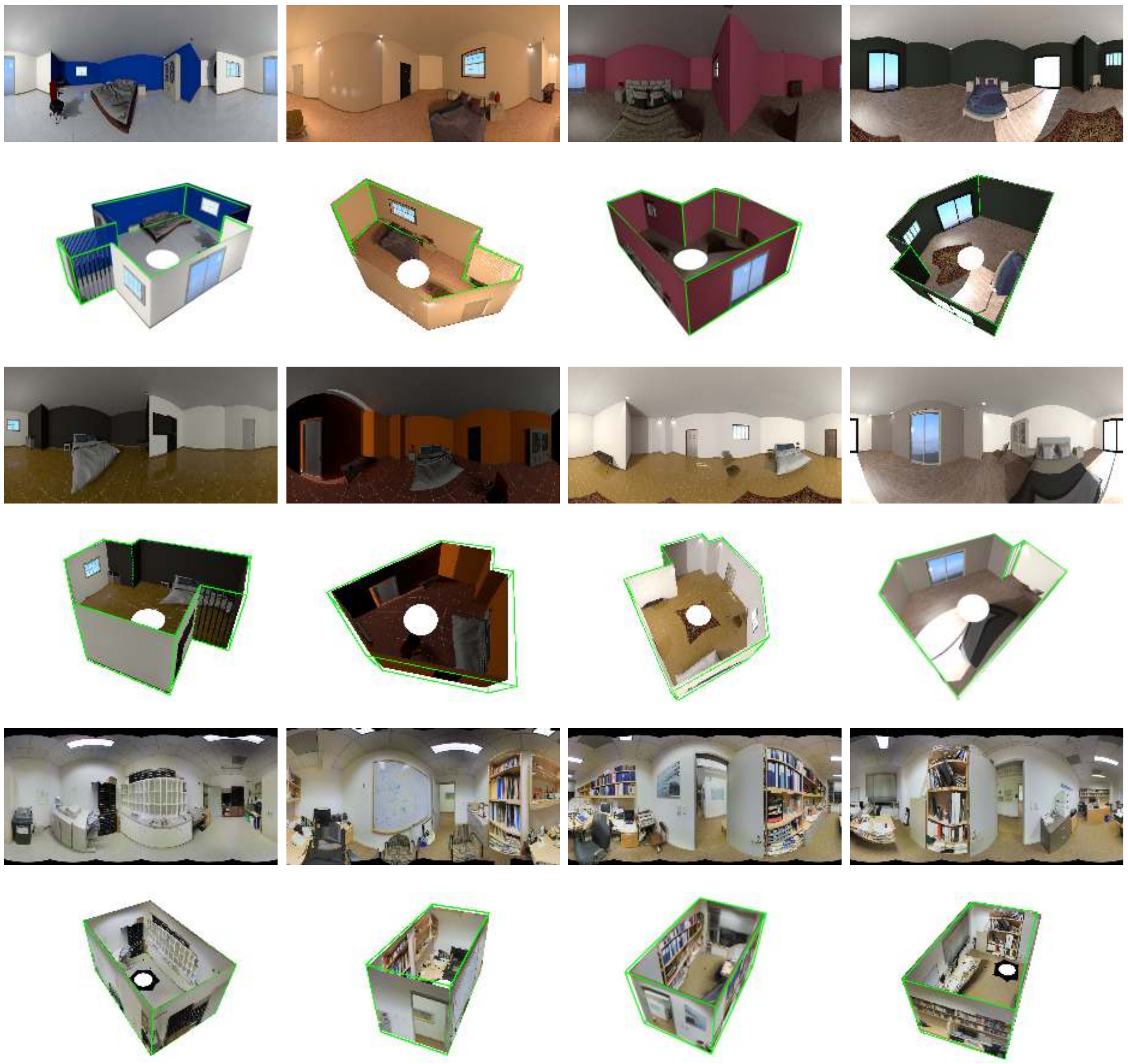


Figure 5.8: Qualitative demonstration of the proposed scaled layout recovery with synthetic images (first and second rows) and real images adapted from the Stanford dataset (last row). As qualitative evaluation, in green is a wire-frame of the room layout.

In this section we present different examples of real non-central panoramas and the scaled layout recovery with our pipeline. The acquisition and annotation of these real examples has been made manually by the authors.

Figure 5.9 shows the real non-central panoramas and their corresponding reconstruction using our pipeline. These panoramas have different calibrations and have been taken in different environments and also in different illumination conditions. Besides, due to the acquisition system, these images are

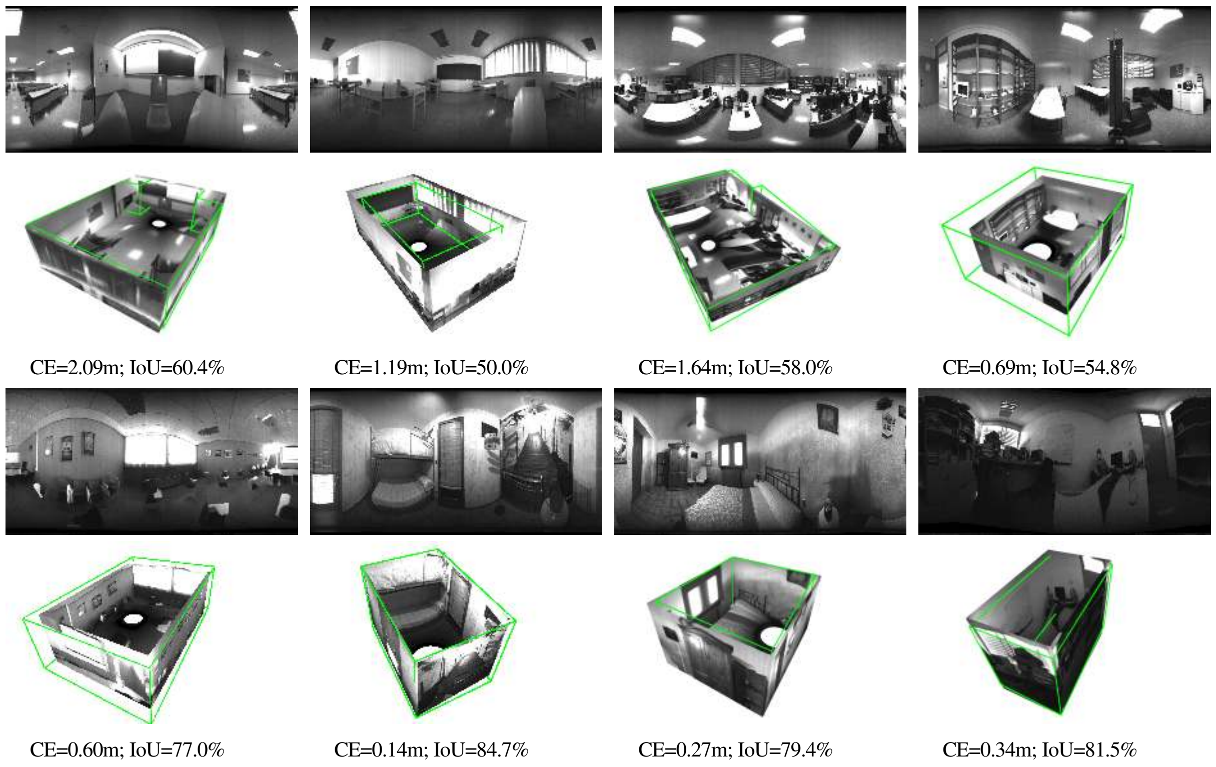


Figure 5.9: Qualitative and quantitative demonstration of the proposed scaled layout recovery with real images. As qualitative evaluation, in green is a wire-frame of the room layout.

grey-scale, which provide less information to the network. To solve this problem, we have trained the network on grey-scale panoramas, creating a second set of weights for this gray-scale version of the network. As a qualitative demonstration, we show a green wireframe that represents the real layout of each environment, measured with a laser meter and reconstructed and aligned with the results. We have computed the same metrics as in other experiments, obtaining an average corner error of CE $= 1.01m$ and an average 3D intersection over union of IoU $= 65.57\%$. We also present the corner error and intersection over union of each image in the Figure 5.9.

5.7 Discussion

Sections 5.6.1, 5.6.2 and 5.6.3 evaluate different parts of our pipeline. In the first one, we have compared three state of the art networks for layout recovery. This experiment supports our initial intuition and we can claim that HorizonNet Sun et al. (2019) is our best option among the evaluated networks for boundary extraction. Even though before the fine tuning it is not the best option, the architecture of HorizonNet is really suitable for structural line extraction in non-central panoramas and provides the best performance after fine tuning. In 5.6.2 we compare our new layout solvers against a state-of-the-art line extractor for non-central panoramas. We observe that our approach outperforms

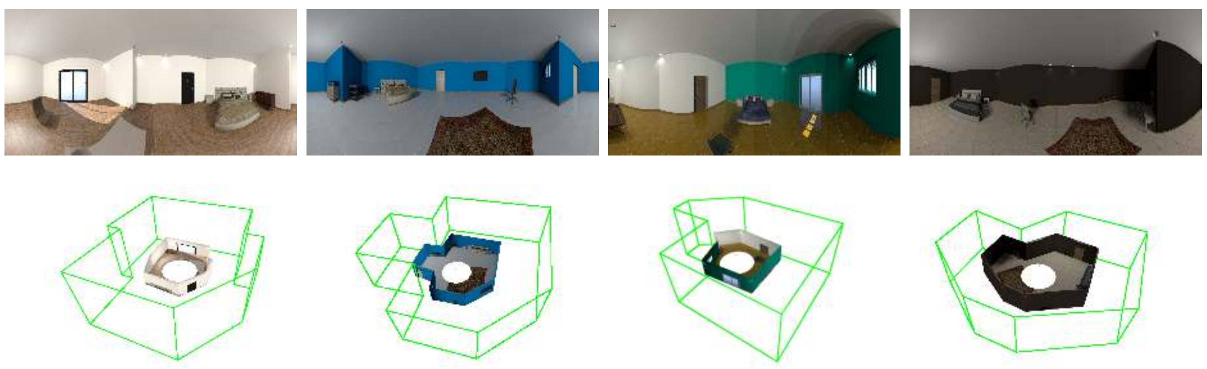


Figure 5.10: We present cases where our method fails in the scale recovery.

previous methods for line extraction in both Manhattan and Atlanta world assumptions by a large margin. Finally, the evaluation of our geometric block validates the use of a more complex geometric pipeline. From the results presented in Table 5.2, we observe that the difference in performance between the solvers and the pipeline using the network labels in Manhattan environments is quite small. However, for Atlanta environments and with the use of the network predictions, the performance of the full geometric pipeline is significantly better.

In section 5.6.4 we compare our method with different state of the art implementations for layout recovery. In a first experiment, we make a comparison with HorizonNet. This is the most fair comparison made since we use panoramas from the same virtual environments and same locations on both methods in order to recover their layout. The results show that our proposal outperforms HorizonNet in the two cases of study: with extra measurements for HorizonNet and in up-to-scale reconstructions. On the second experiment, we make a comparison with other state of the art methods. In this experiment, the datasets used for the experiments are different, so the results do not completely show the performance of each method. Nevertheless, making the comparison of the results of each method, our proposal presents a better performance in most of the metrics. Besides, we are able to recover the scale of the environment without extra measurements while other methods need a metric measure to scale the layout (e.g the camera or room height). Furthermore, we want to highlight the results in Atlanta environments. State-of-the-art methods do not refer in their works the management of occlusions in Atlanta environments, although they do manage occlusion in Manhattan world. Our proposal does handle occlusions in Atlanta environments, as well as in Manhattan environments. On the third experiment, we compare different state-of-the-art methods for cuboid layout extraction. Even though our metrics drop significantly against the results shown from our dataset, we still outperform state-of-the-art methods in the up to scale intersection over union. This drop in performance can be explained by the specialization into cuboid layouts of the dataset labelling. Looking for a more general solution may reduce the performance on more specific tasks. Our proposal aims to obtain the layout from a greater variability of rooms (our synthetic dataset has rooms from 4 to 14 walls) which means that our algorithm is not as particularized to cuboids as other state of the art methods.

As a qualitative demonstration, in section 5.6.5 we present some examples of layout recovery from real images, taken by the authors. We observe that the performance varies from environment to environment. We are able to recover the layout of most of the environments and, in several cases, we achieve a good 3D scaled reconstruction of the environment. The performance in the 3D reconstruction is limited due to the noisy output of the network when dealing with these real images. This is the effect of the lack of real images for training the network and the artefacts that these real images present in the acquisition process.

In the figure 5.10 we show cases where the scale recovery of the environment fails. Analysing those cases, we found different possible sources of error. One of these sources is the information provided by the network. When the images present environments which are too cluttered, occluding practically any boundary of the floor-wall intersection, the output of the network is not very accurate estimating the floor boundary. This lead to a worse 3D reconstruction. Besides, the artefacts in real images fool the network making it believe that there are more wall-wall boundaries than there really are. Other source of error is the occlusion management in Atlanta environments. We found out that an incorrect definition of the 3D corners of the room when dealing with an occlusion lead the final adjustment to under-estimate the scale of the room. This seem logical since the closer the 3D points are to the acquisition reference system, the lower the reprojection error will be. A final source of error is the effective baseline of the non-central acquisition system. This problem is also more evident in the real images. With a smaller effective baseline, related with the radius of the non-central panorama, the accuracy to compute the scale of the room is reduced.

5.8 Conclusions

We have proposed a new pipeline that completely solves the layout recovery problem from a single image (i.e. reconstructing Manhattan and Atlanta environments with scale). We have presented the first application of non-central panoramas that is comparable with state of the art methods for layout recovery, even improving in some of the metrics. We introduce the first indoor dataset of non-central panoramas automatically generated. This dataset provides a good resource for many researchers to further investigate the geometrical properties of the non-central projection system.

The presented experiments show that our proposal can achieve great results. However, there is still room to grow when real images enter into action. Since real images are hard to obtain, the current approach cannot handle these non-central panoramas as well as with the synthetically generated. Nevertheless, the results are promising and may encourage the development of commercial devices able to obtain non-central panoramas.

Chapter 6

Panoramic image stabilization

The use of omnidirectional devices on autonomous systems provides several advantages for localization and mapping. However, most algorithms to process the information require that the images are in a specific orientation to take advantage of prior knowledge. In this chapter we present a visual gyroscope based on equirectangular panoramas. We propose a new pipeline where we take advantage of combining three different methods to obtain a robust and accurate estimation of the attitude of the camera. With the combination of deep learning and photometric algorithms, our pipeline stabilizes a panoramic image with respect a desired reference.

6.1 Introduction to visual gyroscopes

Autonomous navigation of Unmanned Aerial Vehicles (UAV) require the knowledge of pose and orientation. Among the different sensors we can use, cameras provide information of the surroundings of the UAV which can be used for several applications, such as object detection Jiang et al. (2022), while being a lightweight sensor. One of these applications is the use of the camera as a visual gyroscope, estimating the 3D orientation of the camera for a given image with respect to a reference.

Visual gyroscope algorithms consider two kinds of information, direct or indirect. Indirect Visual Gyroscopes leverage image features, either handcrafted, as patches around image points Hadj-Abdelkader et al. (2018), or learned, e.g. estimating first the optical flow and then the 3D orientation of the camera Kim et al. (2021). Instead, Direct Visual Gyroscopes consider pixel brightness of the whole image as input of a 3D rotation optimization method André and Caron (2022). Usually, this pixel information is transformed into different domains before estimating the orientation, such as spherical Fourier transform Makadia and Daniilidis (2006) or Mixture of photometric potentials Caron and Morbidi (2018).

Previous methods present high accuracy on 3D orientation estimation on narrow domains André and Caron (2022), or the possibility of a wide estimation domain but with lower accuracy Makadia and Daniilidis (2006). In this chapter, we propose a hybrid pipeline combining three different methods

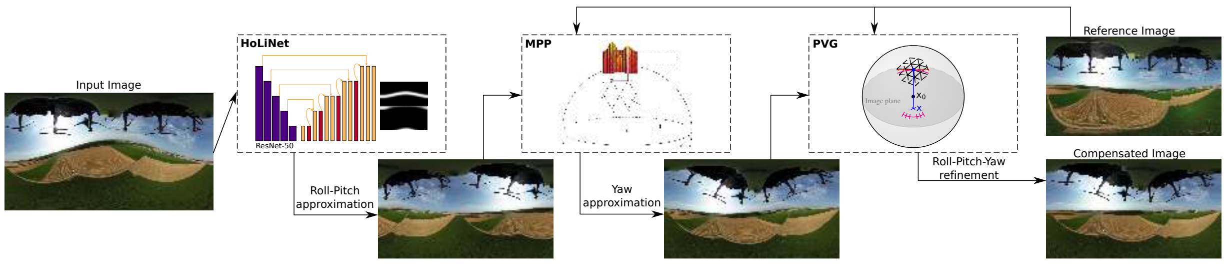


Figure 6.1: Overview of our proposed pipeline. The input equirectangular panorama first goes through HoLiNet, obtaining a first approximation of Roll-Pitch angles with respect to the horizontal plane. The output is feed to the MPP, obtaining an approximation of the Yaw angle with respect to a reference image. Finally, the PVG refines the three angles such that the rotation compensated image is closer to the reference one.

to obtain a robust visual gyroscope within a very large estimation domain and with high orientation accuracy. These three methods work in harmony, leveraging their strengths to overcome the weaknesses of the other methods. From the neural network, we obtain a fast result for two angles without any prior knowledge while with the first photometric method we obtain the third angle with respect a reference image. These two methods have a wide convergence domain, which allows the last method have a good initial guess for a refinement of the orientation estimation. Without this initial guess and with a narrower convergence domain, the third method would not be able to obtain such performance.

6.2 Visual gyroscope

Our proposed pipeline[1] is composed by three different methods that act sequentially to obtain the Roll-Pitch-Yaw angles of the camera in outdoor environments (see Fig. 6.1). The first block is an indirect visual gyroscope composed by a convolutional neural network called HoLiNet (**Ho**rizon **Li**ne **Net**work) where we obtain a first approximation of Roll-Pitch angles from a single input image. The second block is a direct visual gyroscope algorithm that uses MPP (**M**ixture of **P**hotometric **P**otentials) Caron and Morbidi (2018) to retrieve the Yaw angle with respect a reference image, taking as initial solution for the first two angles the orientation from HoLiNet. The third and final block is second direct visual gyroscope algorithm defined as PVG (**P**hotometric **V**isual **G**yroscope) where we refine the previous solutions, obtaining a more accurate solution.

6.2.1 HoLiNet

Our proposed HoLiNet follows an encoder-decoder structure taking as input information an equirectangular panorama in any orientation and providing two heat-maps, one for the horizon line and another for the vertical vanishing points (see Fig. 6.1).

[1]Code is available in https://github.com/Sbrunoberenguel/HoLiNet

For the encoder and feature extractor we use the convolutional part of ResNet-50 He et al. (2016). We use the available pre-trained weights, leveraging the performance of this network for feature extraction. In the decoder we propose a set of convolutional layers followed by up-scaling layers and skip connections from the encoder part. The output of the decoder is a two-channel heat-map with the horizon line and vanishing points prediction. To adapt the neural network to the distortion, we use in all the convolutional layers convolutions adapted to the equirectangular projection. These convolutions, EquiConvs, were first presented in Fernandez-Labrador et al. (2020). However, we have re-implemented these convolutions, pre-computing the kernel's distortion, for a faster inference time (i.e. the original implementation runs at 0.2fps while ours can run at 10fps). These convolutions have also been presented in Section 4.3.4 from Chapter 4, where we also use this implementation to handle equirectangular distortion for the semantic segmentation and depth estimation tasks.

HoLiNet is trained in a collection of gravity-oriented equirectangular panoramas from indoor and outdoor environments Fraguas Bordonaba and Bermúdez Cameo (2020). We apply rotations to these panoramas and generate the ground truth labelling according to these rotations. We also perform several data augmentations such as: horizontal flip, color permutation, color jitter and random cropping. At training, we obtain intermediate representations at different resolutions from the decoder of the network. These intermediate representations allow to evaluate the network on early stages, making the intermediate features of the network aim for the final task, improving the overall performance of the network.

From the output information of the network, we compute the most probable horizon plane, obtaining a first approximation of the camera's Roll-Pitch angles. To compute the horizon plane, we use the information of both outputs of the network. First, we use the vanishing point heat-map to make a rough estimation of the vertical direction. We project the vanishing points heat-map in the unit sphere and compute the average vertical direction. This is a rough estimation since the output is really sparse and can be noisy. Then, we project the horizon line heat-map into the unit sphere. Using RANSAC, we compute the 3D plane that better fits the horizon line heat-map in the unit sphere. Assuming that both network's outputs are coherent with each other, the first estimation of the vertical direction and the normal vector of the horizon plane should be close. So, to maintain this coherence, in the iterative process we discard any possible solution which differs from the first estimation more than a threshold angle (i.e. we assume a threshold of $\pm 30^\circ$).

Once obtained the plane that better fits the output of the network, we can compute the Roll and Pitch angles of the camera (Yaw is the rotation around the plane's normal vector, which cannot be computed with this method). The main advantage of this method is that we can obtain 2 angles from a 360° domain. The main limitation is the angle precision, which is limited by the uncertainty of the output of the network, and the impossibility to obtain the Yaw angle with the proposed representation.

6.2.2 MPP-based visual gyroscope

The MPP is a full spherical image representation of image brightness as a mixture of Gaussians. The MPP is structured as an icosahedron subdivided $n \in \mathbb{N}$ times, of which each vertex is the center of a Gaussian function, weighted by the normalized intensity of the pixel where it projects in the captured image. Thus, there are as many Gaussians as vertices of the sphere but all Gaussians of the MPP share the same variance $\lambda_g \in \{\mathbb{R} > 0\}$ that acts as a smoothing parameter. In the seminal work introducing the MPP-based visual gyroscope Caron and Morbidi (2018), captured dual-fisheye images are first transformed as MPPs: G^* for a reference image, G for a request image. Thus, λ_g acts as a smoothing parameter of the cost function, that is the sum of squared differences between G and G^*, to be minimized by optimizing the three angles of the 3D rotation that aligns the best both MPPs with a Newton-like algorithm.

In this work, the differences with respect to Caron and Morbidi (2018) are motivated by the HoLiNet-MPP-PVG pipeline we propose (Fig. 6.1):

- *input images are equirectangular* instead of dual-fisheye for both the sake of uniformity with respect to HoLiNet and more generality since the equirectangular format is much more common than dual-fisheye.
- *the single Yaw angle is estimated* since not only HoLiNet already outputs estimates of Roll and Pitch angles that can serve as good initial guesses for PVG (see Sec. 6.2.3) but it reduces a little the computation load.

The main advantage of the MPP-based visual gyroscope is its very large convergence domain, observed as being both globally convergent indoor for a single pure rotation angle and almost as large for 3D rotations, even in the presence of translations, in Caron and Morbidi (2018). However, its main drawback is the required trade-off between processing time and estimation accuracy: to reach a processing rate of 3 images per second, the icosahedron maximum subdivision level is $n = 3$ and the average estimation error was reported about 3 degrees with $\lambda_g = 0.325$. Thus, PVG is required as a final step to improve the accuracy of estimations.

6.2.3 Photometric visual gyroscope

To improve both the processing time and the accuracy with respect to the MPP-based visual gyroscope, the PVG André and Caron (2022) uses a spherical image representation directly from image brightness without a transformation to a MPP. The PVG is built on the same principle as the MPP-based visual gyroscope and also relies on the intensities of the image mapped to a subdivided icosahedron. However, instead of computing a MPP, the direct pixel brightness are assigned to each vertices of the icosahedron, allowing a much faster computation of the Jacobians in the Newton-like optimization of the orientations. When mapping full resolution images (typically 1280×720) to the subdivided icosahedron, a higher precision in the orientation estimation can be obtained. The spherical gradient is computed in the image plane (as highlighted in the PVG thumbnail of Fig.6.1) and not with neighbours vertices as in the MPP-based gyroscope.

The main difference of this work with respect to the source work André and Caron (2022) is that *equirectangular input images* are used, instead of dual-fisheye images, allowing a consistency of input images all along the orientation estimation pipeline.

PVG has shown estimation errors below 0.1° in the single axis rotation case, both pure and with translations André and Caron (2022). To reach such a precision, the icosahedron is subdivided $n = 5$ times and the captured image at full resolution is used for a computation time 2%the one of the MPP-based gyroscope. But inversely to the MPP-based gyroscope, the low processing load and the high accuracy come at the price of a narrow convergence domain that was observed at $\pm 12.5^\circ$ for the single rotation angle case. Hence, PVG needs good initial guesses that HoLiNet (Roll, Pitch) and the MPP-based visual gyroscope (Yaw) bring.

6.3 Experiments

We evaluate our pipeline with two datasets. The first dataset is Sequence 8 of PanoraMIS Benseddik et al. (2020) that was taken with a Ricoh Theta S mounted on a Parrot Disco fixed-wing drone. With this sequence of 12000+ images, we make a qualitative evaluation by compensating the estimated rotations with respect to image 8890 in order to compare the success rate with Caron and Morbidi (2018).

To allow the quantitative evaluation, we introduce a new dataset of 2700+ equirectangular images, SVMIS+[2]. SVMIS+ is acquired with a Ricoh Theta S embedded on a Matrice 600 Pro hexarotor, manually flown in a countryside area during 4′36”. The use of the hexarotor has two interests over the Disco drone. First, it allows to embed an extra computer connected to the onboard electronics in order to save the orientation information of the drone’s Inertial Measurement Unit synchronously with the image capture. Second, the different type of motion of a hexarotor compared to a fixed-wing drone provides an additional variety in the evaluation, together with different meteorological conditions in a similar environment. With this dataset, in addition to the quantitative evaluation, we also perform a qualitative evaluation by compensating the estimated rotations of all the images with respect to a selected frame. Frame 1074 is selected such that the image is roughly at the center of the flown area, with attitude angles near zero (horizon line near straight and horizontal in the image).

6.3.1 Results

With PanoraMIS’ Sequence 8, we evaluate qualitatively the success rate of the computed orientations by studying the misalignment of stabilized images with respect to the reference image. Over the 12000+ images, 564 were misaligned. This represents a success rate of 95.3%, thus out-performing

[2]http://mis.u-picardie.fr/~g-caron/pub/data/SVMISplus_er_pano.zip

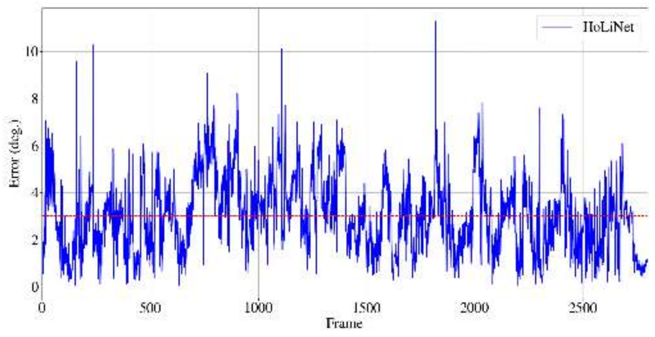


Figure 6.2: Quantitative results of HoLiNet for 2 angles. Dashed lines show mean error.

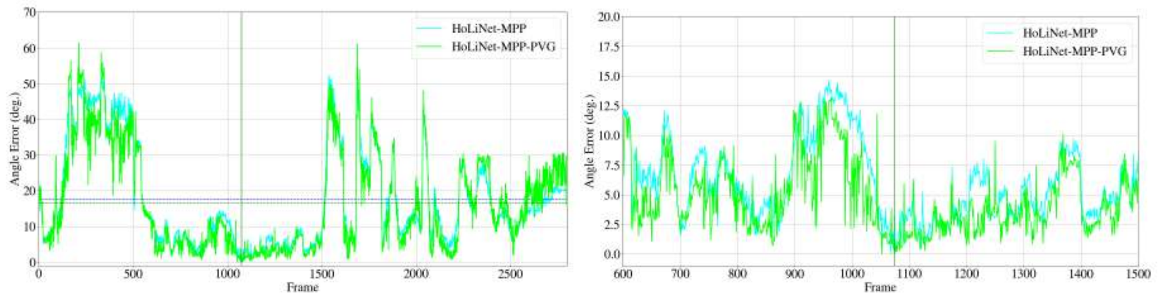


Figure 6.3: a) Quantitative results of HoLiNet+MPP and HoLiNet+MPP+PVG in the SVMIS+ dataset. b) (Zoom) Quantitative results of MPP and PVG around the reference image. Vertical green line defines the reference image and horizontal dashed lines show mean error of each method.

previously shown results of 88% Caron and Morbidi (2018). The stabilization results with the whole first dataset are shared as a video[3] highlighting each intermediate result of our pipeline for orientation estimation.

With the synchronized and reliable ground truth provided with SVMIS+, a quantitative evaluation of the pipeline is put in practice. HoLiNet is only evaluated over 2 angles. To avoid ambiguities in the evaluation of the two angles, we compute the predicted normal vector, n, of the horizon plane and compare with the ground truth measure, $\hat{n}$, computing the angle between both vectors as: $\arccos(n \cdot \hat{n})$. The quantitative results are shown in Fig.6.2, obtaining a mean error of 3.04°.

The evaluation for the 3 angles is made computing the angle difference between the angle-axis rotation of ground truth with respect to the reference image, R_{gt}, and our estimation, R_{pred}, as: $\arccos\left((trace(R_{gt} \cdot R_{pred}{}^{T}) - 1)/2\right)$. The results are presented in Fig.6.3, obtaining a mean error of 17.6° for HoLiNet + MPP, 16.7° for the whole pipeline over the whole sequence, and 4.6° for 900 images (33% of SVMIS+) around the reference image (Fig.6.3b), corresponding to a difference of altitude with respect to the reference image lower than 10m. We also present a video[4] of qualitative results with SVMIS+.

[3]https://www.youtube.com/watch?v=cyitAD5YnYg&t=6s
[4]https://www.youtube.com/watch?v=poNrJaFlrPk&t=37s

6.4 Conclusions

We have presented a novel pipeline that leverages the strong points of three methods, HoLiNet using neural networks and two using direct alignment, for panoramic image rotation compensation in outdoor environments. The quantitative results on two datasets of airborne panoramic images show HoLiNet is robust to any translation and improves the visual alignment success rate. But since the direct alignment methods following HoLiNet in the pipeline assume pure rotation between the reference and current images, the presence of a large translation decreases the rotation estimation accuracy, as shown with the newly introduced public dataset SVMIS+. However, the rotation estimation appears accurate within a reasonable translation to the reference image that further study with SVMIS+ will allow to quantify during future works.

Chapter 7

Calibrated convolutions for fisheye cameras

Convolution kernels are the basic structural component of convolutional neural networks (CNNs). In the last years there has been a growing interest in fisheye cameras for many applications. However, the radially symmetric projection model of these cameras produces high distortions that affect the performance of CNNs, especially when the field of view is very large. In this chapter, we tackle this problem by proposing a method that leverages the calibration of cameras to deform the convolution kernel accordingly and adapt to the distortion. That way, the receptive field of the convolution is similar to standard convolutions in perspective images, allowing us to take advantage of pre-trained networks in large perspective datasets.

7.1 Introduction

Nowadays, Neural Networks (NN) are the standard and globally adopted solutions for many machine learning approaches. Among them, Convolutional Neural Networks (CNNs) are the state-of-the-art approaches to handle images and understand what a computer can see. Following classical computer vision algorithms, the first CNNs worked on conventional images (i.e. perspective images). These images provide information of the environment in a relatively small field of view, which is usually enough for the CNNs to achieve great performance in many tasks. This is possible due to the large number of labelled datasets of perspective images, which provide an excellent foundation for these solutions.

On a new trend, unconventional cameras with wider fields of view are becoming popular in many applications and devices such as autonomous vehicles or augmented reality. In particular, we focus on fisheye cameras, which provide several advantages in scene understanding problems. The wide field of view (possibly wider than 180°) allows us to achieve a better understanding of the device's surrounding with less images. Compared to perspective cameras, with fisheyes it is possible to gather much more spatial information at once, providing more context to the network predictions, especially when the task involve a global understanding of the scene. This is particularly useful in tasks such as

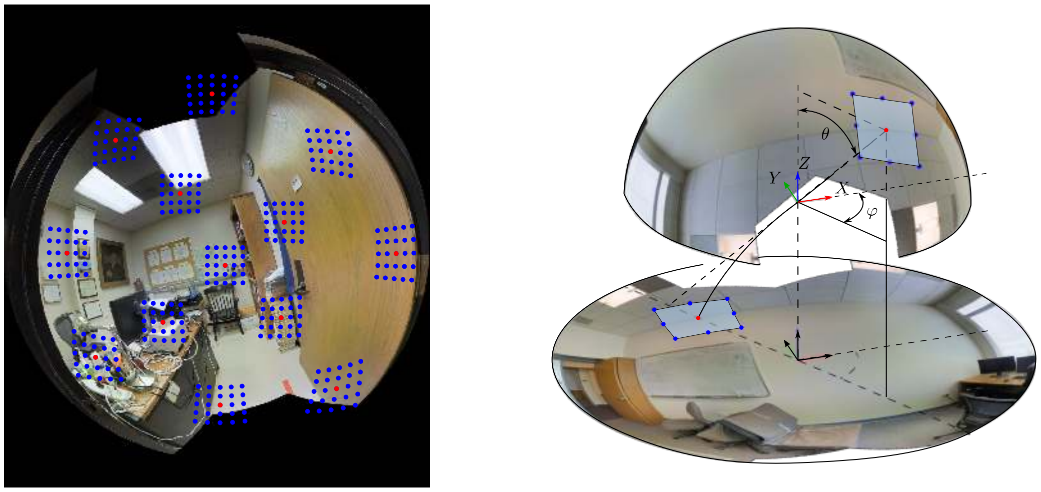


Figure 7.1: Overview of how a standard convolution is deformed by the Kannala-Brandt's projection model in a fisheye image. a) Several convolutional kernels adapted to the calibrated fisheye image. b) How the convolutional kernels are computed with the Kannala-Brandt's projection model.

semantic segmentation, depth estimation, or room layout estimation.

However, these cameras have some important drawbacks yet to be addressed, which mainly are the reduced number of labelled datasets (mostly because of the difficulty of manually label these images), and the strong distortions induced in large field of view cameras. The last reason causes that existing CNN-based approaches do not transfer well to fisheye images, since the appearance of the elements in the scene is very different to what was learned with perspective datasets. Besides, the contextual information has drastically changed with the much wider field of view, which consequently deteriorate the network's ability to understand the scene. There is definitely a domain gap between perspective and fisheye cameras that needs to be addressed to fully exploit the potential of these devices.

Our proposal is a method to reduce this domain gap and easily train and use CNNs with highly distorted cameras. We take advantage of the classical perspective CNNs trained with massive datasets, which already provide impressive performance in many different tasks, and adapt these networks to calibrated fisheye cameras. To do so, we propose to substitute the standard convolution operation with deformable convolutions pre-computed with the camera calibration. During the convolution operation, kernels are deformed to accommodate to the distortion depending on the position in the image (see Fig. 7.1). With minimal fine-tuning on a small set of data, we can achieve the good performance of well-known CNNs from perspective cameras to fisheye cameras, not needing to create new large datasets specific to each desired camera calibration in order to help the network to learn the distortions. The main contributions of this work are:

- We present a novel implementation of calibrated deformable convolution for fisheye cameras under the Kannala-Brandt projection model, which could be used with any fisheye camera (even with a field of view wider than 180°).

- We propose a set of experiments of domain adaptation of well known CNNs for several tasks with fisheye cameras. Particularly, we show results with different fields of view, showing that our method allows great flexibility in the camera configurations with minimal effort.

7.2 Related work

In recent years, there has been a surge of using fisheye and 360° cameras since they introduce more information within a single image, which is advantageous when tackling different computer vision tasks such as: scene understanding Gao et al. (2022), depth estimation Kumar et al. (2018); Berenguel-Baeta et al. (2023b), semantic segmentation Berenguel-Baeta et al. (2023b); Deng et al. (2017), object Guerrero-Viu et al. (2020) and pedestrian detection Haggui et al. (2021), or autonomous driving Toromanoff et al. (2018), among others Baek et al. (2018); Ni et al. (2019); Perez-Yus et al. (2019). There is also an increasing presence of wide-angle cameras such as fisheyes in mobile devices such as phones, or VR headsets and stereo cameras, due to the improved robustness in localization and mapping from the larger field of view.

However, when it comes to deep learning methods applied to wide angle images, one of the most common operations, the convolution, is flawed. The space-varying distortions caused by the image projection models for omnidirectional and wide-angle cameras makes the translational weight sharing ineffective Cohen et al. (2018). Objects appear differently distorted depending on where they are in the image, which makes more difficult for the network to learn each plausible configuration, especially considering different camera calibrations. Therefore, there is still an open challenge about how to use and train traditional CNN architectures with these kinds of images, also considering the lack of large datasets compared to perspective images.

Some researchers have focused on adapting CNNs to the spherical domain. For instance, Cohen et al. (2018) proposed Spherical CNNs, studying convolutions on the sphere using spectral analysis. Jiang et al. (2019) replace conventional convolution kernels with linear combinations of differential operators weighted by learnable parameters on unstructured grids. Su and Grauman (2017) aims to adapt a CNN trained on perspective images to the equirectangular domain adjusting the sizes of the rectangular kernels depending on the elevation angle. Jiang et al. (2021) proposed to fuse features from cubemap projection with regular convolutions on equirectangular images on the decoding stage for depth estimation. In an attempt to make more efficient the computation, Zioulis et al. (2019) use spherical attention masks to make the model aware of its spherical nature. Transformer architectures have also appeared in this topic, with Li et al. (2022) who divides the spherical image in rectified patches and use geometric embeddings to estimate depth in equirectangular projection images.

On the other hand, different approaches have been proposed to enable CNNs to be more dynamic, improving the performance for specific tasks, as well as extending their applicability to new domains. For example, Jeon and Kim (2017) and Dai et al. (2017b) focus on convolution units with no fixed shape and learned offsets for the convolution kernel. Jeon and Kim (2017) use these convolutions to obtain better receptive field for object classification, whereas Dai et al. (2017b) incorporates spatial deformations into the convolutional operation to be able to handle objects with significant variations in shape or appearance.

The deformable convolutions Dai et al. (2017b) were used in Fernandez-Labrador et al. (2020) for layout estimation, with not learned but fixed offsets, pre-computed to account for the image distortion that occurs in equirectangular projections. Thus, the receptive field of the convolution filter is undistorted (like regular convolution on perspective images) regardless of the position in the image (e.g., the center vs near the poles). The idea of making the convolution "on the sphere" and project the convolution kernel with the equirectangular projection model was also proposed by Tateno et al. (2018) with their *distortion-aware convolutions*, introducing a pipeline of transfer learning from learned convolution filters in perspective images applied for depth estimation in equirectangular panoramic images. Strategies like transfer learning or domain adaptation have been successfully used in the past Donahue et al. (2014); Sharif Razavian et al. (2014), and we believe it could alleviate the absence of datasets for our task. This approach was recently explored for distortion-aware convolutions in Artizzu et al. (2023).

However, most works disregard the specific calibration parameters, mostly because they use very simple image projections such as equirectangular projection, that directly map azimuth and elevation angles in pixel locations. Thus, the research on these distortion-aware convolutions for other configurations such as fisheyes is scarce and very specific to the camera pose Meng et al. (2021). Only a few works directly deal with the camera calibration parameters into the convolutions, like CAMConvs Facil et al. (2019). In this chapter, we aim to breach that gap, introducing novel convolutions for radial-distortion models, like the Kannala-Brandt's projection model Kannala and Brandt (2006) or the Scaramuzza's projection model Scaramuzza et al. (2006a), that adapts to the specific calibration of the camera and is apt for any fisheye camera. Drawing inspiration from Fernandez-Labrador et al. (2020), our approach uses deformable convolutions Dai et al. (2017b) to adapt the convolution filters to the distortion caused by the projection. Up to our knowledge, this is the first work that explicitly deals with calibrated convolutions for radially distorted cameras. We show how, with just a little fine-tuning in a relatively small dataset, classical CNN methods can be adapted to any calibrated camera.

7.3 Fisheye convolution

Convolutions are the keystone of CNNs and current computer vision algorithms. Dai et al. (2017b) presents a learned deformable kernel to improve the performance of several CNNs. We propose to use calibration-based deformable kernels. We use the camera calibration of fisheye cameras to compute the offsets of the kernel positions and adapt the convolution to the distortion of these images. In this section we summarize the projection model used and how we apply the distortion to the kernels.

7.3.1 Fisheye projection model

In the literature of non-conventional cameras we find many works that propose mathematical models of several projections. Considering fisheye cameras, we can also find several models such as the *equidistance*, *stereographic*, *orthogonal* or *equisolid angle*. Each of these models propose a different non-linear function to fit the distortion of the projecting rays of different lens configurations and geometries. In our case, we aim to cover a wider set of projection models, so we use empirical models, which are more flexible to different or unknown projection models.

In the field of empirical models for camera calibration, two rise above the others: Scaramuzza's Scaramuzza et al. (2006a) and Kannala-Brandt's Kannala and Brandt (2006). Both models propose a high order polynomial function to model the camera distortion. In this work, we use the second one, Kannala-Brandt's Kannala and Brandt (2006), since it is the most extended in the calibration of commercial devices.

The full Kannala-Brandt model is explained in Kannala and Brandt (2006), where they present a radially symmetric model and a full model were the radial asymmetry of lenses is taken into account. For this work we use the first one, assuming that the radial asymmetry error is much lower than the pixel precision that can be obtained from an image. So, assuming a radially symmetric model, Kannala and Brandt (2006) define the forward projection model as:

$$\begin{pmatrix} u \\ v \end{pmatrix} = d(\theta) \begin{pmatrix} f_x \cos\varphi \\ f_y \sin\varphi \end{pmatrix} + \begin{pmatrix} c_x \\ c_y \end{pmatrix} \tag{7.1}$$

where (u,v) are the pixel coordinates in the image, (c_x,c_y) are the pixel coordinates of the optical center, (f_x,f_y) is the focal length on each axis, (θ,φ) are the spherical coordinates (see Fig. 7.1 (b)) of the incoming ray, $d(\theta) = k_1\theta + k_2\theta^3 + k_3\theta^5 + k_4\theta^9$ is the high order polynomial function and $[k_1,k_2,k_3,k_4]$ the Kannala-Brandt calibration parameters.

The back projection model is computed as:

$$\varphi = \arctan\frac{m_y}{m_x}; \; \theta = d^{-1}\left(\sqrt{m_x^2 + m_y^2}\right), \tag{7.2}$$

where $m_x = (u - c_x)/f_x$, $m_y = (v - c_y)/f_y$ and d^{-1} is computed iteratively.

7.3.2 Fisheye calibrated kernel

Deformable convolutions were presented in Dai et al. (2017b), where the authors propose a method to learn offsets in the kernels for a better adaptation of the CNN to the task at hand. On the other hand, using the calibration of perspective cameras on CNNs has also been addressed by Facil et al. (2019), obtaining improvements in the performance of the network. In this section, we present our implementation of a camera-calibrated kernel for non-linear projection models, adapting the kernel to the fisheye distortion of the Kannala-Brandt projection model.

Let $\mathscr{K}$ be a $(k_i \times k_j)$ rectangular kernel where $(k_i, k_j) \geq 1$ are an odd number and (u_0, v_0) is the anchor pixel around which we will apply the convolution kernel. We define the coordinates of each element of $\mathscr{K}$ as: $\hat{p}_{ij} = (i, j, d)^T$ where i is in range $\left[-\frac{k_i - 1}{2}, \frac{k_i - 1}{2}\right]$; j is in range $\left[-\frac{k_j - 1}{2}, \frac{k_j - 1}{2}\right]$; and d is the focal distance of $\mathscr{K}$. We assume that the standard kernel has the same behaviour as in a perspective camera, so we compute the focal distance as a function of the field of view of the kernel, α, which we linearly map from the fisheye camera field of view, Φ, as: $d = \frac{k_i}{2 \tan \frac{\alpha}{2}}$ where $\alpha = \frac{k_i}{W} \Phi$ where W is the size of the feature map.

We project each kernel point into the unit sphere surface by normalizing the vectors. Then, we want to go back to the pixel domain using the forward projection model of the camera, Eq. 7.1. However, the resolution of the feature maps of a network (almost) always differs from the input image of the network, which is the resolution of the calibration parameters of the camera. Besides, we have to align the anchor of the kernel with the pixel we want to apply the kernel.

To solve the multi-scale/multi-resolution problem, we compute an scaling factor for each resolution that relates the calibration resolutions with the current feature map. The scaling factor is defined as $s = \frac{W_c + p_w}{W_{FM}}$, where W_c is the camera resolution width, W_{FM} is the feature map width and p_w is the width of an additional padding to the input image. We use this padding to set a fixed input resolution to our network in case of different image resolutions. With the scaling factor, we re-compute the calibration parameters for the kernel as:

$$\begin{pmatrix} cx_k \\ cy_k \end{pmatrix} = \begin{pmatrix} c_x \frac{W_{FM} - p_w/s}{W_c} \\ c_y \frac{H_{FM} - p_h/s}{H_c} \end{pmatrix}; \begin{pmatrix} fx_k \\ fy_k \end{pmatrix} = \begin{pmatrix} f_x \frac{W_{FM} - p_w/s}{W_c} \\ f_y \frac{H_{FM} - p_h/s}{H_c} \end{pmatrix}, \tag{7.3}$$

where $(cx_k, cy_k), (fx_k, fy_k), (W_{FM}, H_{FM})$ are the coordinates of the optical center, focal lengths and resolution of the feature map and (W_c, H_c) the resolution of the fisheye camera.

Once the calibration parameters are adjusted to the feature map resolution, we compute the projecting ray of the anchor pixel (u_0, v_0) using Eq.7.2 and rotate the projecting rays of $\mathscr{K}$ to meet the orientation. With the $\mathscr{K}$ in the correct position, we project again each element of the kernel into the fisheye plane with Eq.7.1, obtaining the new locations of the kernel in the fisheye image (or feature map). From this implementation, we obtain a convolution kernel that adapts its shape with the distortion of the camera following the radially symmetric projection model (see Fig. 7.1).

7.4 Experiments

For the experimental part, we evaluate the calibrated convolutions against the standard convolutions on fisheye cameras. For that purpose, we use a well known CNN, U-Net Ronneberger et al. (2015), on two different tasks to evaluate the performance of the proposed kernels. We want to avoid current architectures where convolutions are mixed with other components as recurrent blocks Hochreiter and Schmidhuber (1997) or attention mechanisms Vaswani et al. (2017) to evaluate only the impact of the convolutions in the overall performance. For a fair comparison between convolutions, we propose the following set-up for the experiments:

1. We train the CNN, with standard convolutions, on perspective images of the Stanford dataset Armeni et al. (2017) and use these weights as baseline.

2. We evaluate the network with standard convolutions and calibrated convolutions on fisheye images obtained from the Stanford dataset Armeni et al. (2017).

3. With the baseline as pretrained weights, we fine tune the network with fisheye images in two different situations: one fine tune is made with standard convolutions and other with the proposed calibrated convolutions.

4. After fine tuning, we evaluate again both networks on the fisheye images.

Training and fine tuning is made in the Stanford dataset following the #1 folder split, taking the *Area 5* only for evaluation and the others as training and validation sets. Perspective images are taken from the original dataset, where we find around 70k images with depth and semantic information. For simplicity, we define this dataset as *Pers* in the experiments. Fisheye images are randomly synthesized from the panoramic dataset in different orientations and with known calibration. We have generated 11.2k images for two different fisheye calibrations (i.e. 5.6k images of each calibration). For simplicity, we define these datasets by the field of view of the camera, such as: *F165* defines the dataset of fisheye images with a field of view of 165° and *F195* with a field of view of 195°.

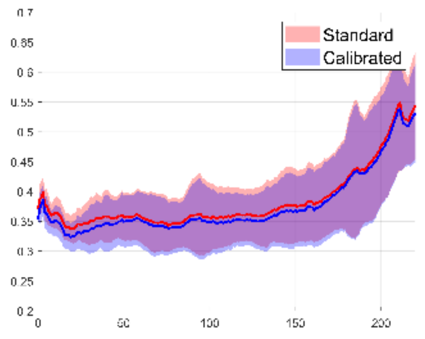
7.4.1 Monocular depth estimation

The first task we evaluate is monocular depth estimation from single images. The convolutional network has been trained for 50 epochs on the Stanford dataset with perspective images, which has taken around 100 hours to complete. This network is our baseline (BL) for the experiment. Then, the fine tuning (FT) has been made on top of this training for less than 10% of the training time. The network has been fine tuned for 20 epochs in the fisheye dataset, taking between 2-4 hours. This time changes with the resolution of the fisheye images, being different for each field of view.

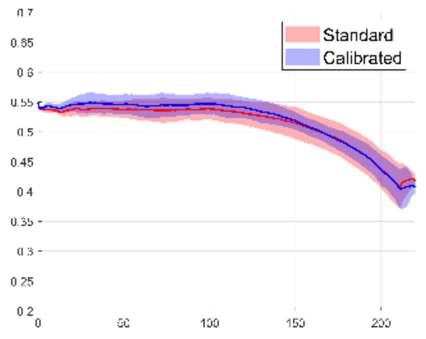
Results of this experiment are shown in Table 7.1. We use the standard metrics for depth estimation presented in Zioulis et al. (2018). We also compute the metrics with respect the distance of each pixel to the principal point of the camera (i.e. $d(\theta)$ from equation 7.1) to observe the behaviour of each convolution with the increasing distortion of the image. These results are presented in Fig. 7.2 for both fisheye datasets. Additionally, we present qualitative results of monocular depth estimation in Fig. 7.3 and a point-cloud reconstruction from the depth estimation in Fig. 7.4.

Table 7.1: Monocular depth estimation with standard (Std) and calibrated (Calib) convolutions for U-Net neural network. BL: Base Line; FT: Fine Tuned.

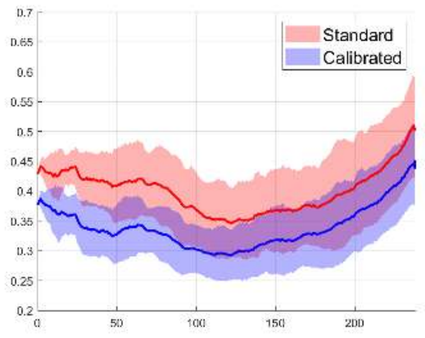
	Dataset	Kernel	MRE ↓	MAE ↓	RMSE ↓	RMSE_{log} ↓	δ^1 ↑	δ^2 ↑	δ^3 ↑
BL	*Pers*	Std	0.1538	0.2462	0.4908	0.1146	0.6391	0.8678	0.9407
	F195	Std	0.2726	0.3914	0.4943	0.2408	0.4186	0.7027	0.8517
		Calib	0.2922	0.4133	0.5262	0.2604	0.3839	0.6745	0.8371
	F165	Std	0.2670	0.3790	0.5005	0.2324	0.4377	0.7198	0.8579
		Calib	0.2798	0.3971	0.5216	0.2441	0.4019	0.6998	0.8500
FT	*F195*	Std	0.2432	0.3729	0.4023	0.2022	0.4241	0.7277	0.8827
		Calib	0.2017	0.3159	0.3418	0.1575	0.5450	0.7972	0.9075
	F165	Std	0.2508	0.3582	0.4040	0.1899	0.4962	0.7628	0.8879
		Calib	0.2505	0.3561	0.3875	0.1865	0.4992	0.7648	0.8884



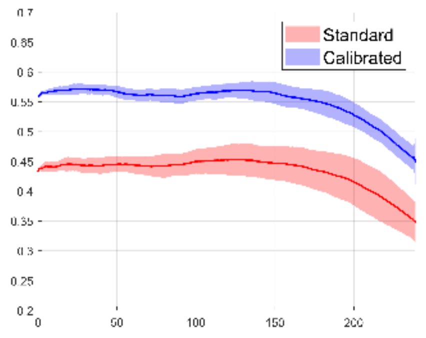
F165 RMSE



F165 δ^1



F195 RMSE



F195 δ^1

Figure 7.2: Comparison and results of depth estimation with U-Net neural network with standard (red) and calibrated (blue) convolutions. The x-axis defines the distance of the pixels to the optical center and the y-axis the computed error, defined as mean and one standard deviation.

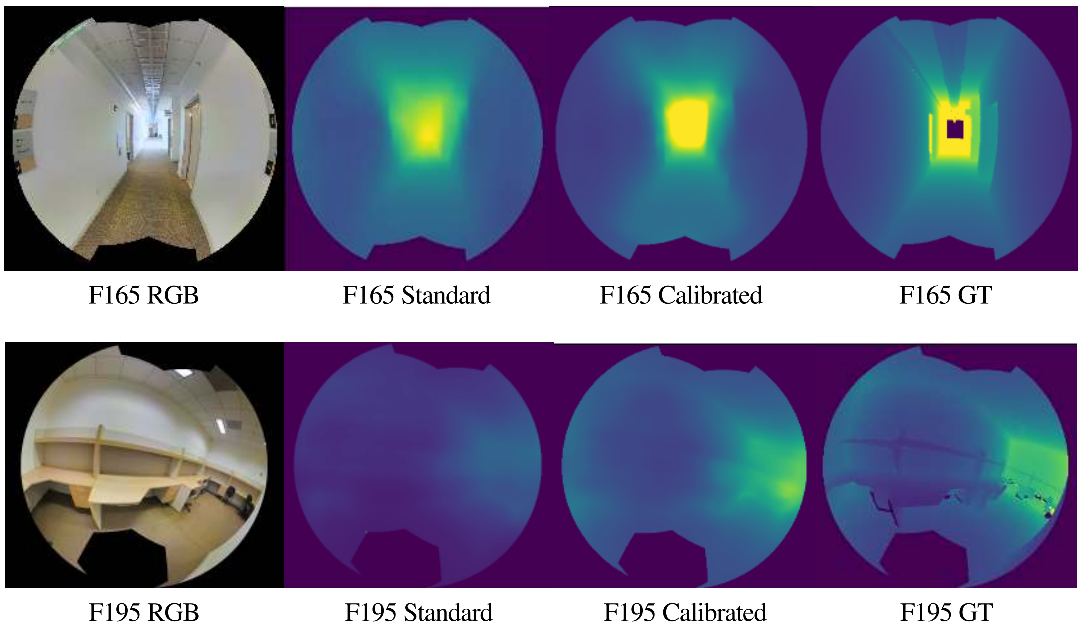


Figure 7.3: Qualitative results of monocular depth estimation on different fisheye calibrations.

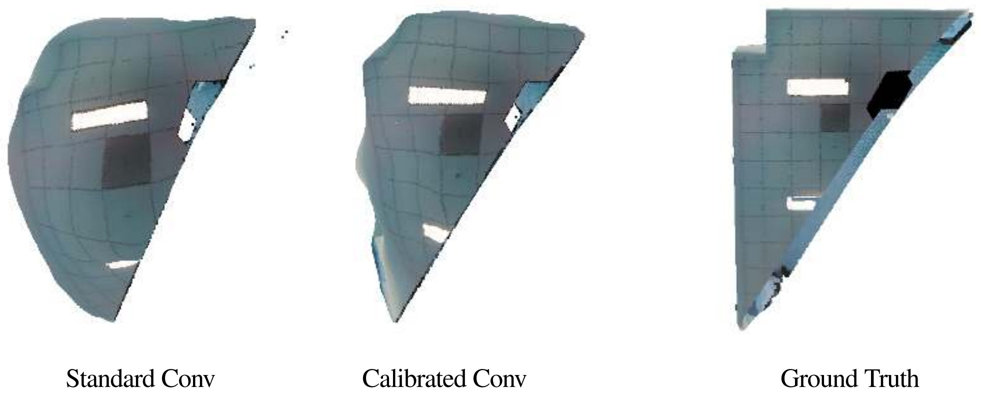


Figure 7.4: Qualitative results of depth estimation for FOV of 195º, top view of a 3D point cloud generated from depth data.

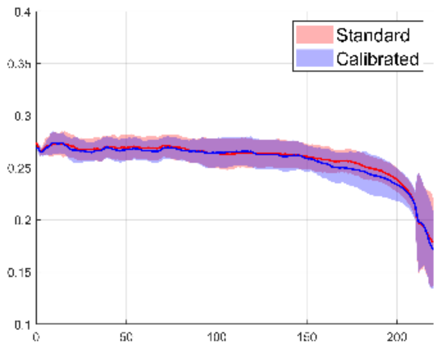
7.4.2 Semantic segmentation

The second task that we evaluate with U-Net Ronneberger et al. (2015) is semantic segmentation. We use the same set-up and dataset than in the previous experiment. We train the network for 50 epochs, taking 75 hours to train, and then fine tune it 20 more epochs, taking between 5-7 more hours.

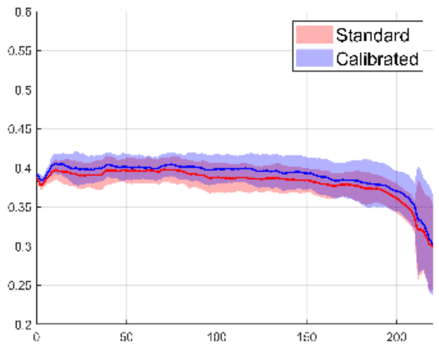
Results of this experiment are shown in Table 7.2. The metrics used are the mean Intersection over Union (mIoU) and the mean Accuracy (mAcc) over all the classes, except the unknown class. We also compute the metrics with respect the distance of each pixel to the principal point of the camera to

Table 7.2: Semantic segmentation with standard (Std) and calibrated (Calib) convolutions for U-Net neural network. BL: Base Line; FT: Fine Tuned.

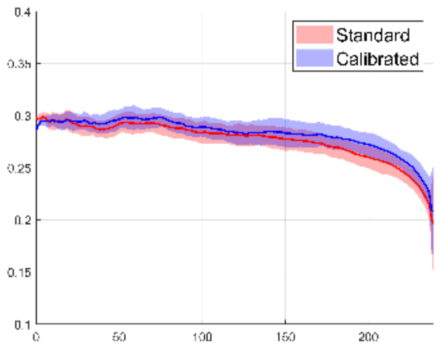
	Dataset	Kernel	mIoU	mAcc
BL	*Pers*	Std	55.03	63.02
	F195	Std	15.05	24.41
		Calib	13.73	22.55
	F165	Std	15.12	24.36
		Calib	13.23	21.46
FT	*F195*	Std	29.48	43.40
		Calib	30.90	46.90
	F165	Std	27.70	36.51
		Calib	27.89	36.71



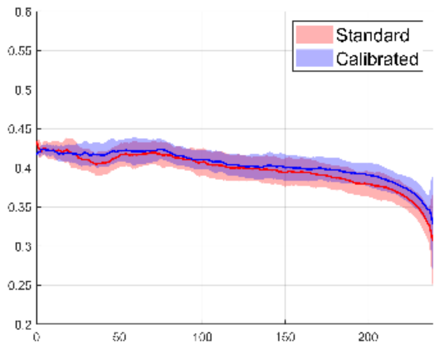
F165 mIoU



F165 mAcc



F195 mIoU



F195 mAcc

Figure 7.5: Comparison and results of semantic segmentation with the U-Net like network with standard (red) and calibrated (blue) convolutions. The x-axis defines the distance of the pixels to the optical center and the y-axis the computed error, defined as mean and one standard deviation.

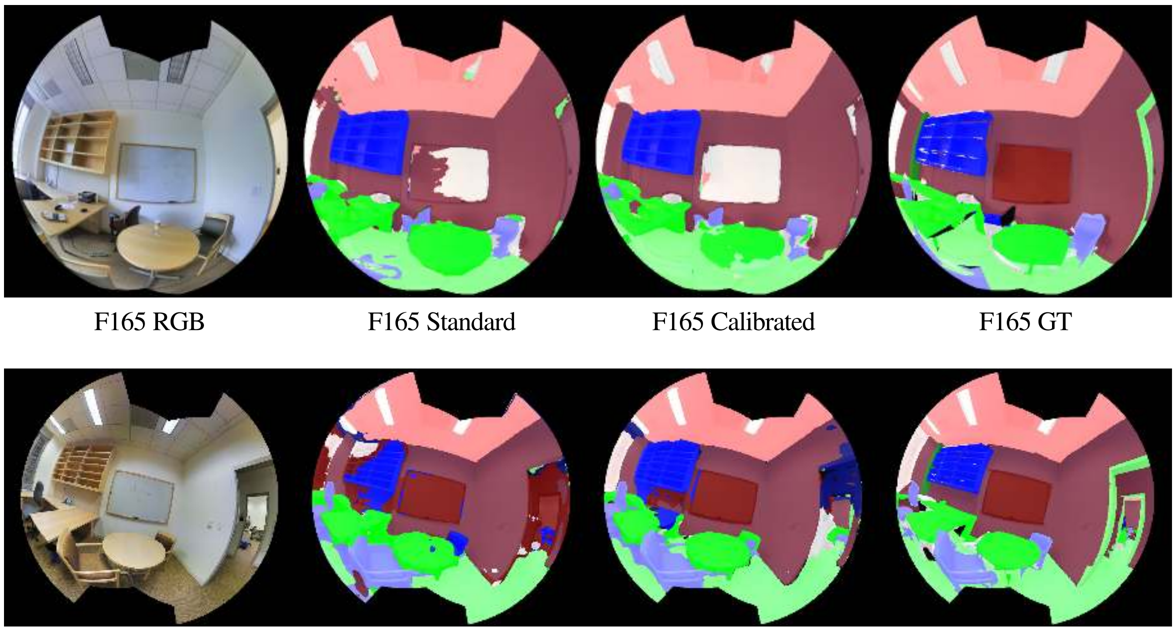


Figure 7.6: Qualitative results of semantic segmentation on different fisheye calibrations.

evaluate the behaviour of the network with the increasing distortion of the image. These results are presented in Fig. 7.5 and qualitative results of semantic segmentation are presented in Fig. 7.6.

7.4.3 Image rectification

One of the main questions that now may rise is: *Why don't we rectify the fisheye image and use the Convolutional Neural Networks for perspective images on the rectified image?* The answer for this question comes in two parts:

1. Not all the wide-field-of-view images can be rectified. Cameras with a field of view equal or greater than 180° cannot be rectified completely, since the perspective projection model has a limit at this field of view. That means that we may loss part of the environment information in the rectification process.

2. Although the image rectification corrects the radial distortion, when having wide field of view the effect of projecting onto a plane results in large deformations in regions far from the principal point. In some cases, this projective deformation modifies the receptive field of the CNNs even more than the original radial distortion.

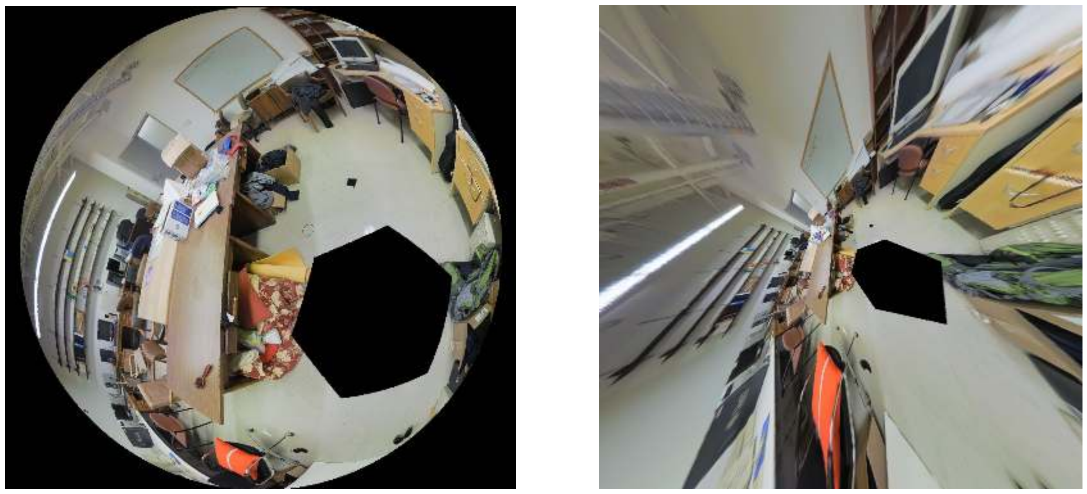


Figure 7.7: Comparison of a fisheye image from the dataset F165 and the rectification of the image in the perspective projection model.

Table 7.3: Monocular depth estimation with standard convolutions for U-Net neural network with fisheye (F165) and rectified (F165-R) input images. BL: Base Line; FT: Fine Tuned.

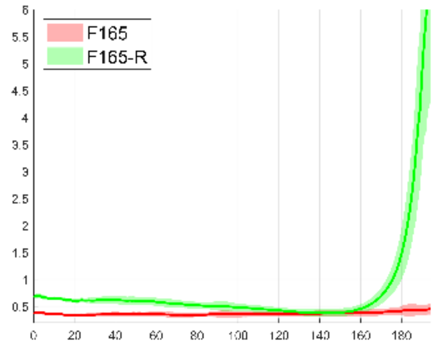
	Input	MRE ↓	MAE ↓	RMSE ↓	RMSE_{log} ↓	δ^1 ↑	δ^2 ↑	δ^3 ↑
BL	*F165*	0.2670	0.3790	0.5005	0.2324	0.4377	0.7198	0.8579
	F165-R	0.8595	0.6412	0.9714	0.4178	0.3000	0.5509	0.7305
FT	*F165*	0.2508	0.3582	0.4040	0.1899	0.4962	0.7628	0.8879
	F165-R	0.7758	0.5999	0.8933	0.3661	0.3016	0.5710	0.7618

In the previous experiments, we present results of two different fisheye camera calibrations, each corresponding to a different field of view. For one of the cameras, this rectification process is impossible to achieve completely (i.e. the camera has a field of view of 195°) while with the other, with a field of view of 165°, this rectification is possible but the rectified images are greatly deformed (see Fig. 7.7).

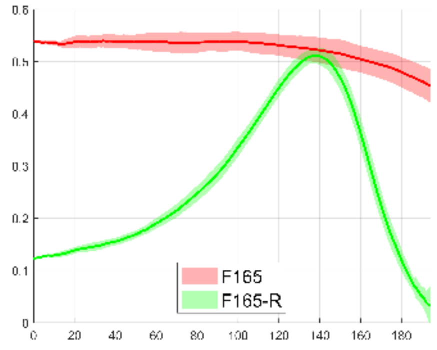
To evaluate the performance of a network with rectified images, we compare the CNN based on standard convolutions when directly using the fisheye image as input (F165), as previously considered, versus using a rectified version of the image as input (F165-R). The evaluation is made rectifying the whole input image, estimate depth or semantic segmentation and un-rectify the output of the network to compare with ground truth in the fisheye domain. Average quantitative results are shown in Tables 7.3 and 7.4 and quantitative results with respect the distance to the principal point are shown in figures 7.8 and 7.9 for depth estimation and semantic segmentation respectively.

Table 7.4: Semantic segmentation with standard convolutions for U-Net neural network with fisheye (F165) and rectified (F165-R) input images. BL: Base Line; FT: Fine Tuned.

	Input	mIoU	mAcc
BL	*F165*	15.12	24.36
	F165-R	11.81	19.82
FT	*F165*	27.70	36.51
	F165-R	21.99	29.61

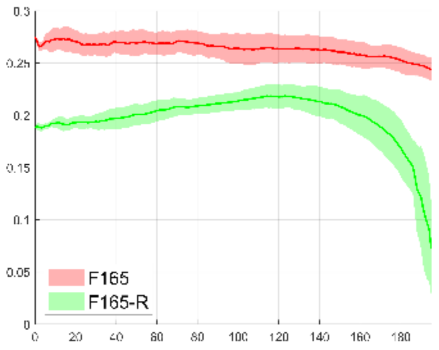


F165 RMSE

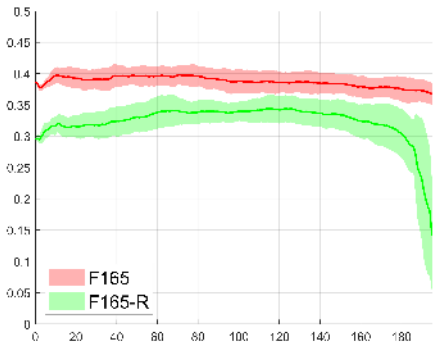


F165 δ^1

Figure 7.8: Comparison and results of depth estimation with the U-Net like network with standard convolutions on the fisheye (F165) and rectified (F165-R) images. The x-axis defines the distance of the pixels to the optical center and the y-axis the computed error, defined as mean and one standard deviation.



F165 mIoU



F165 mAcc

Figure 7.9: Comparison and results of semantic segmentation with the U-Net like network with standard convolutions on the fisheye (F165) and rectified (F165-R) images. The x-axis defines the distance of the pixels to the optical center and the y-axis the computed error, defined as mean and one standard deviation.

7.5 Discussion

The experiments and results presented show that the transfer learning problem is still an open topic and difficult to achieve in the conversion from perspective to omnidirectional images. Both monocular depth estimation and semantic segmentation results present a great decrease of performance with the baseline weights of the network, being in some cases more accentuated when we use calibrated convolutions. However, after a short fine tune of the network, these results improve significantly, particularly with calibrated convolutions.

From the results of depth estimation, we observe that the fine tuned networks have slightly worse performance than the baseline with perspective images. The main difference is that with the fisheye images we cover a wider field of view, obtaining more information of the scene with the same number of images. In the comparison of convolutions, we observe that with wider fields of view, the calibrated convolutional kernels provide better performance, while with the smaller field of view, the performance is quite similar. However, when we make a deeper analysis of these results, in the Figure 7.2 we observe the error distribution of the standard and calibrated convolutions. These results show that the estimation of the calibrated convolutions is more precise, with less dispersed error than the results of standard convolutions. Even if they are also affected by the increasing distortion of the fisheye images, the prediction is closer to the average error.

Regarding semantic segmentation experiments, the quantitative results from Table 7.2 show that the performance with fisheye images decreases significantly from the perspective image case. This is to be expected, since the segmentation problem difficulty increases with the field of view, including more objects to segment in the same image, and distortion, changing the appearance of the same object in different locations in the image. However we mitigate the second problem with the calibrated convolutions, obtaining better results with our proposal than with standard convolutions consistently along the radius in most metrics, particularly in larger radius (see Fig. 7.5). In addition, the qualitative results from Fig. 7.6 show significant differences in the performance between standard and calibrated convolutions. We can observe how the boundaries of objects and some details are better obtained with the calibrated convolutions than with the standard ones.

From the results of the image rectification we observe that rectify and un-rectify the fisheye images is not a good option for the proposed tasks with a CNN. The great deformation introduced in the rectification process seems to worsen the performance of the network with standard convolutions. Specifically, for depth estimation, the rectified image provides worse results in the outer part of the image. Besides, in the δ^1 metric, the behavior of the network presents better results in the middle range of the image radius, where the deformation in the rectified image is lower, and a really bad performance in the center and border of the image. For semantic segmentation, we observe a similar behaviour for both approaches (directly use the fisheye and with rectification). While the use of standard convolutions in the fisheye image directly, outperforms the rectification approach, both methods have worse performance at the border of the image. In this case, it is better to work directly on the fisheye image than rectifying it and use it in the network.

These results and conclusions led us to believe that calibrated convolutions provide a faster domain adaptation of CNNs, that means, in the same training conditions with limited data, the calibrated convolutions provide better performance that the standard ones. The adaptation of networks trained on perspective images can be done with small datasets of fisheye images, achieving similar performance.

7.6 Conclusion

We have presented a novel implementation of deformable convolutional kernels taking into account the intrinsic calibration of fisheye cameras. Integrating the Kannala-Brandt projection model for revolution symmetry cameras in the kernel of convolutional neural networks, we obtain a domain adaptation mechanism to take advantage of previous works on perspective images and adapt these networks to work with fisheye cameras. On a similar approach, this method could also be extended to other projection models that take into account the calibration of omnidirectional cameras, such as the Scaramuzza's model Scaramuzza et al. (2006a).

Results of the performed experiments show that the calibrated convolutions perform better than standard convolutions for domain adaptation. Besides, the impossibility of rectifying omnidirectional images of more than 180 degrees of field of view and the poor results obtained on the rectified ones increases the interest in studying how to adapt current deep learning methods to omnidirectional devices as the fisheye cameras.

A comparison with other methods, as CAM-Convs Facil et al. (2019), is not trivial. These methods should be extended to other projection models, since currently only work on the pin-hole camera model. With a naive approach, including directly the Kannala-Brandt model to the CAM-Convs proposal, we observe abrupt changes in the feature maps, which make us believe that further research is needed. This novel approach, the extension of methods as CAM-Convs to omnidirectional images, remains as future work.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

Omnidirectional cameras have become an important source of information in the computer vision and deep learning research fields. As shown in this thesis, omnidirectional systems have unique properties that make them really useful and advantageous for several applications such as scene understanding and navigation. Leveraging their advantages, such as their wide field of view, we can obtain more context information of the surrounding environment from a single image. However, this kind of images present several challenges that must be tackled in a smart way to achieve a good performance. In our research, we managed to propose several solutions for indoor scene understanding, image stabilization and image domain adaptation. Some of these solutions have achieved state of the art performance at their publishing time and are still in the top performance solutions with omnidirectional images.

As conclusion of adapting deep neural networks to omnidirectional images, the works presented in Chapter 4 and Chapter 7 show that the use of convolutions in different domains introduces important changes in the performance and behaviour of convolutional neural networks. In both works, we use convolutions adapted to the distortion of the image we use, i.e. we use EquiConvs for panoramic images and convolutions adapted to the calibrated fisheye camera for fisheye images. We observe how the networks improve when we use these *’distorted’* convolutional kernels on more challenging problems, from domain adaptation to indoor scene understanding of non-aligned images. Besides, in the work from Chapter 4, we also propose to change to the frequential domain using the fast Fourier convolutions. We realize that convolutions in the frequential domain could handle really well the distortion of several omnidirectional images. With or without specific convolutions adapted to the distortion, fast Fourier convolutions also allows the network to improve. We believe that working on the frequential domain, the distortion of the image does not affect the generation of new feature maps. Besides, the wider receptive field of these convolutions also favour the acquisition of more context information provided by omnidirectional images.

On other line of research, in Chapter 5 and Chapter 6, we combine deep learning solutions and classical methods such as geometric solvers and photometric algorithms. The synergy between classic and modern methods have proven essential for fast and accurate implementations on different applications. In general, geometric solutions and photometric algorithms for high resolution images require expensive computations in the absence of good first estimations, which is the most common case in unknown environments. However, deep neural networks have proven to provide good and general results in a fast way. So, this synergy seems direct: deep neural networks are fantastic tools to obtain a first good estimation in unknown environments while we can refine our objective with more precise methods, such as geometric solvers or photometric algorithms. I do believe that the future of applications where safety is a key component, such as navigation or autonomous driving, goes through the use of combined methods, where deep learning approaches can obtain a first estimation of the parameters to take into account in general and complex environments and geometric methods provide an accurate solution to avoid any dangerous situation.

8.2 Future Work

The use of omnidirectional images is still a really active research field with many research lines still to exploit. One interesting research line we still haven't exploited is the use of transformers Vaswani et al. (2017) or generative models Goodfellow et al. (2014) with omnidirectional images. In the last 5 years, the number of papers including one of these techniques has increased exponentially, also achieving remarkable results in different applications. It is our believe that most of these deep learning models are not yet well adapted to the challenges omnidirectional images pose. The distortion and properties of the projection models are missed in many new implementations. For example, architectures that use visual transformers Dosovitskiy et al. (2020) divide the image into rectangular patches which then are feed to the architecture. This approach is ill-conditioned for omnidirectional images, since these patches are not equi-area in the 3D nor take into account the image distortion or projection model.

Here is where our current and future research line enters in action. We want to study how to adapt these new deep learning architectures to omnidirectional images, taking into account the distortion, continuity and properties that these projection models provide. We want to explore new representations for omnidirectional images and novel approaches in the transformer and generative models implementation to better handle omnidirectional images. The creation of novel representations of omnidirectional images and adapting current and future methods to these images can improve the development of many applications and devices to push the future forward. Omnidirectional cameras can reduce cost of devices, reducing the number of cameras needed; reduce cost of image processing, since we only need to process one image instead of several; and improve the overall performance of algorithms, giving more rich information to computers.

Capítulo 9

Conclusiones

9.1 Conclusiones

Las cámaras omnidireccionales se han convertido en una importante fuente de información en los campos de investigación de la visión por computador y el aprendizaje profundo. Como se muestra en esta tesis, los sistemas omnidireccionales tienen propiedades únicas que los hacen realmente útiles y ventajosos para diversas aplicaciones, como la comprensión de escenas y la navegación. Aprovechando sus ventajas, como su amplio campo de visión, podemos obtener más información contextual del entorno circundante a partir de una única imagen. Sin embargo, este tipo de imágenes presentan varios desafíos que deben abordarse de manera inteligente para lograr un buen desempeño. En nuestra investigación, logramos proponer varias soluciones para la comprensión de entornos interiores, estabilización imágenes y la adaptación al dominio de la imagen. Algunas de estas soluciones han logrado ser la mejor solución del estado del arte en el momento de su publicación y todavía se encuentran entre las soluciones de mayor rendimiento con imágenes omnidireccionales.

Como conclusión de la adaptación de redes neuronales profundas a imágenes omnidireccionales, los trabajos presentados en el Capítulo 4 y el Capítulo 7 muestran que el uso de convoluciones en diferentes dominios introduce cambios importantes en el rendimiento y comportamiento de redes neuronales convolucionales. En ambos trabajos utilizamos convoluciones adaptadas a la distorsión de la imagen que utilizamos, es decir, utilizamos EquiConvs para imágenes panorámicas y convoluciones adaptadas a la cámara ojo de pez calibrada para imágenes ojo de pez. Observamos cómo las redes mejoran cuando usamos convoluciones *'distorsionadas'* para en problemas más complejos, desde la adaptación del dominio hasta la comprensión de escenas interiores de imágenes no alineadas. Además, en el trabajo del Capítulo 4, también proponemos cambiar al dominio frecuencial usando las convoluciones de Fourier. Nos damos cuenta de que las convoluciones en el dominio frecuencial podían manejar muy bien la distorsión de varias imágenes omnidireccionales. Con o sin convoluciones específicas adaptadas a la distorsión, las convoluciones de Fourier también permiten mejorar a la red. Creemos que trabajando en el dominio frecuencial, la distorsión de la imagen no afecta la generación de nuevos mapas de características. Además, el campo receptivo más amplio de estas convoluciones también favorece la adquisición de más información contextual proporcionada por imágenes omnidireccionales.

En otra línea de investigación, en el Capítulo 5 y el Capítulo 6, combinamos soluciones de aprendizaje profundo y métodos clásicos como solucionadores geométricos y algoritmos fotométricos. La sinergia entre los métodos clásicos y modernos ha demostrado ser esencial para implementaciones rápidas y precisas en diferentes aplicaciones. En general, las soluciones geométricas y los algoritmos fotométricos para imágenes de alta resolución requieren cálculos costosos en ausencia de buenas primeras estimaciones, que es el caso más común en entornos desconocidos. Sin embargo, se ha demostrado que las redes neuronales profundas proporcionan resultados buenos y generales de forma rápida. Así pues, esta sinergia parece directa: las redes neuronales profundas son herramientas fantásticas para obtener una buena primera estimación en entornos desconocidos mientras podemos refinar nuestro objetivo con métodos más precisos, como solucionadores geométricos o algoritmos fotométricos. Creo que el futuro de las aplicaciones donde la seguridad es un componente clave, como la navegación o la conducción autónoma, pasa por el uso de métodos combinados, donde métodos de aprendizaje profundo puedan obtener una primera estimación de los parámetros en entornos generales y complejos y los métodos geométricos proporcionan una solución precisa para evitar cualquier situación peligrosa.

Bibliography

Abdel-Aziz, Y. I., Karara, H., and Hauck, M. (2015). Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry. *Photogrammetric Engineering & Remote Sensing*, pages 103–107.

Agrawal, A. and Ramalingam, S. (2013). Single image calibration of multi-axial imaging systems. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 1399–1406. IEEE/CVF.

Agrawal, A., Taguchi, Y., and Ramalingam, S. (2010). Analytical forward projection for axial non-central dioptric and catadioptric cameras. In *European Conference on Computer Vision*, pages 129–143. Springer.

Agrawal, A., Taguchi, Y., and Ramalingam, S. (2011). Beyond alhazen's problem: Analytical projection model for non-central catadioptric cameras with quadric mirrors. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 2993–3000. IEEE.

André, A. N. and Caron, G. (2022). Photometric visual gyroscope for full-view spherical camera. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 5232–5235. IEEE/CVF.

Angus, M., ElBalkini, M., Khan, S., Harakeh, A., Andrienko, O., Reading, C., Waslander, S., and Czarnecki, K. (2018). Unlimited road-scene synthetic annotation (ursa) dataset. In *Proceedings of the International Conference on Intelligent Transportation Systems*, pages 985–992. IEEE.

Armeni, I., Sax, S., Zamir, A. R., and Savarese, S. (2017). Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*.

Artizzu, C.-O., Allibert, G., and Demonceaux, C. (2023). Omni-conv: Generalization of the omnidirectional distortion-aware convolutions. *Journal of Imaging*, 9(2):29.

Badrinarayanan, V., Kendall, A., and Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *Transactions on Pattern Analysis and Machine Intelligence*, pages 2481–2495.

Baek, I., Davies, A., Yan, G., and Rajkumar, R. R. (2018). Real-time detection, tracking, and classification of moving and stationary objects using multiple fisheye images. In *Intelligent vehicles symposium*, pages 447–452. IEEE.

Baker, S. and Nayar, S. K. (1999). A theory of single-viewpoint catadioptric image formation. *International Journal of Computer Vision*, pages 175–196.

Bao, S. Y., Sun, M., and Savarese, S. (2011). Toward coherent object detection and scene layout understanding. *Image and Vision Computing*, pages 569–579.

Benseddik, H., Morbidi, F., and Caron, G. (2020). Panoramis: An ultra-wide field of view image dataset for vision-based robot-motion estimation. *The International Journal of Robotics Research*, 39(9):1037–1051.

Berenguel-Baeta, B., Andre, A., Caron, G., Bermudez-Cameo, J., and Guerrero, J. (2023a). Visual gyroscope: Combination of deep learning features and direct alignment for panoramic stabilization. In *Proceedings of the Conference on Computer Vision and Pattern Recognition Workshop*, pages 6444–6447. IEEE/CVF.

Berenguel-Baeta, B., Bermudez-Cameo, J., and Guerrero, J. J. (2020a). Omniscv: An omnidirectional synthetic image generator for computer vision. *Sensors*.

Berenguel-Baeta, B., Bermudez-Cameo, J., and Guerrero, J. J. (2022). Atlanta scaled layouts from non-central panoramas. *Pattern Recognition*, page 108740.

Berenguel-Baeta, B., Bermudez-Cameo, J., and Guerrero, J. J. (2023b). Fredsnet: Joint monocular depth and semantic segmentation with fast fourier convolutions from single panoramas. In *Proceedings of the International Conference on Robotics and Automation*, pages 6080–6086. IEEE.

Berenguel-Baeta, B., Guerrero-Viu, M., Nova, A., Bermudez-Cameo, J., Perez-Yus, A., and Guerrero, J. J. (2020b). Floor extraction and door detection for visually impaired guidance. In *International Conference on Control, Automation, Robotics and Vision*, pages 1222–1229. IEEE.

Bermudez-Cameo, J., Barreto, J. P., Lopez-Nicolas, G., and Guerrero, J. J. (2014). Minimal solution for computing pairs of lines in non-central cameras. In *Asian Conference on Computer Vision*, pages 585–597. Springer.

Bermudez-Cameo, J., Demonceaux, C., Lopez-Nicolas, G., and Guerrero, J. J. (2016). Line reconstruction using prior knowledge in single non-central view. In *British Machine Vision Conference*.

Bermudez-Cameo, J., Lopez-Nicolas, G., and Guerrero, J. J. (2015). Automatic line extraction in uncalibrated omnidirectional cameras with revolution symmetry. *International Journal of Computer Vision*, pages 16–37.

Bermudez-Cameo, J., Lopez-Nicolas, G., and Guerrero, J. J. (2018). Fitting line projections in non-central catadioptric cameras with revolution symmetry. *Computer Vision and Image Understanding*, pages 134–152.

Bermudez-Cameo, J., Saurer, O., Lopez-Nicolas, G., Guerrero, J. J., and Pollefeys, M. (2017). Exploiting line metric reconstruction from non-central circular panoramas. *Pattern Recognition Letters*, pages 30–37.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Burt, P. J. and Adelson, E. H. (1983). A multiresolution spline with application to image mosaics. *Transactions on Graphics*, 2(4):217–236.

Caron, G. and Morbidi, F. (2018). Spherical visual gyroscope for autonomous robots using the mixture of photometric potentials. In *International Conference on Robotics and Automation*, pages 820–827. IEEE.

Caron, M., Touvron, H., Misra, I., Jegou, H., Mairal, J., Bojanowski, P., and Joulin, A. (2021). Emerging properties in self-supervised vision transformers. In *Proceedings International Conference on Computer Vision*, pages 9650–9660. IEEE/CVF.

Chang, A., Dai, A., Funkhouser, T., Halber, M., Niebner, M., Savva, M., Song, S., Zeng, A., and Zhang, Y. (2017). Matterport3d: Learning from rgb-d data in indoor environments. In *International Conference on 3D Vision*, pages 667–676. IEEE.

Chi, L., Jiang, B., and Mu, Y. (2020). Fast fourier convolution. In *Conference on Neural Information Processing Systems*, pages 4479–4488. NeurIPS Foundation.

Cohen, T. S., Geiger, M., Kohler, J., and Welling, M. (2018). Spherical cnns. In *International Conference on Learning Representations*.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 3213–3223. IEEE/CVF.

Cordts, M., Omran, M., Ramos, S., Scharwachter, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2015). The cityscapes dataset. In *Proceedings of the Conference on Computer Vision and Pattern Recognition Workshop*. IEEE/CVF.

Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T., and Niessner, M. (2017a). Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 5828–5839. IEEE/CVF.

Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., and Wei, Y. (2017b). Deformable convolutional networks. In *Proceedings of the International Conference on Computer Vision*, pages 764–773. IEEE.

Deng, L., Yang, M., Qian, Y., Wang, C., and Wang, B. (2017). Cnn based semantic segmentation for urban traffic scenes using fisheye camera. In *Intelligent Vehicles Symposium*, pages 231–236. IEEE.

Doan, A.-D., Jawaid, A. M., Do, T.-T., and Chin, T.-J. (2018). G2d: from gta to data. *arXiv preprint arXiv:1806.07381*.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655. PMLR.

dos Reis, E. S., Seewald, L. A., et al. (2021). Monocular multi-person pose estimation: A survey. *Pattern Recognition*, page 108046.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). Carla: An open urban driving simulator. In *Proceeding of the Conference on Robot Learning*, pages 1–16. PMLR.

Dvornik, N., Shmelkov, K., Mairal, J., and Schmid, C. (2017). Blitznet: A real-time deep network for scene understanding. In *Proceedings of the International Conference on Computer Vision*, pages 4154–4162. IEEE.

Eder, M., Shvets, M., Lim, J., and Frahm, J.-M. (2020). Tangent images for mitigating spherical distortion. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 12426–12434. IEEE/CVF.

EpicGames (2020). Unreal engine 4 documentation.

Facil, J. M., Ummenhofer, B., Zhou, H., Montesano, L., Brox, T., and Civera, J. (2019). Cam-convs: Camera-aware multi-scale convolutions for single-view depth. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 11826–11835. IEEE/CVF.

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

Fouhey, D. F., Delaitre, V., Gupta, A., Efros, A. A., Laptev, I., and Sivic, J. (2014). People watching: Human actions as a cue for single view geometry. *International Journal of Computer Vision*, pages 259–274.

Fraguas Bordonaba, E. and Bermúdez Cameo, J. (2020). Estimación de línea del horizonte en imágenes de 360 grados con aprendizaje automático profundo. *Master thesis*.

Fukano, K., Mochizuki, Y., Iizuka, S., Simo-Serra, E., Sugimoto, A., and Ishikawa, H. (2016). Room reconstruction from a single spherical image by higher-order energy minimization. In *Proceedings of the International Conference on Pattern Recognition*, pages 1768–1773. IEEE.

Gao, S., Yang, K., Shi, H., Wang, K., and Bai, J. (2022). Review on panoramic imaging and its applications in scene understanding. *Transactions on Instrumentation and Measurement*, 71:1–34.

Gasparini, S. and Caglioti, V. (2011). Line localization from single catadioptric images. *International Journal of Computer Vision*, pages 361–374.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, pages 1231–1237.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 580–587. IEEE/CVF.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.

Greene, N. (1986). Environment mapping and other applications of world projections. *Computer Graphics and Applications*, pages 21–29.

Guerrero-Viu, J., Fernandez-Labrador, C., Demonceaux, C., and Guerrero, J. J. (2020). What's in my room? object recognition on indoor panoramic images. In *International Conference on Robotics and Automation*, pages 567–573. IEEE.

Gupta, R. and Hartley, R. I. (1997). Linear pushbroom cameras. *Transactions on Pattern Analysis and Machine Intelligence*, pages 963–975.

Hadj-Abdelkader, H., Tahri, O., and Benseddik, H.-E. (2018). Closed form solution for rotation estimation using photometric spherical moments. In *International Conference on Intelligent Robots and Systems*, pages 627–634. IEEE.

Haggui, O., Bayd, H., Magnier, B., and Aberkane, A. (2021). Human detection in moving fisheye camera using an improved yolov3 framework. In *International Workshop on Multimedia Signal Processing*, pages 1–6. IEEE.

He, K., Gkioxari, G., Dollar, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the International Conference on Computer Vision*, pages 2961–2969. IEEE.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 770–778. IEEE/CVF.

Heo, M., Lee, J., Kim, K.-R., Kim, H.-U., and Kim, C.-S. (2018). Monocular depth estimation using whole strip masking and reliability-based refinement. In *Proceedings of the European Conference on Computer Vision*, pages 36–51. Springer.

Hermans, A., Floros, G., and Leibe, B. (2014). Dense 3d semantic mapping of indoor scenes from rgb-d images. In *International Conference on Robotics and Automation*, pages 2631–2638. IEEE.

Hernandez-Olivan, C. and Beltran, J. R. (2022). Music composition with deep learning: A review. *Advances in speech and music technology: computational aspects and applications*, pages 25–50.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jeon, Y. and Kim, J. (2017). Active convolution: Learning the shape of convolution for image classification. In *Proceedings of Conference on Computer Vision and Pattern Recognition*, pages 4201–4209. IEEE/CVF.

Jiang, C., Huang, J., Kashinath, K., Marcus, P., Niessner, M., et al. (2019). Spherical cnns on unstructured grids. *arXiv preprint arXiv:1901.02039*.

Jiang, H., Sheng, Z., Zhu, S., Dong, Z., and Huang, R. (2021). Unifuse: Unidirectional fusion for 360 panorama depth estimation. *Robotics and Automation Letters*, 6(2):1519–1526.

Jiang, P., Ergu, D., Liu, F., Cai, Y., and Ma, B. (2022). A review of yolo algorithm developments. *Procedia Computer Science*, 199:1066–1073.

Johnson-Roberson, M., Barto, C., Mehta, R., Sridhar, S. N., Rosaen, K., and Vasudevan, R. (2016). Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? *arXiv preprint arXiv:1610.01983*.

Jung, C. and Kim, C. (2012). Real-time estimation of 3d scene geometry from a single image. *Pattern Recognition*, pages 3256–3269.

Kannala, J. and Brandt, S. S. (2006). A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *Transactions on Pattern Analysis and Machine Intelligence*, pages 1335–1340.

Karsch, K., Hedau, V., Forsyth, D., and Hoiem, D. (2011). Rendering synthetic objects into legacy photographs. *Transactions on Graphics*, pages 1–12.

Kim, D., Pathak, S., Moro, A., Yamashita, A., and Asama, H. (2021). Self-supervised optical flow derotation network for rotation estimation of a spherical camera. *Advanced Robotics*, 35(2):118–128.

Kingslake, R. (1989). *A history of the photographic lens*. Academic Press.

Kukelova, Z., Bujnak, M., and Pajdla, T. (2011). Polynomial eigenvalue solutions to minimal problems in computer vision. *Transactions on Pattern Analysis and Machine Intelligence*, pages 1381–1393.

Kumar, V. R., Milz, S., Witt, C., Simon, M., Amende, K., Petzold, J., Yogamani, S., and Pech, T. (2018). Monocular fisheye camera depth estimation using sparse lidar supervision. In *International Conference on Intelligent Transportation Systems*, pages 2853–2858. IEEE.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

Lee, J.-H. and Kim, C.-S. (2019). Monocular depth estimation using relative depth maps. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 9729–9738. IEEE/CVF.

Lee, J. M. and Lee, J. M. (2012). *Smooth manifolds*. Springer.

Li, Y., Guo, Y., Yan, Z., Huang, X., Duan, Y., and Ren, L. (2022). Omnifusion: 360 monocular depth estimation via geometry-aware fusion. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 2801–2810. IEEE/CVF.

Li, Y., Shum, H.-Y., Tang, C.-K., and Szeliski, R. (2004). Stereo reconstruction from multiperspective panoramas. *Transactions on Pattern Analysis and Machine Intelligence*, pages 45–62.

Lin, S.-S. and Bajcsy, R. (2006). Single-view-point omnidirectional catadioptric cone mirror imager. *Transactions on Pattern Analysis and Machine Intelligence*, pages 840–845.

Lopez-Nicolas, G. and Sagues, C. (2014). Unitary torus model for conical mirror based catadioptric system. *Computer Vision and Image Understanding*, pages 67–79.

Makadia, A. and Daniilidis, K. (2006). Rotation recovery from spherical images without correspondences. *IEEE transactions on pattern analysis and machine intelligence*, 28(7):1170–1175.

McCormac, J., Handa, A., Leutenegger, S., and Davison, A. J. (2017). Scenenet rgb-d: Can 5m synthetic images beat generic imagenet pre-training on indoor segmentation? In *Proceedings of the International Conference on Computer Vision*, pages 2678–2687. IEEE.

MegaPOV (2020). Megapov.

Menem, M. and Pajdla, T. (2004). Constraints on perspective images and circular panoramas. In *British Machine Vision Conference*, pages 1–10.

Meng, M., Xiao, L., Zhou, Y., Li, Z., and Zhou, Z. (2021). Distortion-aware room layout estimation from a single fisheye image. In *International Symposium on Mixed and Augmented Reality*, pages 441–449. IEEE.

Mur-Artal, R., Montiel, J. M. M., and Tardos, J. D. (2015). Orb-slam: a versatile and accurate monocular slam system. *Transactions on Robotics*, pages 1147–1163.

Naseer, M., Khan, S., and Porikli, F. (2018). Indoor scene understanding in 2.5/3d for autonomous agents: A survey. *IEEE Access*, pages 1859–1887.

Ni, D., Ji, P., and Song, A. (2019). Vanishing point detection in corridor for autonomous mobile robots using monocular low-resolution fisheye vision. *Advances in Mechanical Engineering*, 11(10):1687814019884767.

Perez-Yus, A., Lopez-Nicolas, G., and Guerrero, J. J. (2019). Scaled layout recovery with wide field of view rgb-d. *Image and Vision Computing*, 87:76–96.

Pintore, G., Agus, M., Almansa, E., Schneider, J., and Gobbetti, E. (2021a). Slicenet: deep dense depth estimation from a single indoor panorama using a slice-based representation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 11536–11545. IEEE/CVF.

Pintore, G., Agus, M., and Gobbetti, E. (2020). Atlantanet: Inferring the 3d indoor layout from a single 360 image beyond the manhattan world assumption. In *Proceedings of the European Conference on Computer Vision*, pages 432–448. Springer.

Pintore, G., Almansa, E., Agus, M., and Gobbetti, E. (2021b). Deep3dlayout: 3d reconstruction of an indoor layout from a spherical panoramic image. *Transactions on Graphics*, pages 1–12.

Pottmann, H. and Wallner, J. (2001). *Computational Line Geometry*. Springer Science & Business Media.

POV-Ray (2020). The persistence of vision raytracer.

Puig, L., Bermudez, J., Sturm, P., and Guerrero, J. J. (2012). Calibration of omnidirectional cameras in practice: A comparison of methods. *Computer Vision and Image Understanding*, pages 120–137.

Qiu, W., Zhong, F., Zhang, Y., Qiao, S., Xiao, Z., Kim, T. S., and Wang, Y. (2017). Unrealcv: Virtual worlds for computer vision. In *Proceedings of the International Conference on Multimedia*, pages 1221–1224. ACM.

Ranftl, R., Vineet, V., Chen, Q., and Koltun, V. (2016). Dense monocular depth estimation in complex dynamic scenes. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 4058–4066. IEEE/CVF.

Rao, S., Kumar, V., Kifer, D., Giles, C. L., and Mali, A. (2021). Omnilayout: Room layout reconstruction from indoor spherical panoramas. In *Proceedings of the Conference on Computer Vision and Pattern Recognition Workshops*, pages 3706–3715. IEEE.

Richter, S. R., Hayder, Z., and Koltun, V. (2017). Playing for benchmarks. In *Proceedings of the International Conference on Computer Vision*, pages 2213–2222. IEEE.

Richter, S. R., Vineet, V., Roth, S., and Koltun, V. (2016). Playing for data: Ground truth from computer games. In *Proceedings of the European Conference on Computer Vision*, pages 102–118. Springer.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 10684–10695. IEEE/CVF.

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

Ros, G., Sellart, L., Materzynska, J., Vazquez, D., and Lopez, A. M. (2016). The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 3234–3243. IEEE/CVF.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, pages 211–252.

Russell, B. C., Torralba, A., Murphy, K. P., and Freeman, W. T. (2008). Labelme: a database and web-based tool for image annotation. *International Journal of Computer Vision*, pages 157–173.

Sanchez-Garcia, M., Martinez-Cantin, R., and Guerrero, J. J. (2019). Indoor scenes understanding for visual prosthesis with fully convolutional networks. In *Visigrapp*, pages 218–225.

Scaramuzza, D., Martinelli, A., and Siegwart, R. (2006a). A flexible technique for accurate omnidirectional camera calibration and structure from motion. In *Proceedings of the International Conference on Computer Vision Systems*, pages 45–45. IEEE.

Scaramuzza, D., Martinelli, A., and Siegwart, R. (2006b). A toolbox for easily calibrating omnidirectional cameras. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 5695–5701. IEEE.

Schlegel, D., Colosi, M., and Grisetti, G. (2018). Proslam: Graph slam from a programmer's perspective. In *Proceedings of the International Conference on Robotics and Automation*, pages 1–9. IEEE.

Schneider, D., Schwalbe, E., and Maas, H.-G. (2009). Validation of geometric models for fisheye lenses. *ISPRS Journal of Photogrammetry and Remote Sensing*, pages 259–266.

Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *Transactions on Signal Processing*, pages 2673–2681.

Sekkat, A. R., Dupuis, Y., Vasseur, P., and Honeine, P. (2020). The omniscape dataset. In *Proceedings of the International Conference on Robotics and Automation*, pages 1603–1608. IEEE.

Sharif Razavian, A., Azizpour, H., Sullivan, J., and Carlsson, S. (2014). Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of Conference on Computer Vision and Pattern Recognition workshops*, pages 806–813. IEEE/CVF.

Shum, H.-Y. and Szeliski, R. (1999). Stereo reconstruction from multiperspective panoramas. In *Proceedings of the International Conference on Computer Vision*, pages 14–21. IEEE.

Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. (2012). Indoor segmentation and support inference from rgbd images. In *Proceedings of the European Conference on Computer Vision*, pages 746–760. Springer.

Song, S., Lichtenberg, S. P., and Xiao, J. (2015). Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 567–576. IEEE/CVF.

Song, S. and Xiao, J. (2016). Deep sliding shapes for amodal 3d object detection in rgb-d images. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 808–816. IEEE/CVF.

Straub, J., Whelan, T., Ma, L., Chen, Y., Wijmans, E., Green, S., Engel, J. J., Mur-Artal, R., Ren, C., Verma, S., et al. (2019). The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*.

Su, Y.-C. and Grauman, K. (2017). Learning spherical convolution for fast features from 360 imagery. *Advances in Neural Information Processing Systems*, 30.

Sumikura, S., Shibuya, M., and Sakurada, K. (2019). Openvslam: A versatile visual slam framework. In *Proceedings of the International Conference on Multimedia*, pages 2292–2295. ACM.

Sun, C., Hsiao, C.-W., Sun, M., and Chen, H.-T. (2019). Horizonnet: Learning room layout with 1d representation and pano stretch data augmentation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 1047–1056. IEEE/CVF.

Sun, C., Sun, M., and Chen, H.-T. (2021). Hohonet: 360 indoor holistic understanding with latent horizontal features. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 2573–2582. IEEE/CVF.

Suvorov, R., Logacheva, E., Mashikhin, A., Remizova, A., Ashukha, A., Silvestrov, A., Kong, N., Goka, H., Park, K., and Lempitsky, V. (2022). Resolution-robust large mask inpainting with fourier convolutions. In *Proceedings on the Winter Conference of Applications of Computer Vision*, pages 2149–2159. IEEE.

Tateno, K., Navab, N., and Tombari, F. (2018). Distortion-aware convolutional filters for dense prediction in panoramic images. In *Proceedings of the European Conference on Computer Vision*, pages 707–722. Springer.

Teller, S. and Hohmeyer, M. (1999). Determining the lines through four lines. *Journal of graphics tools*, pages 11–22.

Tian, J., Mithun, N. C., Seymour, Z., Chiu, H.-P., and Kira, Z. (2022). Striking the right balance: Recall loss for semantic segmentation. In *Proceedings of the International Conference on Robotics and Automation*. IEEE.

Toromanoff, M., Wirbel, E., Wilhelm, F., Vejarano, C., Perrotton, X., and Moutarde, F. (2018). End to end vehicle lateral control using a single fisheye camera. In *International Conference on Intelligent Robots and Systems*, pages 3613–3619. IEEE/RSJ.

Unity (2020). Unity engine.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Conference on Neural Information Processing Systems*. NeurIPS Foundation.

Wang, F.-E., Yeh, Y.-H., Sun, M., Chiu, W.-C., and Tsai, Y.-H. (2020). Bifuse: Monocular 360 depth estimation via bi-projection fusion. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 462–471. IEEE/CVF.

Wang, F.-E., Yeh, Y.-H., Tsai, Y.-H., Chiu, W.-C., and Sun, M. (2022). Bifuse++: Self-supervised and efficient bi-projection fusion for 360 depth estimation. *Transactions on Pattern Analysis and Machine Intelligence*.

Wang, Z., Song, R., Duan, P., and Li, X. (2021). Efnet: Enhancement-fusion network for semantic segmentation. *Pattern Recognition*, page 108023.

Wei, H. and Wang, L. (2018). Understanding of indoor scenes based on projection of spatial rectangles. *Pattern Recognition*, pages 497–514.

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

Zhang, C., Wang, L., and Yang, R. (2010). Semantic segmentation of urban scenes using dense depth maps. In *Proceedings of the European Conference on Computer Vision*, pages 708–721. Springer.

Zhang, J., Yang, K., Ma, C., Reiss, S., Peng, K., and Stiefelhagen, R. (2022). Bending reality: Distortion-aware transformers for adapting to panoramic semantic segmentation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 16917–16927. IEEE/CVF.

Zhang, Y., Song, S., Tan, P., and Xiao, J. (2014). Panocontext: A whole-room 3d context model for panoramic scene understanding. In *Proceedings of the European Conference on Computer Vision*, pages 668–686. Springer.

Zhang, Z., Cui, Z., Xu, C., Jie, Z., Li, X., and Yang, J. (2018). Joint task-recursive learning for semantic segmentation and depth estimation. In *Proceedings of the European Conference on Computer Vision*, pages 235–251. Springer.

Zheng, S., Cheng, M.-M., Warrell, J., Sturgess, P., Vineet, V., Rother, C., and Torr, P. H. S. (2014). Dense semantic image segmentation with objects and attributes. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 3214–3221. IEEE/CVF.

Zhou, Q., Wu, X., Zhang, S., Kang, B., Ge, Z., and Latecki, L. J. (2022). Contextual ensemble network for semantic segmentation. *Pattern Recognition*, page 108290.

Zioulis, N., Karakottas, A., Zarpalas, D., Alvarez, F., and Daras, P. (2019). Spherical view synthesis for self-supervised 360 depth estimation. In *International Conference on 3D Vision*, pages 690–699. IEEE.

Zioulis, N., Karakottas, A., Zarpalas, D., and Daras, P. (2018). Omnidepth: Dense depth estimation for indoors spherical panoramas. In *Proceedings of the European Conference on Computer Vision*, pages 448–465. Springer.

Zou, C., Su, J.-W., Peng, C.-H., Colburn, A., Shan, Q., Wonka, P., Chu, H.-K., and Hoiem, D. (2021). Manhattan room layout reconstruction from a single 360 image: A comparative study of state-of-the-art methods. *International Journal of Computer Vision*, pages 1410–1431.

Appendix A

Dataset of non-central panoramas

A.1 Introduction

Omnidirectional images are one of the main sources of information for learning based scene understanding algorithms. However, annotated datasets of omnidirectional images cannot keep the pace of these learning based algorithms development. Among the different panoramas and in contrast to standard central ones, non-central panoramas provide geometrical information in the distortion of the image from which we can retrieve 3D information of the environment [2]. However, due to the lack of commercial non-central devices, up until now there was no dataset of these kinds of panoramas. In this data paper, we present the first dataset of non-central panoramas for indoor scene understanding. The dataset is composed by **2574** RGB non-central panoramas taken in around 650 different rooms. Each panorama has associated a depth map and annotations to obtain the layout of the room from the image as a structural edge map, list of corners in the image, the 3D corners of the room and the camera pose. The images are taken from photorealistic virtual environments and pixel-wise automatically annotated.

A.2 Specifications

Subject	Computer Science: Computer Vision and Pattern Recognition
Specific subject area	Non-central circular panoramas for indoor scene understanding.
Type of data	RGB Image (.png) Colour code depth maps (.png) Layout annotations (.png, .npy, .txt, .mat)
How the data were acquired	Random generation of virtual environments. Ad-hoc programmable camera projection model for image rendering via ray tracing. The RGB images are rendered with POV-Ray[1] and the depth maps with Mega-POV[2]. Layout annotations are obtained from the 3D model of the virtual environment.
Data format	Raw Filtered
Description of data collection	From the generated virtual environments, we randomly place the non-central acquisition system in different locations inside the environment. The radius of acquisition is 1 meter. The size of the panoramas is 1024x512 pixels. Once acquired the non-central panoramas, we exclude those that will be physically impossible to acquire in a real situation (i.e. the non-central acquisition system goes through an object or a wall, creating a black hole in the image).
Data accessibility	URL to Github repository: https://github.com/jesusbermudezcameo/NonCentralIndoorDataset
Related research article	B. Berenguel-Baeta, J. Bermudez-Cameo and J.J. Guerrero, Atlanta Scaled Layouts from Non-central Panoramas. Pattern Recognition (2022). DOI:https://doi.org/10.1016/j.patcog.2022.108740

[1]The Persistence of Vision Raytracer.http://www.povray.org (accessed May 2022)
[2]MegaPOV. http://megapov.inertart.net (accessed May 2022)

A.3 Value of the data

- The presented dataset is the first existing dataset with non-central panoramas. Besides it includes annotations for different purposes as layout recovery, line extraction and depth estimation.
- Researchers who want to take advantage of the geometrical properties of non-central systems can find in this dataset a perfect source of information for evaluation and development of new algorithms.
- Since it is the first non-central dataset, it can be used to adapt existing algorithms for omnidirectional central images to the no-central case. Besides, in the related research [1], only the RGB images and layout annotations have been used, leaving the depth maps for future research topics.

A.4 Data description

The dataset contains a set of gravity oriented panoramas. We make this specification since non-central panoramas cannot be rotated as a data augmentation in a different axis that the revolution axis of the non-central system. The dataset includes the folders: *img* contains the RGB non-central panoramas are located; *depth_coded* contains depth maps coded in 3 channels (RGB channels) associated with the RGB panoramas; *EM_gt* contains one channel images where the structural lines of the environments are defined; *DataPython* and *DataMatLab* contains ground truth information used to evaluate the work presented in [1], including the 3D position of the corners of the room (*3D_gt*), the camera location (*cam_pose*), the labelling for the floor-wall and ceiling-wall intersections as spherical coordinates of the projecting rays (*label_ang*) and the pixel coordinates of the 3D corners in the panoramas (*label_cor*).

The main folder follows the following distribution.

- `NonCentralIndoorDataset`
    - img
    - depth_coded
    - EM_gt
    - DataPython
        - *3D_gt*
        - *cam_pose*
        - *label_ang*
        - *label_cor*
    - DataMatLab
        - *mat_gt*

A.5 Experimental design, materials and methods

The data of this dataset is obtained from environments randomly and synthetically generated. We first generate a random layout constrained by different structural limits (see Table A.2). These limits include minimum and maximum: area of the room, number of walls, wall length, angle between walls (if non-Manhattan), room height and Manhattan ratio.

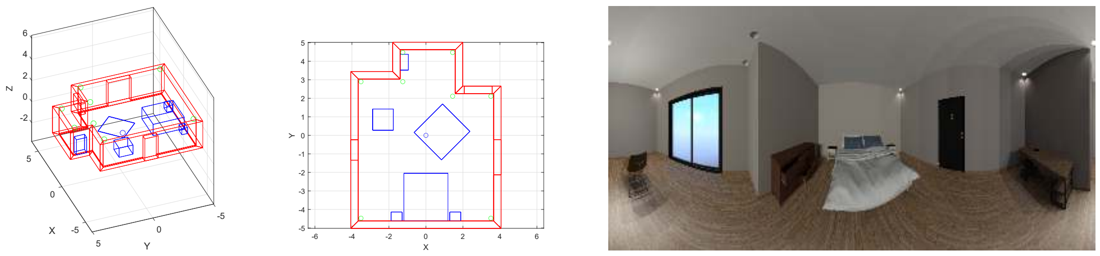
Table A.2: Layout limits for randomized generation.

Parameter	Min.	Max.
Area $[m^2]$	25	110
n. walls	4	14
wall length $[m]$	0.5	8
walls angle $[degs]$	25	100
Room height $[m]$	2.5	4.25
Manhattan ratio	0.7	

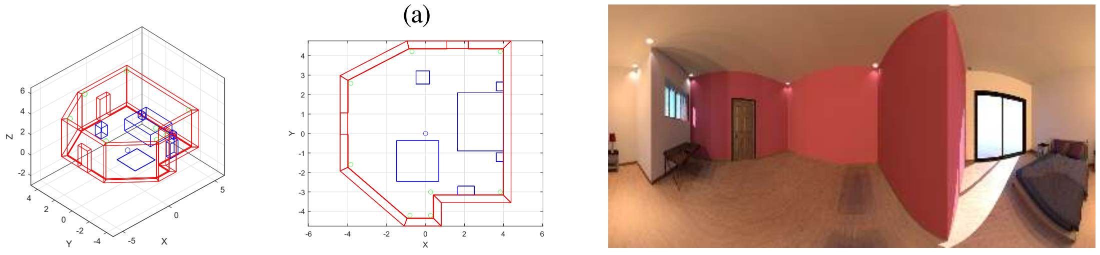
The first version of each layout is a Manhattan layout constrained by the previous limits. Then, depending on the Manhattan ratio, a set of random vertical planes clip the layout introducing oblique walls in angles with respect to the previous walls in the range of the walls angle limit. After the layout clip, we evaluate the final layout, checking if it meets the constraints. If any of the constraints is not satisfied, the layout is deleted and a new one is generated.

Once the structure of the room is obtained, we randomly set different kinds of walls containing doors, windows, colors and textures. These characteristics of the walls are randomly selected from different pools (e.g. we have different models of doors and windows to select). At this point we select the color and textures of the ceiling and floor of the room. Once defined the 3D structure of the room, we include objects in it. For that purpose, we build a free space map where we can place different kinds of objects. First we consider the kind of objects that are placed next to a wall in a fixed orientation (beds, wardrobes, desks). Then we place objects in the room in a random position and orientation (chairs, sofas, carpets). Finally, we place objects that are placed on top of other objects (cups, clocks, clothes) and we also place lighting sources for a more realistic rendering. All of these objects are randomly picked from different pools depending on the object class. Besides, the ambient illumination conditions are also randomly picked from 3 different configurations.

Once defined our room, we have all the necessary information to generate the ground truth and the labelling of the dataset (see Fig. A.1). The next step is to render the RGB non-central panoramas and generate the depth maps. The color images are rendered with the ray tracing software POV-Ray while the depth maps are obtained with the use of MegaPOV. The non-central panoramic camera is modelled by using an ad-hoc programmable camera projection model included in last versions of POV-Ray. For each scene we can render different acquisitions modifying the position and the orientation of the camera. In the case of the proposed dataset, the camera is always oriented with the gravity direction.



(a)



(b)

Figure A.1: Layout generation and non-central panorama rendering examples. (a) Manhattan random room. (b) Atlanta random room.

Notice that, by contrast with central panoramas (e.g. equirectangular images), we can not post-process a single render for obtaining different panoramas with different orientations.

About the author

Samuel Bruno Berenguel Baeta was born in Zaragoza, Spain, in 1994. He started his studies in the University of Zaragoza, recieving the Industrial Engineering degree and Industrial Engineering MSc degree in September 2017 and December 2019 respectively.

After a brief period in the private industry as engineer while finishing the MSc degree, in January 2020 he joined as researcher and PhD Student in the Aragon Institute for Engineering Research and the Department of Computer Science and Systems Engineering of the University of Zaragoza. The research developed as PhD student is detailed in this thesis dissertation, which was presented in December 2023.

The main topics of his research are: omnidirectional computer vision, deep learning and 3D scene understanding. You can find more about him and his research in:
ORCID https://orcid.org/my-orcid?orcid=0000-0003-2674-4844;
ResearchGate https://www.researchgate.net/profile/Bruno-Berenguel-Baeta
GitHub https://github.com/Sbrunoberenguel
and email samuelbrunoberenguel@gmail.com.

Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza

1542

Universidad de Zaragoza
December 2023
Zaragoza, Spain