Universidad Zaragoza
1542

Trabajo Fin de Grado

Object recognition on panoramic images

Reconocimiento de objetos en imágenes panorámicas

Autora

Julia Guerrero Viu

Directores

Clara Fernández Labrador
José Jesús Guerrero Campo

ESCUELA DE INGENIERIA Y ARQUITECTURA
2019

Agradecimientos

Quiero agradecer a mis directores, Clara y Josechu, por todo su tiempo, dedicación y ayuda. Gracias Clara por tener siempre, incluso desde Francia o California, total disponibilidad para resolver mis dudas y guiarme en mi trabajo.

Mis agradecimientos también a la Cátedra Mobility City de la Universidad de Zaragoza, que ha ayudado con su financiación a la realización de este proyecto.

Abstract

This report depicts the development of an object recognition on panoramic images system. Panoramic images, with their wide field of view, have a strong potential for successfully recognition on indoor environments as they include the whole scene context. However, they also represent a challenge to deal with the particular distortion effects, caused by their spherical projection and the non-uniform resolution, or the lack of massive labeled datasets. After an intensive research on state of the art of object recognition, and its particularities on panoramas, the lack of concrete models to work with this kind of images is detected.

In this work, a deep learning model called Panoramic BlitzNet is developed. It is based on BlitzNet network and addresses object detection and semantic segmentation tasks on equirectangular indoor panoramic images. It is composed by a Fully Convolutional Network almost completely shared for both tasks, with skip connections and which conducts multi-scale recognition. Our model is fed with complete panoramas, which is one of the keys of its success. Furthermore, the impact of equirectangular convolutions, replacing standard ones to adapt better to the spherical domain, is analyzed in depth. Additionally, an approach to instance segmentation task and a proof of concept to combine object recognition with 3D layout recovery of the rooms are also conducted and explained.

The model is trained and tested on SUN360 dataset, which has been adapted to include segmentation masks. Experimental results are satisfactory, supporting the importance of developing specific systems to work with panoramic images and the usefulness of jointly training two scene understanding tasks. Results prove the effectiveness of the model, that outperforms other state of the art methods in indoor object recognition with panoramas.

Resumen

Este documento describe el desarrollo de un sistema de reconocimiento de objetos en imágenes panorámicas. Las imágenes panorámicas, con su amplio campo de visión, suponen un gran potencial para tareas de reconocimiento en interiores, ya que incluyen el contexto completo de las escenas. Sin embargo, también representan un reto para lidiar con los efectos de distorsión, debidos a su proyección esférica y a la resolución variable, así como con la falta de datos masivos etiquetados. Tras una intensa investigación sobre el estado del arte en reconocimiento de objetos, y sus particularidades sobre panorámicas, se ha detectado la escasez de modelos concretos que trabajen con este tipo de imágenes.

En este trabajo, se ha desarrollado un modelo basado en aprendizaje profundo, llamado Panoramic BlitzNet. Utiliza como base la red BlitzNet y aborda las tareas de detección de objetos y segmentación semántica en imágenes equirectangulares de interior. Está compuesto por una red totalmente convolucional (FCN) compartida prácticamente en su totalidad por las dos tareas, que cuenta con conexiones *skip* y que realiza reconocimiento multi-escala. Este modelo trabaja con las panorámicas completas, lo que representa una de las claves de su éxito. Además, se analiza en profundidad el impacto de las convoluciones equirectangulares, que sustituyen a las convencionales para adaptarse mejor al dominio esférico. Adicionalmente, se ha llevado a cabo una aproximación a la tarea de segmentación por instancias, así como una prueba de concepto para combinar el reconocimiento de objetos con la reconstrucción del *layout* 3D en habitaciones.

El modelo se ha entrenado y evaluado utilizando el *dataset* SUN360, adaptado para incluir máscaras de segmentación. Los resultados experimentales son claramente satisfactorios, respaldando la importancia de desarrollar sistemas específicos que trabajen con imágenes panorámicas y la utilidad del aprendizaje conjunto de dos tareas de interpretación de escenas. Los resultados demuestran la efectividad del modelo, superando a otros métodos de estado del arte en reconocimiento de objetos con panorámicas en entornos de interior.

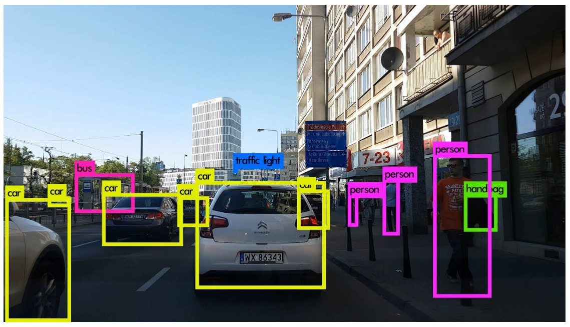
Contents

Chapter 1

Introduction

Our world is surrounded by images. Through visual sense, humans are able to obtain any kind of information from those images, and that is what Computer Vision field tries to imitate. Among this wide field, many different objectives can arise, where we can include some tasks such as face recognition, localization and mapping or 3D reconstruction.

Within this context, one of the most well-known tasks in computer vision is object recognition, which tries to identify and locate objects in images. When humans look at a photograph or watch a video, we can readily spot people, objects, scenes, or visual details. The main goal of object recognition is then to teach a computer to do what comes naturally to humans: Gaining a level of understanding of what an image contains. Object recognition is a useful task for very different applications, from autonomous cars to virtual and augmented reality.



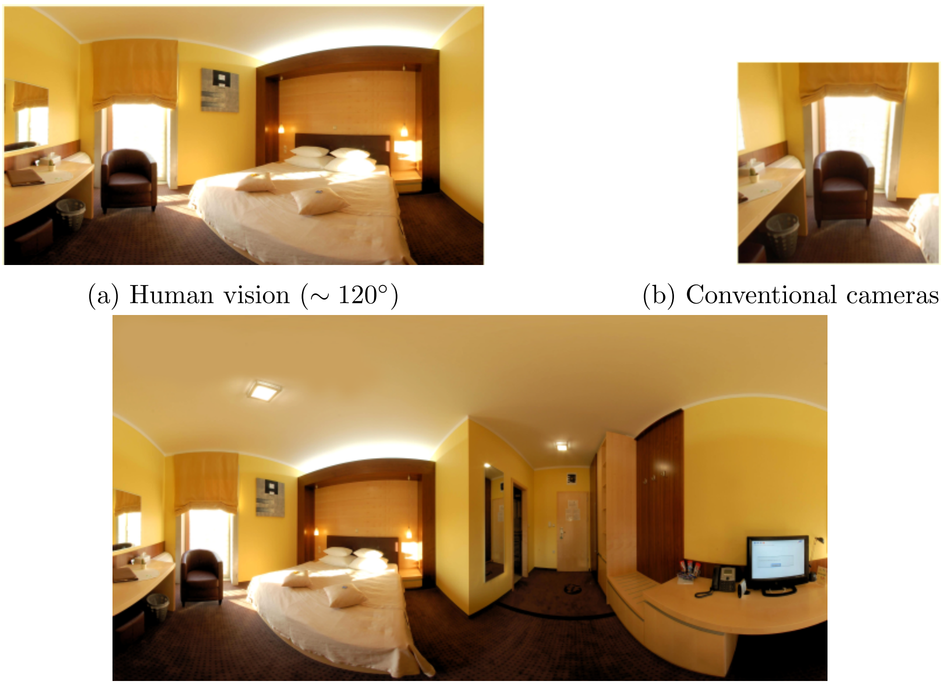
(a) Autonomous cars

(b) Augmented reality

Figure 1.1: Examples of applications from Computer Vision and Object Recognition

Since this task of recognizing a visual concept is relatively trivial for a human to perform, it is worth considering the hard challenges involved from the perspective of a Computer Vision algorithm. Objects in images can be oriented in many different ways with respect to the camera and can vary their size, both in terms of scale in the image and their real world size. They can also be occluded, blended into the environment because of their color or appearance, or affected by different illumination conditions, which change drastically their aspect on the pixel level. What's more, the concept behind an object's name is sometimes relatively broad, having many different types and appearances that could be described under the same concept, and also subjective and non-clear frontiers to other concepts. For example, where do you consider the limits between a 'sofa' and an 'armchair'? And between an 'armchair' and a 'chair'?

Dealing with those challenges is not an easy task so here it is when machine learning techniques appear, specially deep learning ones, which have become a popular method for doing object recognition. Convolutional Neural Networks (CNNs) have demonstrated to be leaders in the field, as they are capable of automatically learning objects' inherent features in order to correctly identify their intrinsic concepts.


(a) Human vision ($\sim 120°$)

(b) Conventional cameras ($\sim 40°$)

(c) Panoramas ($360°$)

Figure 1.2: Visual differences between fields of view [16]

However, images from standard cameras have a small field of view, much smaller than human vision [Figure 1.2], which implies that contextual information cannot be as useful as it should. To overcome this limitation, 360° full-view panoramic images are used, a type of images that nowadays can be easily obtained and therefore are arising more and more interest in Computer Vision research. Panoramas allow us to visualize, in a single image, the whole scene at the same time. Together with all of their potential we have to deal with challenges produced by their own spherical projection, such as distortion, or the lack of complete and massive datasets. This requires the development of specific techniques that take advantage of their strengths and allow working with panoramic images in an efficient and effective way.

1.1 Objectives

The main objective of this project is to build an object recognition system that works directly on panoramic images, taking advantage of their wide field of view and dealing with their inherent challenges. Within that big objective, the following objectives could be differentiated:

- **Research about object recognition:** Develop a deep understanding and analyze the state of the art in object recognition, with the particularities of the task on panoramic images.
- **Choose and manage a public dataset:** Study public indoor panoramic datasets and choose one for training, validation and evaluation of the system. Process dataset and its labels to adapt it for the developed system.
- **Develop a deep learning model:** Design and implement a deep learning system capable of detecting and classifying objects in panoramic images and conducting semantic segmentation of the scene.
- **Evaluate the system:** Conduct experiments to evaluate the model and compare results with the state of the art.

1.2 Outline

In this report the development of the project 'Object recognition on panoramic images' is presented. Firstly, after this short introduction, the state of the art of object recognition, with a short historical overview, as well as the use of panoramas for scene understanding are explained and analysed in Chapter 2. A short explanation about equirectangular projection, Deep Learning and Convolutional Neural Networks and the differences between the main object recognition tasks are also included. In Chapter 3 datasets used and considered are presented, together with the explanation of the SUN360 dataset processing to create segmentation masks, and to adapt it to the developed model. The design and implementation of the model are explained in Chapter 4, from the base network architecture, BlitzNet, to all the modifications conducted and the addition of equiconvs and post-processing for instance segmentation. A proof of concept of the possible combination with the layout reconstruction task is included at the end of the chapter. In Chapter 5 evaluation of the model is conducted and discussed, both for detection and segmentation tasks, comparing the results with state of the art. Finally, conclusions of the project are presented in Chapter 6, as well as a reflection about future work. Details about dataset pre-processing, complete tables along with further qualitative results, and information about timeline, tools and management of the project can be found in the appendices.

Chapter 2

State of the art

As this project includes both research on general object recognition and panoramic images, this chapter is organized in those two main sections, describing their state of the art and a short historical overview. Additionally, there is a first section that describes the essentials of deep learning and convolutional neural networks because they have are the base of all our system. As an introduction of each of the main sections, it includes a short description of the different object recognition tasks, as well as an explanation of the equirectangular projection of panoramas, to make the complete chapter easier to follow and understand.

2.1 Convolutional Neural Networks

Modern history of object recognition cannot be understood without Convolutional Neural Networks (CNNs). Artificial neural networks are machine learning models that, inspired by how the human brain works, attempt to approximate and learn any possible function by creating a network of simplified neurons, organized in layers. Each single neuron takes all its inputs and performs a linear function f as its output.

$$o = f(\mathbf{x}) = b + \mathbf{w} \cdot \mathbf{x} = b + \sum_{\forall i} w_i x_i$$

where o is the output, $\mathbf{x}$ are the inputs and $\mathbf{w}$ and b are the weights and the bias, parameters that have to be learnt from the data by the model, to minimize a loss function. Non-linearities or activation functions, such as sigmoid or Rectified Linear Unit (ReLU), are introduced to allow learning any kind of model.

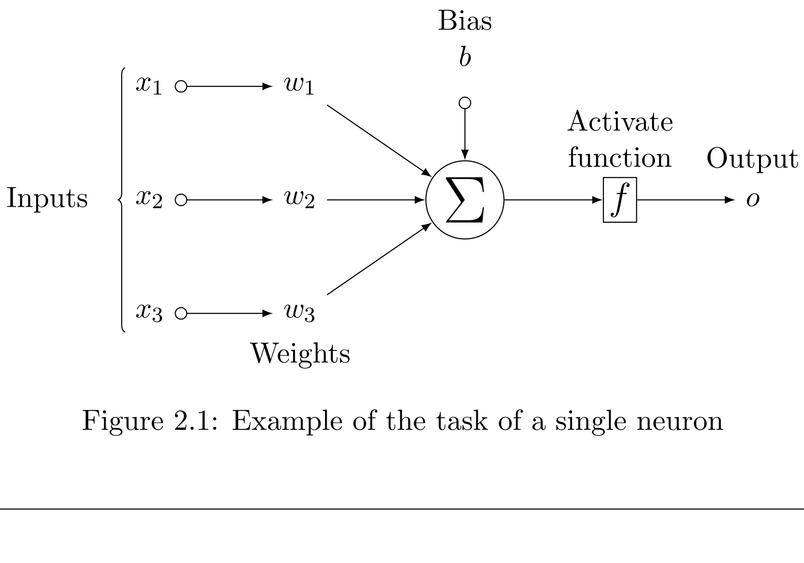


Figure 2.1: Example of the task of a single neuron

Convolutional Neural Networks (CNNs) are then very similar to ordinary neural networks. They are also made of neurons with weights and biases, have activation functions and are organized in layers. However, CNNs make the explicit assumption that the inputs are images, allowing them to encode certain properties into the architecture. What mainly differentiates CNNs from regular neural networks is the convolution operator, which allows it to learn spatial distribution, apart from making it much more efficient and vastly reducing the amount of parameters in the network, taking into account the images huge dimensionality.

Convolutions operate over 3-dimensional volumes (2D images plus number of channels) and they consist on a set of learnable filters or kernels, which are small along width and height but extend through the full depth of the image. Size of the output depends on size of input, kernel size, number of kernels, stride and padding, as the example presented in Figure 2.2. Typically, convolutions reduce dimensionality in each layer and learn features from the most general ones (such as contours, edges or corners) to the most specific ones related to the concrete task. Pooling layers, which perform a simple function such as maximum, are also usually added to the architecture to simplify data and reduce dimensions, although it is being recently substituted by bigger strides in convolutions.

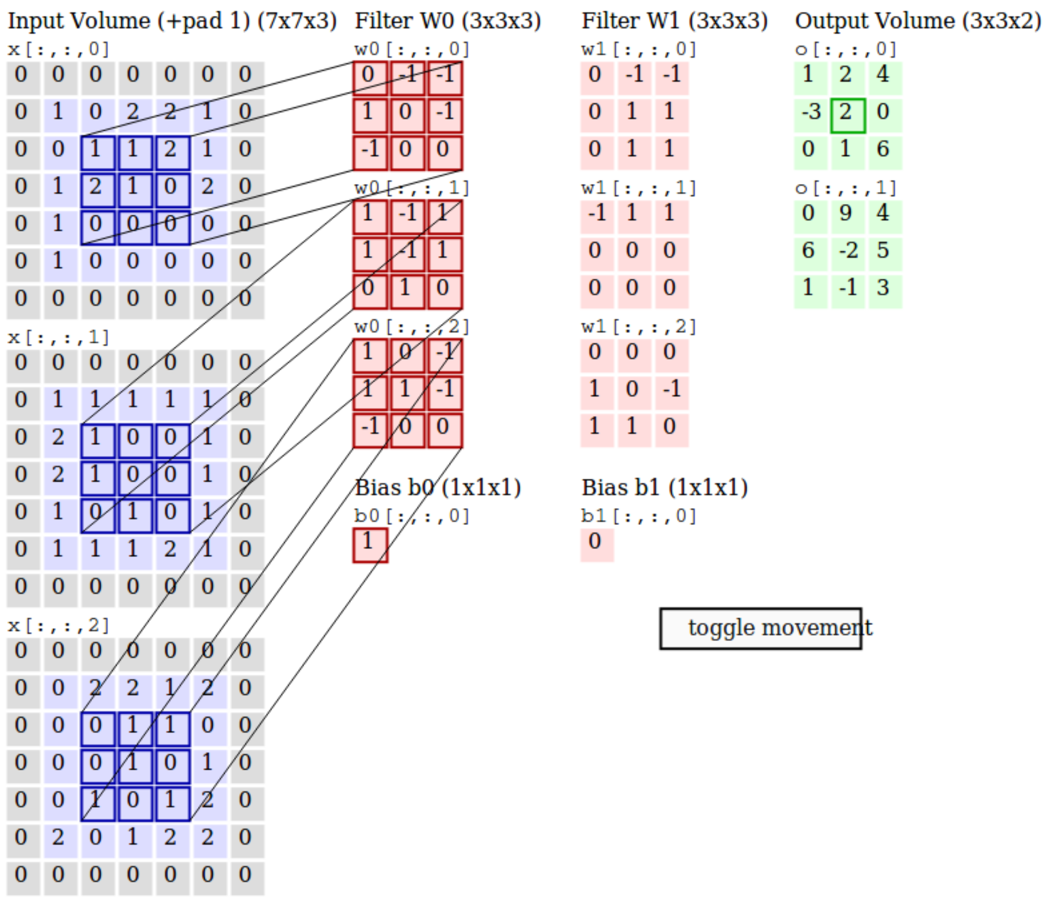
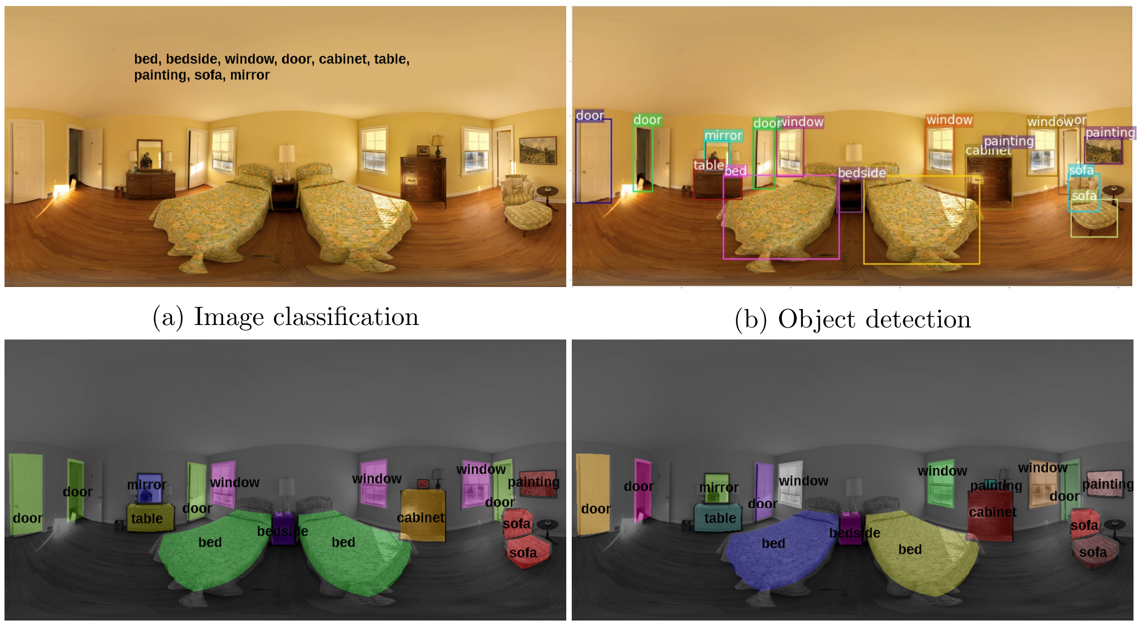


Figure 2.2: Example of a convolutional layer. Since 3D volumes are hard to visualize, all the volumes are visualized with depth slices stacked in rows. Input volume size is 5x5x3 (RGB image), with a zero-padding of size 1. There are 2 kernels of size 3x3 applied to the image with a stride of 2. Output dimensions are then reduced to 3x3x2. Figure obtained from [10].

CNNs are nowadays very popular in research community, as in conjunction with the development of very powerful GPUs, a huge number of layers can be added (hence the name Deep Learning) and computed at a reasonable amount of time. This makes CNNs the best models known up to now for many Computer Vision tasks, such as object recognition.

2.2 Object recognition

Object recognition is a general term to describe a collection of related computer vision tasks, all of which involve identifying objects in digital images. Within this general term, different tasks can be identified [18], which are described below.


bed, bedside, window, door, cabinet, table, painting, sofa, mirror
door
mirror
table
bed
window
bedside
cabinet
painting
sofa

(a) Image classification (b) Object detection

(c) Semantic segmentation (d) Instance segmentation

Figure 2.3: Evolution of different tasks in object recognition or scene understanding

As it can be seen in Figure 2.3, **image classification** is the simplest object recognition task and consists on obtaining the list of object categories that are present in an image. Image classification can also name the task of simply getting the category of the dominant object inside the image, if there is one. Meanwhile, **object localization** task finds the presence of objects in an image indicating their location with a bounding box. Further on, **object detection** mixes both previous approaches, locating the objects with a bounding box and classifying each located object category. In segmentation tasks the objective is to pixel-wise classify the image, per category in the case of **semantic segmentation** and per object in **instance segmentation**.

2.2.1 History of object recognition

The history of object recognition can be traced back to the 1960's, when Computer Vision research has its origins [2]. From the very beginning, the capacity of automatically recognizing objects has arisen human interest and has been used for many different applications, from the food industry (e.g., for the automated classification of agricultural products [3]), to the electronics and machinery industry (for automated assembly and industrial inspection purposes [36]). Even early research in character recognition for office automation related tasks [52] and biomedical research for the chromosome visual recognition task [28][21] can be arguably considered as a preamble for object recognition field.

However, the most important transformation of the field occurred with the revolution of convolutional neural networks and deep learning.

Modern history of object recognition began in 2012, when AlexNet won, by a large margin, the ILSVRC competition [44], and is still being written. It has become an alive race in research and industry for being the most accurate and fastest model, taking part in any of the main public challenges, which will be explained in section 2.2.2.

AlexNet [31] was based on the decades-older LeNet [32], combined with data augmentation, ReLU, dropout, and GPU implementation. It proved the effectiveness of CNNs, outperforming by a wide difference all other methods, and then opened a new era for Computer Vision.

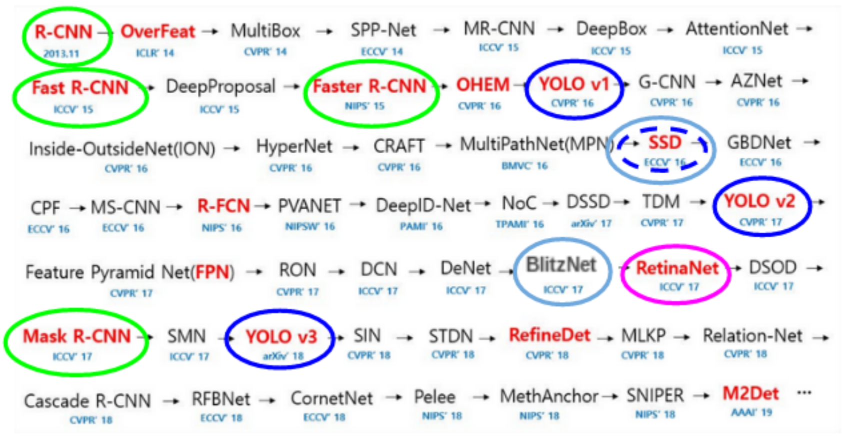


Figure 2.4: Graphical review of state of the art in object recognition (2013-2019). Highlighted models are going to be described in this section. Figure adapted from [25].

After the impact of AlexNet, Deep Learning became the prime method for modern object recognition, which state of the art progress is summarized in Figure 2.4. It includes the names of the most important models published in main Computer Vision and Machine Learning conferences. Within this general view, some 'families' or groups of models can be distinguished, opening a competition between two main approaches: direct classification or one-stage (e.g. YOLO, SSD) and refined classification or two-stages (e.g. R-CNN, Mask R-CNN).

Direct classification simultaneously regresses prior box and classifies object directly from the same input region, while the refined classification approach first regresses the prior box for a refined bounding box, and then pools the features of the refined box from a common feature volume and classify object by these features. The first approach is faster but less accurate since the features it uses to classify are not extracted exactly from the refined prior box region.

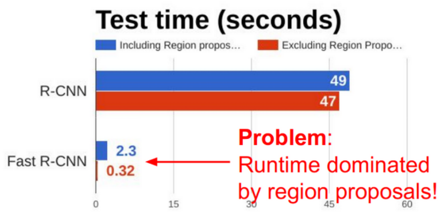
Different approaches and models are explained in the following sections, trying to maintain, when possible, their historical evolution order.

Two-stages: ’R-CNN family’

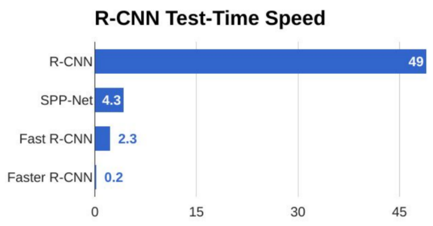
One of the first models developed for object detection was Region-based Convolutional Neural Network, R-CNN [20] in late 2013. It was two-stages method composed by a combination of a heuristic region proposal method and a CNN feature extractor. They generated $\sim$ 2000 bounding box proposals using selective search [54], which were cropped and warped to a fixed-size image to then use AlexNet to extract features for each warped image. They added a trained SVM model to classify the object in each region and multiple class-specific bounding box regressors, also trained to refine the bounding box proposals. Although it outperformed state of the art in ILSVRC 2013, it had a very high per-region computation, which made it impossible to work in real time. It also had the disadvantage of having three different models to train independently (CNN+SVM+linear regressor).

OverFeat [46], which was also based on AlexNet but using fixed evenly-spaced square windows as proposals, was two orders of magnitude faster but did not reach the same accuracy as R-CNN, whose region proposals fitted objects better.

In early 2015, Fast R-CNN [19], from the same authors, improved the system and its efficiency by using a global CNN feature extractor, based this time on SPPNet [23] and projecting proposals directly on feature map. They also added a RoIPooling and integrate all tasks into a single model with a multi-task loss.



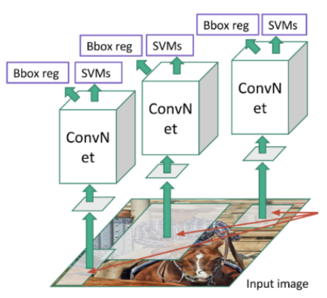
(a) Region proposal execution time vs. complete model



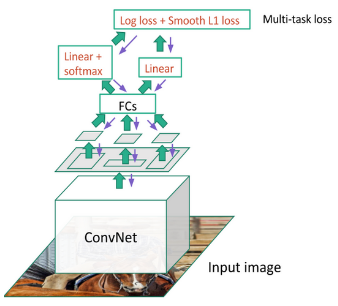
(b) Complete model execution time in test

Figure 2.5: Analysis of execution time in models from R-CNN family [43].

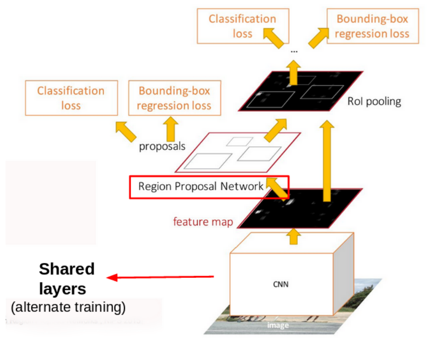
Although its performance was clearly improved, execution time was dominated by region proposals, a problem that was solved by Faster R-CNN [43], replacing selective search by a region proposal network (RPN) inspired by MultiBox [51]. Faster R-CNN proposed sliding windows with 9 prior boxes in 3 different scales and 3 different aspect ratios and was this time based on VGG16 backbone, which will be explained in next section. Both RPN and Detection CNN shared first layers of feature extraction, allowing them to benefit from each other by what they called ’alternate training’. The evolution of architectures in R-CNN family can be seen in Figure 2.6 and execution time analysis is shown in Figure 2.5.



(a) R-CNN



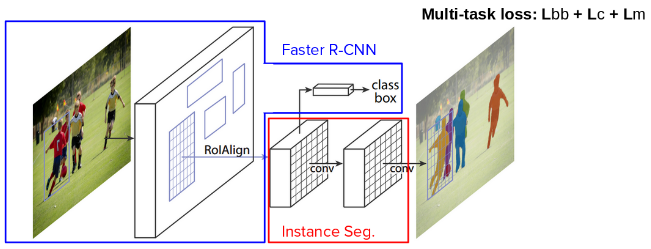
(b) Fast R-CNN



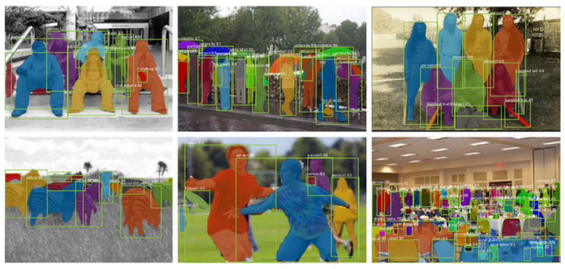
(c) Faster R-CNN

Figure 2.6: Evolution of architectures in R-CNN family models. Notice main differences between R-CNN and Fast R-CNN such as the global CNN feature extractor or multi-task loss, and between Fast R-CNN and Faster R-CNN with the introduction of the RPN instead of heuristic method selective search. Figure adapted and combined from [6] [43].

All already commented models worked only on object detection, locating objects with bounding boxes. In 2017, Mask R-CNN [22], extended Faster RCNN for Instance Segmentation by adding a branch for predicting class-specific and pixel-wise object mask, in parallel with the existing bounding box regressor and object classifier. Since RoIPool was not designed for pixel-to-pixel alignment between network inputs and outputs, Mask R-CNN replaced it with a new method called RoIAlign, based on bilinear interpolation to compute the exact values of the input features at each sub-window. Results outperformed all existing models in state of the art of both object detection and instance segmentation and are shown together with the architecture in Figure 2.7.



(a) Architecture diagram



(b) Results of both object detection and instance segmentation. Notice the high accuracy in both tasks.

Figure 2.7: Mask R-CNN architecture and results [22].

All models of this R-CNN family follow the refined classification approach, with higher levels of accuracy but slower models.

Backbones: VGG, Inception and ResNet

In 2014, VGG [47] created an architecture that, although did not outperform or win a famous challenge, is still one of the most common architectures used as a base in object recognition models, thanks to its simplicity and effectiveness. It was made of 16 convolutional layers with a very uniform architecture. Its main contribution was to replace large-kernels by stacking several small-kernels. This idea proved to work very well to extract main features in images, although the high number of parameters (138 million) could be a bit challenging to handle.

The winner of the ILSVRC 2014 competition was, however, GoogleNet [50] (Inception V1). It achieved a top-5 error rate of 6.67%, very close to human level performance. Their architecture consisted of a 22 layer deep CNN but reduced the number of parameters from 60 million (AlexNet) to 4 million. Instead of traditionally stacking up convolutional and maxpooling layers sequentially, it stacked up 'Inception modules', which consisted of multiple parallel convolutional and maxpooling layers with different kernel sizes. It used also 1x1 convolutional layer (the network in network idea) to reduce the depth of feature volume output. There are currently 4 Inception versions and is also one of the most typical feature extractors used in object detectors.

In 2015, with CNNs becoming deeper and deeper, ResNet [24] revolutionary idea outperformed state of the art and became the preferred backbone architecture for this kind of models. Deeper architectures seemed to be very promising for object recognition tasks. However, they had a major problem known as degradation: when they started converging, accuracy got saturated and then degraded too rapidly. ResNet solved this issue by adding what they called residual blocks: Instead of learning a direct mapping of input-output with a function $F(x)$ that is composed by a few stacked non-linear layers, a residual function is defined $H(x) = F(x) - x$, which can be reformed into $F(x) = H(x) + x$, where $H(x)$ represents the stacked non-linear layers and x the identity function (input=output). The intuitive idea behind it is that the network can also learn the identity, and therefore decide whether to use or not to use that layer, learning not only the parameters of the network but also its own depth. In ResNet output of each block is defined by:

$$o = F(\mathbf{x}, W) + W_s \cdot \mathbf{x}$$

being W_s equal to identity if dimensions do not change, or parameters needed just to maintain correct dimensions by only padding or conducting 1x1 convolutions.

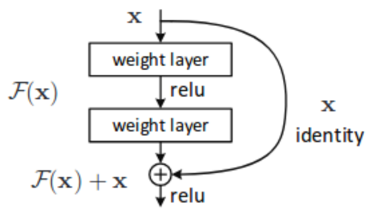

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2.8: Residual learning. Graphical representation of a residual building block.

ResNet won the ILSVRC 2015 competition with an unbelievable 3.6% error rate (human performance is established in 5.1%), and gained more and more popularity in the research community, becoming the most important feature extractor backbone for many Computer Vision tasks. Due to its high popularity, its architecture has been studied in depth and many variations have been proposed, both by changing the number of layers (ResNet18, ResNet34, ResNet50, ResNet101...) or reorganizing the architecture (ResNeXt [55] or DenseNet [26]).

One-stage: YOLO, SSD and BlitzNet

On the other hand, one-stage models, which follow the direct classification approach, started with YOLO (You Only Look Once) in 2015 [39]. It was a direct development of MultiBox [51], turning it from a region proposal solution to a complete object recognition method. What mainly distinguished YOLO was the method for obtaining prior bounding boxes or proposals. They divided the input image into a regular grid, where each grid cell was considered as a prior box. If the center of an object fell into a grid cell, that grid cell was responsible for detecting that object, holding exclusively their center location information, and predicting the object size independently, as it is graphically shown in Figure 2.9. In this method, like MultiBox, all the box regressor, confidence scorer and object classifier look at features extracted from the whole image, giving important context information. YOLO outperformed state of the art in terms of execution time with reasonable accuracy, but had limitations with variated aspect ratios and small objects.

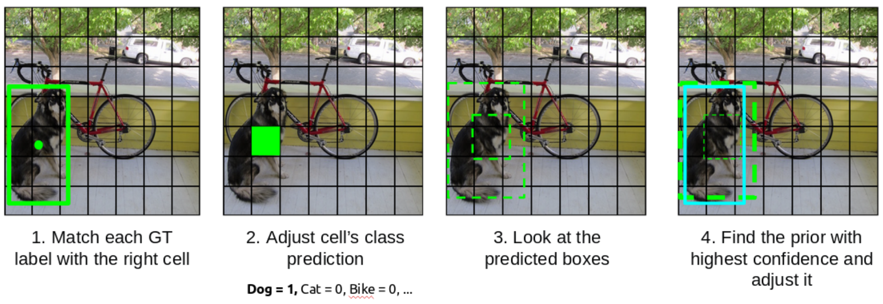

1. Match each GT label with the right cell
2. Adjust cell's class prediction
Dog = 1, Cat = 0, Bike = 0, ...
3. Look at the predicted boxes
4. Find the prior with highest confidence and adjust it

Figure 2.9: Graphical explanation of grid prior box proposals in YOLO during training [29].

In late 2016, SSD [35], based on VGG16 backbone and Faster R-CNN's RPN but following a direct classification approach with grid proposals like YOLO, reached new records in terms of performance and precision for object detection tasks. Its main characteristic was to introduce much more diversity on prior boxes by considering variate aspect ratios and by running intermediate predictions on multiple convolutional layers, at different scales or depth levels.

This was a key idea for improvements in YOLO v2 [41], which was also turned into a Fully Convolutional model. It was also the base of another model, BlitzNet [12], which apart from doing object detection intended to predict pixel-wise semantic segmentation of the image. BlitzNet was motivated for addressing both object recognition problems simultaneously, by using ResNet as a feature extractor and SSD approach to perform object recognition in one-stage and with real-time speed. They added a deconvolutional block for addressing precise localization and, inspired from DSSD [17], introduced a mechanism with skip connections to combine feature maps from the downscale and up-scale streams.

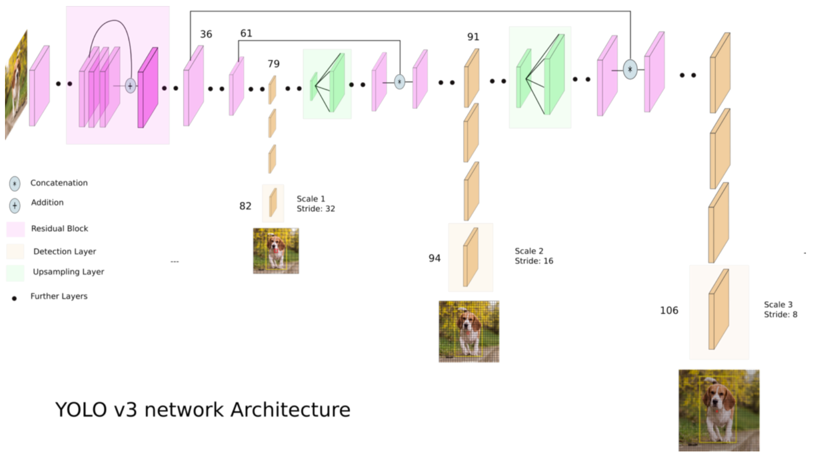


Figure 2.10: YOLO v3 architecture [29].

More recently, in 2018, YOLO v3 [42] appeared, with a renovated and deeper architecture, fully convolutional, with skip connections and upsampling layers [Figure 2.10]. They substituted pooling by convolutions and added different grid-scales, apart from intermediate predictions. YOLO is still considered an state of the art method for object detection, clearly standing out in terms of execution time, but also maintaining an outperforming level of accuracy.

RetinaNet and Focal Loss

In 2017, RetinaNet [33] was proposed, whose main impact came from the identification of a major issue in object detection task: Class imbalance. They specially focused on foreground-background class imbalance and proposed to reshape the standard cross entropy loss such that it down-weighted the loss assigned to well-classified examples:

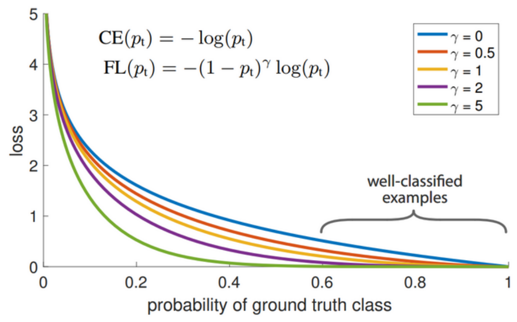


Figure 2.11: Focal Loss adds the factor $(1 - p_t)^\gamma$ to the standard cross entropy criterion, to reduce (with $\gamma > 0$) the relative loss for well-classified examples, such as easy background regions, putting more focus on hard, misclassified examples [33].

Their novel Focal Loss [Figure 2.11] focused training on a sparse set of hard examples and prevented the vast number of easy negatives from overwhelming the detector during training, proving a clear improvement in their model. RetinaNet matched the speed of previous one-stage detectors while surpassing the accuracy of existing two-stage methods.

2.2.2 Challenges

As already mentioned, object recognition has become a competition in research for becoming the most accurate and fastest model. Therefore, there are currently many public challenges, sometimes organized for main conferences such as CVPR or ICCV, where a dataset is prepared to train and test the models and several submissions all over the world compete under some clear evaluation rules. Some of the main challenges and associated datasets are presented here, together with some results of their winners.

- **ILSVRC (ImageNet Large Scale Visual Recognition Challenge)** [44]: It is one of the oldest and the competition par excellence in object recognition, although is not alive anymore. ImageNet dataset is a huge general visual database with more than 14 million images and more than 20000 categories. ILSVRC was an annual competition based on subsets of ImageNet, which started in 2010 with image classification and was extended to other tasks such as object localization and object detection. In 2017, more than 75% of participants achieved less than 0.05 error rate so organizers decided to finish the challenge. Maximum mean Average Precision (mAP) [See Section 5.1] obtained by the last winner was 0.7322.

- **PASCAL-VOC Challenge (Visual Object Classes)** [13]: VOC-Challenge is an old object recognition challenge, which started in 2005 with a very simple dataset and continued until 2012, including object detection and semantic segmentation tasks, among others. Although not deep learning models competed on it, the PASCAL-VOC dataset (mainly 2007 and 2012 versions) is still one of the most popular ones used to compare state of the art methods. This dataset contains more than 11000 images of 20 different general categories. Maximum mAP obtained by the last winner was 0.8219.

- **COCO Challenge (Common Objects in Context)** [34]: COCO annual challenge started in 2015 and is still alive. It covers different tasks, including object detection, instance segmentation or person keypoint estimation, although object detection with bounding boxes has been excluded in 2019 challenge. COCO is a famous general dataset formed out of more than 120000 images and 80 different classes. Maximum mAP of last winner was 0.526.

- **Open Images Challenge** [38]: A new annual challenge that started in 2018. It addresses both object detection and action prediction since the beginning and has just added instance segmentation for 2019. Open Images is a massive general dataset that contains almost 2 million images annotated with 500 classes. It is still not as well-known as the previously mentioned ones, but will probably become very important in the following years. Maximum mAP of the 2018 winner was 0.5866.

- **SUNRGB-D 3D Object Detection Challenge** [45]: Another new challenge to be hosted in CVPR, less generalized than the others. It is composed of more than 10000 indoor images of rooms which include both RGB and depth information. For the competition, they evaluate object detection with bounding box but in 3D, although it also has 2D annotations for evaluation. Results of winners are not currently available.

This non-exhaustive list is just a short description of some of the main international challenges, but every year new datasets and challenges appear.

This kind of competitions should not be considered as an exclusive way of evaluating a model performance, because as competitions they could sometimes be close or miss some other important criteria, but it is an objective manner of comparing state of the art methods. In addition, datasets from these challenges are some of the main ones used to evaluate object recognition models in literature. It must be noticed that all of them are formed by conventional images, which a limited field of view and therefore specific research on panoramic images is explained in next section.

As it can be seen with the results, dataset in which a model is evaluated noticeably influences its performance and hence one should be aware of the characteristics of the dataset whenever evaluating numeric results.

2.3 Panoramic images

As the whole project deals with panoramic images, and they are not a conventional type of images, the spherical model and equirectangular projection is going to be introduced. Panoramic images have a 360º horizontal field of view, which means that they include the whole scene and therefore are usually projected on a sphere.

Equirectangular projection is a type of projection for mapping the surface of a sphere to a flat image, such as the typical world map projection we are all very used to, as shown in Figure 2.12.

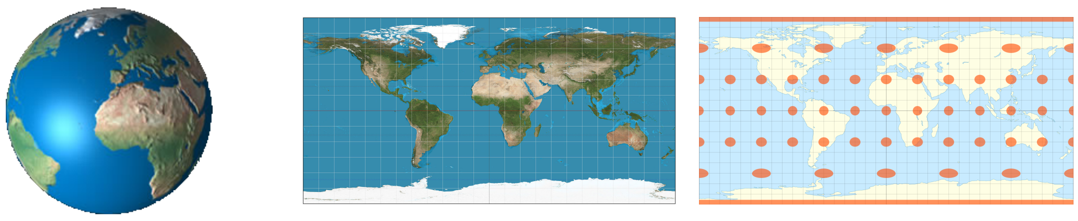


Figure 2.12: Equirectangular projection on world map. Notice orange marks of Tissot's indicatrix of deformation on right image [9].

In an equirectangular panoramic image all verticals remain vertical, and the horizon becomes a straight line across the middle of the image. Coordinates in the image relate linearly to inclination and azimuth angles in the real world. Poles are located at the top and bottom edges and are then stretched to the entire width of the image, with a distortion that becomes smaller when approaching the vertical center of the image.

Considering panoramic image resolution as W x H pixels, being W the width and H the height and taking into account that they cover 360º horizontally and 180º vertically, it can be deduced that $W = 2H$. Coordinates center is situated in the center of the image, $i.e.(\frac{W}{2}, \frac{H}{2})$.

On the other hand, spherical coordinates define spatial position of a point $P(X, Y, Z)$ with three magnitudes: radius r, inclination or polar angle ϕ and azimuth θ, as it can be seen in Figure 2.13, being the coordinates center situated on the center of the sphere.

Having a 2D pixel, a 3D ray can be obtained, which goes from the center to that pixel position on the sphere surface. The unit sphere is then used in order to have a concrete 3D point. Any 3D point $P(X, Y, Z)$ is projected on the unit sphere $p(x, y, z)$ as follows:

$$(x, y, z) = \frac{1}{\sqrt{X^2 + Y^2 + Z^2}}(X, Y, Z)$$

Changing to spherical coordinates:

$$(\sin\phi \cdot \cos\theta, \cos\theta \cdot \cos\phi, \sin\theta) = (x, y, z)$$

And changing to image coordinates (u, v):

$$(u, v) = \left(\phi\frac{W}{2\pi} + \frac{W}{2}, \theta\frac{H}{\pi} + \frac{H}{2}\right)$$

Therefore, given a point on the equirectangular image with coordinates (u, v) its angles on the sphere are calculated as:

$$(\phi, \theta) = \left(\frac{2\pi(u - \frac{W}{2})}{W}, \frac{\pi(v - \frac{H}{2})}{H}\right)$$

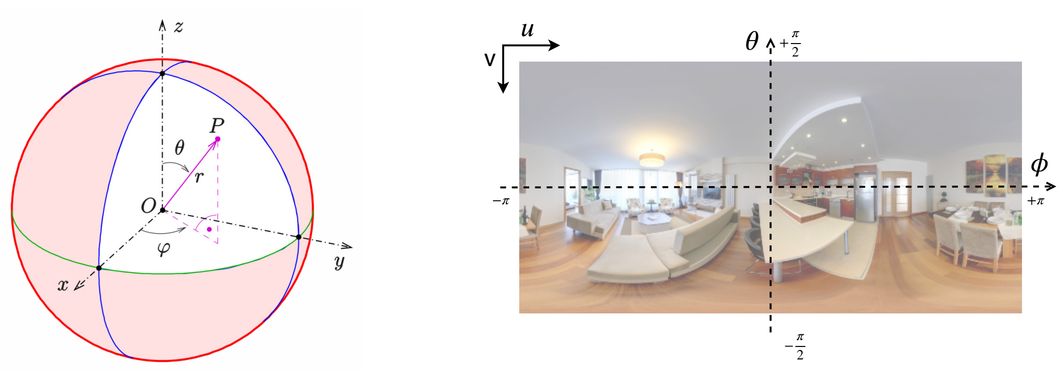
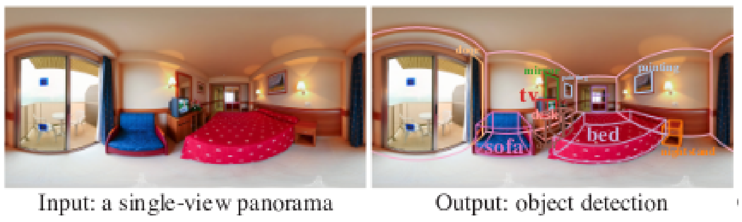


Figure 2.13: Equirectangular projection: Conversion from spherical to image coordinates.

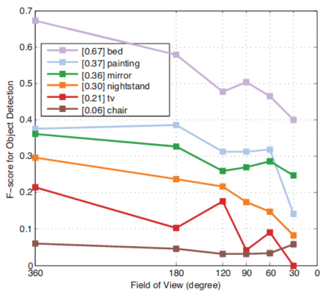
2.3.1 Scene understanding

After analyzing state of the art in object recognition, it has been identified that recent research focuses on using conventional images. This type of images have a limited field of view, which prevents contextual information from being as crucial as it is in scene understanding for humans.

Differently from outdoor object recognition, where thanks to the increasing research on autonomous driving, has some recent works using this kind of images [37] [57], it has been noticed that there is no wide research on object recognition with indoor panoramas. The most relevant work that addresses this problem, and therefore serves as motivation for our work is PanoContext [58]. PanoContext work proposed to use panoramic images to address the complete indoor scene understanding problem. It included 2D object detection and semantic segmentation of the image, but also, and where they mainly focused their contribution, 3D reconstruction of layout of rooms and 3D bounding boxes of main objects. They were actually pioneers in the field, achieving outstanding results and demonstrating how helpful can larger field of view be in both layout estimation and object recognition, taking advantage of the full context [See Figure 2.14b]. Their method did not use deep learning and was only based on geometrical reasoning and traditional computer vision feature extractors. In semantic segmentation, they created initial proposals with selective search, to create 3D detection bounding boxes and finally converted them into semantic segmentation masks, taking into account occlusion testing.



(a) PanoContext results in 3D object detection



(b) F-Score in object detection with decreasing FOV

Figure 2.14: PanoContext contributions to scene understanding problem on panoramas [58].

PanoContext worked with SUN360 bedroom and living room databases and, from the best of our knowledge, it is still considered as state of the art in object recognition on indoor panoramas. Authors of [58] also contributed to annotate it and offered their labels publicly.

Recent works address this indoor 3D layout reconstruction problem using deep learning techniques: in combination with geometrical methods such as [15], as the main part of the pipe-line but with extra pre- or post-processing such in LayoutNet [59] or Dula-net [56], or as an end-to-end fully convolutional network, like CFL (Corners for Layouts) [14] [See Figure 2.15].

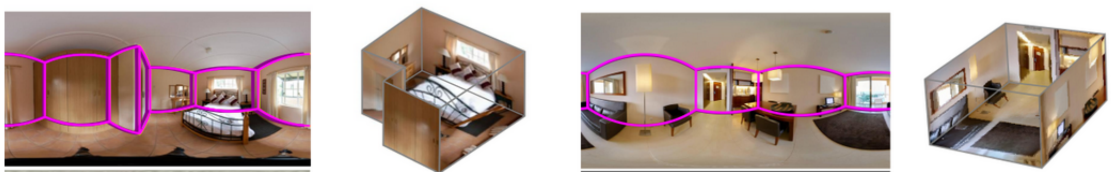


Figure 2.15: Examples of 3D layout recovery task. As seen on the image, they extract edges and corners on the 2D image to conduct 3D reconstruction of the room layout. Figure is taken from results of [14].

2.3.2 EquiConvs

With the use of deep learning and CNNs for panoramic images, a new challenge comes up accounting distortions caused by the equirectangular projection: The traditional translational weight sharing becomes ineffective with those space-varying distortions. Spherical CNNs [8] were a related relevant theoretical contribution, but it is still not clearly demonstrated whether they can reach the same accuracy and efficiency on equirectangular images.

Fernandez et. al. in [14] proposed equirectangular convolutions (EquiConvs). EquiConvs, based on deformable convolutions [11], adapt the receptive field of convolutional kernels by deforming their shape, following distortion of equirectangular projection. As it can be seen in Figure 2.16, EquiConvs are defined in the spherical domain instead of image domain, and hence when approaching the poles, deformation of kernel becomes higher, to be able to follow the translation of a conventional kernel but on the sphere surface. Additionally, padding is not needed and when kernel approaches the border it continues in its correct position on the other side of the panoramic image.

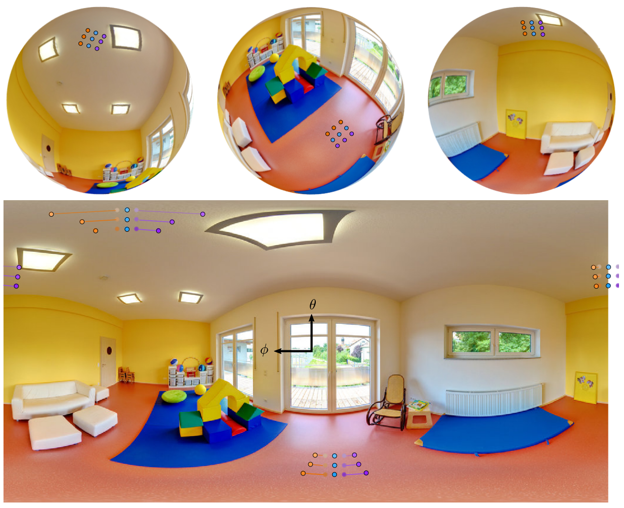


Figure 2.16: EquiConvs kernel on 360° images. Notice the three different kernel positions both projected on the sphere and on the equirectangular image, that highlight differences between offsets depending on the case [14].

In detail, the kernel in these equirectangular convolutions is considered as a surface patch on the sphere and defined by its angular size α and resolution r [See Figure 2.17]. The kernel is rotated along the sphere and its position refers to the spherical coordinates (ϕ, θ) of the center. To create this kernel, they generate fixed offsets based on the equirectangular distortion model, that varies along vertical dimension but remains constant over the horizontal one.

To obtain distorted pixel locations from the original ones, they firstly define the coordinates for every element in the kernel and afterwards rotate them to the point of the sphere where the kernel is being applied. Each point of the kernel is defined as:

$$\hat{p}_{ij} = \begin{bmatrix} \hat{x}_{ij} \\ \hat{y}_{ij} \\ \hat{z}_{ij} \end{bmatrix} = \begin{bmatrix} i \\ j \\ d \end{bmatrix}, \tag{2.1}$$

where i and j are integers in the range $[-\frac{r-1}{2}, \frac{r-1}{2}]$ and d is the distance from the center of the sphere to the kernel grid, which in order to cover the field of view α is $d = \frac{r}{2\tan(\frac{\alpha}{2})}$. They project each point into the sphere surface by normalizing the vectors, and rotate them to align the kernel center to the point where the kernel is applied.

$$p_{ij} = \begin{bmatrix} x_{ij} \\ y_{ij} \\ z_{ij} \end{bmatrix} = R_y(\phi_{0,0}) R_x(\theta_{0,0}) \frac{\hat{p}_{ij}}{|\hat{p}_{ij}|}, \tag{2.2}$$

where $R_a(\beta)$ stands for a rotation matrix of an angle β around the a axis. $\phi_{0,0}$ and $\theta_{0,0}$ are the spherical angles of the center of the kernel and $(u_{0,0}, v_{0,0})$ its pixel location on the equirectangular image. Spherical coordinates are defined as follows:

$$\phi_{0,0} = (u_{0,0} - \frac{W}{2})\frac{2\pi}{W} \quad ; \quad \theta_{0,0} = -(v_{0,0} - \frac{H}{2})\frac{\pi}{H}, \tag{2.3}$$

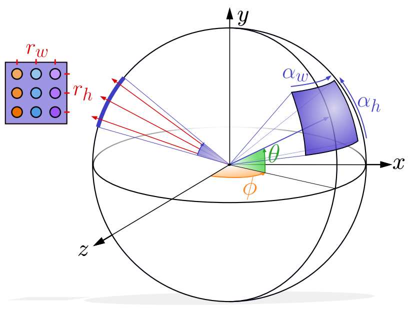
where W and H are, respectively, the width and height of the equirectangular image in pixels.

Finally, the rest of elements are back-projected to the equirectangular image domain, for which they firstly convert all of them to spherical coordinates:

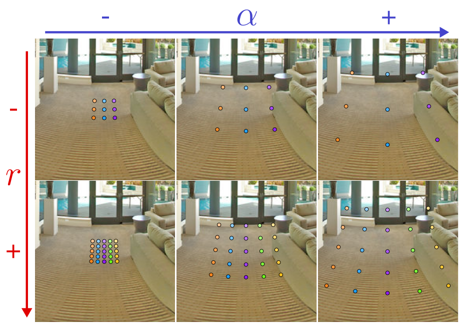
$$\phi_{ij} = \arctan\left(\frac{x_{ij}}{z_{ij}}\right) \quad ; \quad \theta_{ij} = \arcsin\left(y_{ij}\right) \tag{2.4}$$

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

$$u_{ij} = \left(\frac{\phi_{ij}}{2\pi} + \frac{1}{2}\right)W \quad ; \quad v_{ij} = \left(-\frac{\theta_{ij}}{\pi} + \frac{1}{2}\right)H \tag{2.5}$$



(a) Spherical parametrization of EquiConvs. Given square kernels, it can be considered $\alpha_w = \alpha_h = \alpha$ and $r_w = r_h = r$.



(b) Effect of changing field of view $\alpha = (0.2, 0.5, 0.8)$ and resolution $r = (3, 5)$.

Figure 2.17: Equirectangular Convolutions details [14].

In [14] EquiConvs were applied to obtain corners for 3D layout recovery and proved their success. Although performance numeric results were quite similar to the ones with standard convolutions, equirectangular convolutions avoided learning bias on image patterns that produce overfitting to, for example, a concrete camera pose. They also allowed a more direct use of pre-trained networks on conventional images.

Therefore, they are considered a promising method to test in different tasks, such as object recognition with deep learning on panoramic images.

Chapter 3

Dataset

One of the most important parts of a deep learning approach is the data. To train and validate our model, a massive and correctly labeled dataset of panoramic images was needed, so the next part of the project consisted on doing research on existing public datasets and select the best that fitted our needs.

The main objective was to find a public dataset of indoor panoramic images, with object detection bounding boxes labels and semantic or instance segmentation information. For future work, we were also interested in finding a dataset with 3D information, as our object recognition could be used to improve 3D reconstructions of indoor rooms. Panoramic images datasets, however, are not as common and developed as the ones for conventional images so research was not trivial. Some of the considered datasets are listed here:

- **SUN360**: Panoramas 360^o exclusive dataset with indoor and outdoor realistic scenes [27]. Extra annotations 2D-3D provided in PanoContext work, including layouts and object bounding boxes [58].
- **Standford 2D-3D-S**: Panoramas 360^o exclusive dataset with 2D, 2.5D and 3D domains, instance-level semantic and geometric annotations in indoor scenes [4].
- **SUNCG**: 3D exclusive dataset with models of synthetic indoor scenes, including all visible layout elements and objects with pose, semantic information, and texture [48]. New **SUMO** challenge dataset has just been released, derived from processing scenes from the SUNCG dataset to obtain 2D panoramic RGB-D images and 2D semantic information [49].
- **Matterport 3D**: Large-scale RGB-D dataset indoor scenes. Annotations are provided with surface reconstructions, camera poses, and 2D and 3D semantic segmentations [5].

Finally, the SUN360 dataset was selected, more precisely the bedroom and living room subsets, for which annotations were available. We decided to use this data because, although not massive, it was the only one where bounding boxes of objects were accessible, together with 2D and 3D layouts of rooms that, as already mentioned, were specially interesting for future work. Segmentation ground truth was not directly available but could be obtained as it is detailed in section 3.1.1.

3.1 SUN360

SUN360 (Scene UNderstanding 360° panorama) database [27], from Princeton University, contains 360° x 180° visual angle panoramas of resolution 1024 x 512 (they can also be downloaded in higher resolution if needed). As already mentioned, it has both indoor and outdoor images, categorized in 80 sets of different sizes depending on the scene. PanoContext work labelled two of the indoor subsets with bounding boxes, layouts and additional 3D information and made it public. They identified 273 different labels for object categories and with a maximum of 48 objects per image.

For this project we have used bedroom and living room sets, formed by 418 and 248 images respectively, making a total of 666 images, with 14 different object classes considered, as detailed in Table 3.1.

Categories	**Number of objects**
bed	540
table	1,319
mirror	366
window	848
curtain	1,059
chair	1,377
light	749
sofa	855
door	1,473
cabinet	898
bedside	550
tv	463
shelf	123
painting	1,261
TOTAL	**11,881**

Table 3.1: Final 14 categories of the used dataset and the number of objects there are for each category.

3.1.1 PanoContext labels processing

To create this dataset, PanoContext annotations have been used and pre-processed in order to get the final representative categories, as initially there were too many different ones, without enough examples of each of them and with some issues related to their manual annotation process. This pre-processing was conducted using Matlab and it is further detailed in Appendix A.

After that, one important task was the creation of segmentation masks. For this purpose, we used Matlab in order to make use of PanoContext Image Processing Toolbox, which contained some useful functions. Both binary masks for each instance and semantic segmentation masks for the full image were created.

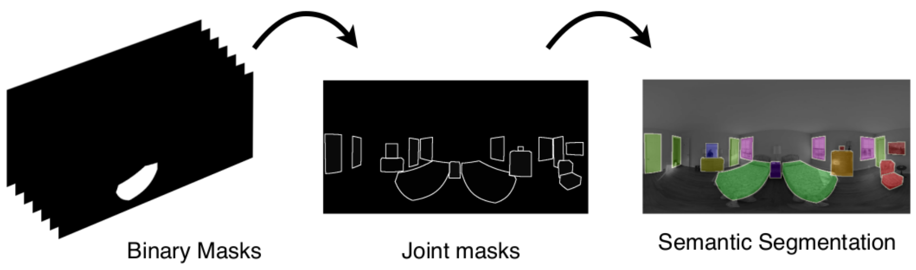


Figure 3.1: Process of segmentation masks creation from binary masks.

Firstly, to create binary masks we used 2D bounding boxes. Bounding boxes were given by coordinates of all object corners, instead of the typical four corners of the enclosing rectangle, which was quite useful for the creation of the segmentation masks. We took all 2D points on the image, projected them onto the unit sphere (this was done by changing from (u, v) image coordinates to (ϕ, θ) and then to (x, y, z), as explained in section 2.3) and drew the contours as the minimum path between points on the sphere surface. This was done in order to get better contours for the objects, following their spherical geometry [See Figure 3.2]. Finally, they were re-projected on the image, obtaining the resulting mask. It is not a perfect segmentation mask, specially for objects with non-cuboid shapes, but was considered reasonably appropriate for the system.

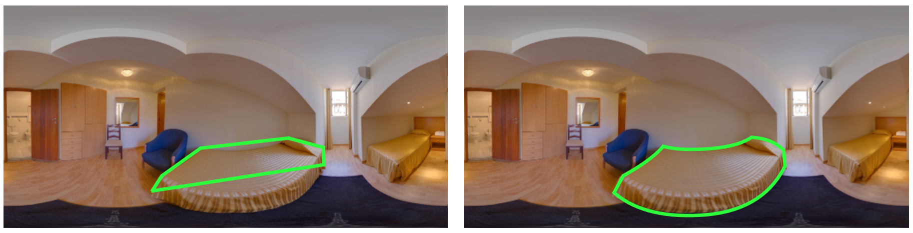


Figure 3.2: Differences between drawing contours directly on image domain (left image) or changing to spherical domain (right image).

Special mention deserve objects that were cropped by the borders of the image, due to the equirectangular projection. They needed to be treated apart, obtaining the bounding points on the borders and making the union of the two polygons to create mask [See Figure 3.3]. With standard convolutions, those objects had to be separated into two different detections, something that was not needed for EquiConvs, which work on the spherical domain.

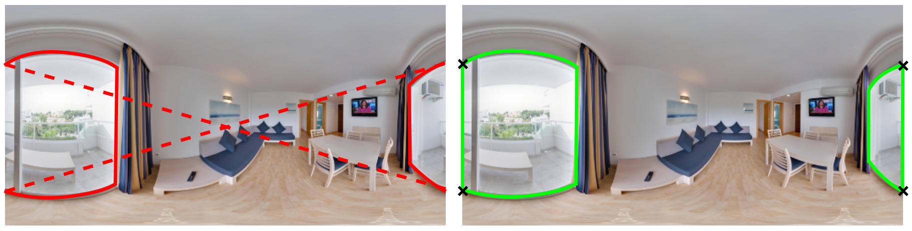


Figure 3.3: **Cropped objects:** If not treating them apart (left image), when reprojecting the contours the polygon is not closed and the mask is wrongly created by automatically closing it with straight lines. When treating them apart (right image), bounding points on the borders of the image have to be calculated on the sphere domain and two different closed polygons are created to make the correct binary mask.

Secondly, semantic segmentation masks of the whole image was also created. Based on binary mask, they could just be projected all together on the same image and get a final result. However, the main challenge was to decide occlusions when objects overlap, which is the case in many of the objects. Without depth or other 3D information, we needed to make the assumption that objects are not in general complete occluded, and therefore the following hypothesis was considered: *For each pair of objects in conflict, calculate area of each object and ratio of overlap. If overlap is bigger than a threshold, consider that the smallest object is closer and therefore completely visible (it happens also when small objects are far but behind a window or door), otherwise consider the biggest object is closer.*

This hypothesis proved to work quite well in many cases, allowing to correctly segment most of the visible objects in the images [See Figure 3.4]. Finally, to adapt ground truth segmentation to train the BlitzNet model, contours were added to the objects as shown in Figure 3.1.

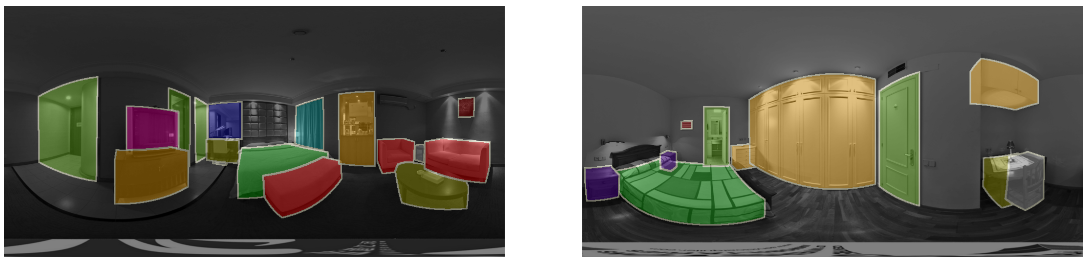


Figure 3.4: Examples of created semantic segmentation masks. Notice the good results given by our hypothesis on left image, and some inaccuracies such as cuboid form for chair or background bedside on top of foreground bed, on right image.

An initial try of creating better segmentation masks was to make use of some of the models with available code, and with high segmentation accuracy. Based on the state of the art research, it was decided to execute Mask R-CNN (already trained on COCO) [1] with our data, in order to get an accurate instance segmentation at least for the classes that they had in common.

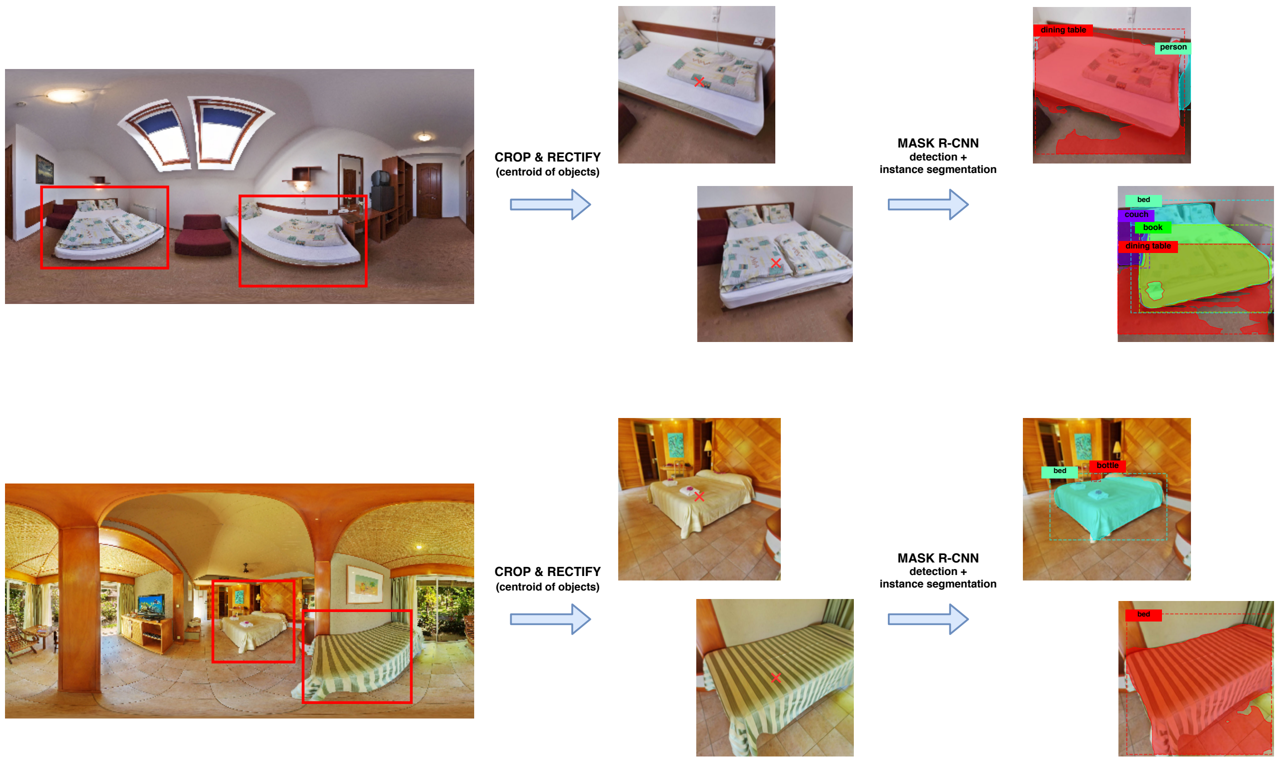


Figure 3.5: Examples of failed segmentations with Mask R-CNN. Images were first cropped and rectified centered on the centroid of each object and given as input for Mask R-CNN Network. Notice wrong segmentation results.

As it did not work well directly on panoramas (due to distortions that had not been learnt), panoramic images were cropped in patches and rectified centering the image in each of the objects to create the instance segmentation masks. However, results were not successful at all, as it can be seen in Figure 3.5. May be because of the concrete characteristics of our data, but we finally decided to use our own-generated masks. This experiments show that the problem is not yet solved and motivate the development of the concrete model for panoramas.

After the dataset processing, all images and masks were saved together with a *JSON* file, having all information easily accessible for using it.

3.2 Main drawbacks and challenges

Dataset used has some important drawbacks that must be considered as they will influence significant decisions during the design of the model and evaluation results.

Firstly, bounding box annotations were manually created by five people in PanoContext project. Although they have an acceptable level of accuracy, we detected some imprecisions or errors due to this manual process, as it is detailed in Appendix A. Our pre-processing and classes clustering resolved many of these issues but some of them still remained.

Secondly, segmentation masks were own-generated. On the one hand, because of being created based just on bounding boxes, they are inaccurate on non-cuboid shapes, which makes it more difficult for segmentation performance in those objects. On the other hand, the lack of occlusion information or other types of 3D labels when creating segmentation masks leaded to the use of assumptions and hypothesis, which can fail in certain cases, giving a non-exact ground truth that can affect both training and testing of the system.

This ground truth inaccuracies cannot be solved by the design of the model but should be considered when analysing results of the system. Solutions include to create new annotation labels or improve existing ones, ideally by a non-manual process which assures better accuracy, and to obtain depth information for a better occlusion testing when creating segmentation masks.

Apart from that, the non-massive dataset ($\sim$ 500 images are not enough to successfully train a very deep learning model from scratch) supposed a clear challenge. It has been managed by conducting pre-training with other massive datasets on conventional images (ImageNet, SUN RGB-D), which is possible with better accuracy thanks to the use of equirectangular convolutions, and by the use of specific data augmentation. Due to the characteristics of the dataset, which images are quite similar in terms of camera pose and distribution of objects, overfitting to this kind of images will be a major issue difficult to evaluate without different datasets. Other issues such as the slight class imbalance will also be mentioned and considered in the model.

3.3 SUN RGB-D

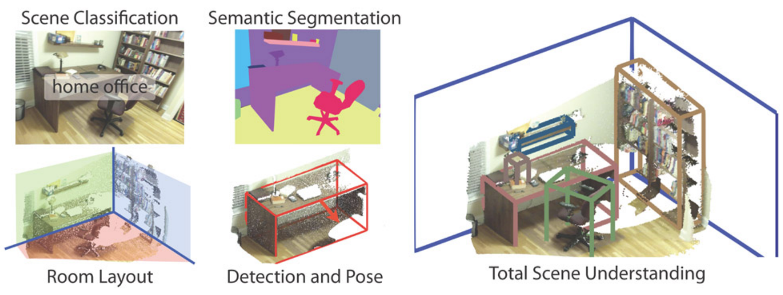


Figure 3.6: SUN RGB-D dataset [45]

To do pre-training, a dataset of conventional images, with similar indoor classes, bounding boxes and segmentation masks, and preferably with layout and 3D information, was needed. For this purpose, SUN RGB-D [45] dataset, from Princeton University, was selected and pre-processed to filter the same 14 categories of SUN360. It is a very complete benchmark of indoor scenes with more than 10000 RGB-D images and all annotations for different scene understanding tasks, including object detection and semantic segmentation.

Chapter 4

Model

In this Chapter, the development of the proposed CNN model for object recognition on panoramic images is explained. Main originality of this work is the creation of a model that can work directly on panoramic images, considering their distinctive characteristics. The necessity of this type of models was initially proved by trying models already thought and trained for conventional images and feeding them with panoramas, both directly the complete panorama or the rectification of different crops from the original image, which had a clear lower performance than in other types of images [See Section 5.2]. Apart from all adjustments made to the original base network to be able to work directly on panoramas, special mention deserves the use of EquiConvs [14] and the analysis of its impact on the results of the system.

Our model conducts simultaneously two object recognition tasks: object detection and semantic segmentation. After that, an approach to instance segmentation task, implemented as post-processing of the system, is also described in Section 4.3. Finally, in the last Section 4.4, a proof of concept to add layout recovery to the system and combine it to work on a whole scene understanding pipeline is introduced and explained.

4.1 CNN base architecture

The core model used in this work is based on BlitzNet [12] network, whose code is publicly available on GitHub, implemented in Python and Tensorflow. It is a joint Fully Convolutional object detection and semantic segmentation pipeline, that performs detection in one-stage [See Section 2.2.1].

As shown in Figure 4.1 the input image is first processed by a CNN feature extractor, that creates a general high-level feature map. In this case, this extractor is ResNet50 [See Section 2.2.1] because of its high performance for classification and good trade-off in speed.

After that, and following the SSD [35] approach, feature maps are iteratively down-scaled to perform multi-scale detection. Following, and specially thinking about predictions of specific localization in pixel-wise segmentation, feature maps are up-scaled via deconvolutional layers, following the typical encoder-decoder model in many FC networks. This approach is quite similar to DSSD [17], but with a new proposed deconvolution module called 'ResSkip', which includes residual and skip connections.

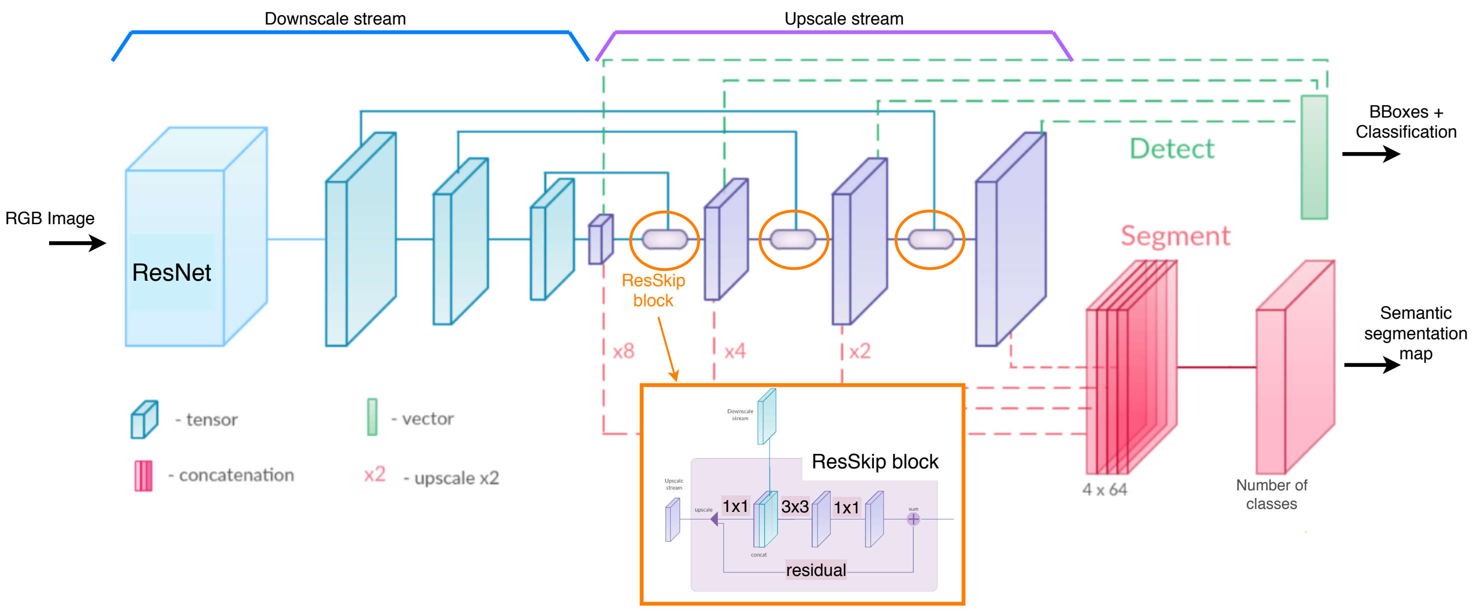


Figure 4.1: Our model architecture that performs detection and segmentation with one FCN. ResNet feature extractor is followed by downscale and upscale streams to finish with detection and segmentation heads. ResSkip blocks are highlighted to see their architecture in depth. Main parts of this figure are taken from [12].

Each of these ResSkip blocks takes both maps, from the downscale and from the upscale stream (as shown in Figure 4.1) and combines them with a simple strategy: both incoming feature maps are upsampled to the size of the corresponding connection via bilinear interpolation. The resulting maps are concatenated and passed through a block (1x1 conv + 3x3 conv + 1x1 conv) and summed together with the upsampled input through a typical residual connection. In [12] they proved the benefits for this type of blocks, which may be related to the use of similar residual connections to the ones used in the Backbone ResNet50, making the overall architecture more homogeneous. Finally, predictions are conducted by both detection and segmentation heads, two single convolutional layers in one forward pass. Object detection head takes all bounding boxes in the feature maps of the multi-scale stream and predicts class and coordinate corrections for each bounding box. Similarly, segmentation head up-scales all activations of the multi-scale stream, concatenates them and feed it to the final classification layer that predicts pixel labels, creating the semantic segmentation map.

As evidenced by the architecture, almost all weights of the pipeline are shared for both tasks. This strategy is based on the assumption that semantic segmentation and object detection share several key properties. They both require a per-region classification, based on pixels inside the region but also taking into account context on its surroundings. Besides the computational time gain of having a unique network to perform the two tasks simultaneously, this weight sharing allows both tasks to benefit from each other, as it is discussed in Section 5.3.4.

In [12], they developed two slightly different architectures: BlitzNet300 and BlitzNet512, to use square inputs of resolution 300 and 512, respectively. BlitzNet512 added an extra layer in the upscaling stream and changed from stride 4 to 8 in the last layer. We used BlitzNet300 for initial experiments, using square and low resolution panoramas. Finally, we selected BlitzNet512 architecture, adapting it for non-square panoramic images. All other parameters used in experiments are detailed in Chapter 5.

Loss functions

We use a multitask loss which is defined as the sum of both segmentation and detection loss. As values of both losses are within the same range, they both contribute to the total loss, allowing the network to learn both tasks. Weighting losses differently when computing the sum did not have noticeable improvements in terms of accuracy.

$$L = L_s + L_d \tag{4.1}$$

Segmentation loss L_s is simply defined as the cross-entropy between predicted and target class of every pixel in the segmentation mask. Output of the network in the segmentation head is given as M feature maps, being M the number of classes. Values are in range $[0, 1]$ in both ground truth and predicted mask and the segmentation loss is calculated as:

$$L_s = -\sum_{c=1}^{M} y_{o,c} log(p_{o,c}), \tag{4.2}$$

with M the number of classes, y a binary indicator (0 or 1) if class label c is the correct classification for observation o and p the predicted probability that observation o is of class c.

Detection loss L_d is a bit more complex, taking into account a weighted sum between both localization and confidence loss.

$$L_d = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, a, g)) \tag{4.3}$$

where L_{conf} and L_{loc} are confidence and localization loss respectively (which are following explained), N is the number of matched predicted bounding boxes (if $N = 0$, wet set the total loss to 0), α is the weight adjusted by cross-validation, $x_{ij}^p = \{1, 0\}$ is an indicator for matching the i-th predicted box to the j-th ground truth box of category p, c are classes confidences and l, a and g are predicted and ground truth bounding box parameters (x, y, w, h), respectively.

First of all, we should match anchor bounding boxes with ground truth bounding boxes in order to calculate loss. Remember that, as a one-stage method, prediction of the class and regression of bounding box (offsets with respect to anchor box) are calculated simultaneously. For matching, Intersection over Union [see Section 5.1] for each pair of bounding boxes is used. Matching is done twice: firstly each ground truth bounding box is matched to the anchor box with the best IoU overlap and secondly matching is done with pairs that have IoU bigger than a threshold, like they do in [35]. This simplifies the learning problem, assuring that all ground truth bounding boxes are considered and allowing the network to predict high scores for multiple overlapping default bounding boxes instead of require to match only with the one with maximum overlap. Notice that, with this matching strategy, we can have $\sum_i x_{ij}^p \geq 1$.

Confidence loss L_{conf} is the softmax loss applied over multiple classes confidences c.

$$L_{conf}(x, c) = -\sum_{i=1}^{N} x_{ij}^p log(\hat{c}_i^p) - \sum_{i \in Neg} log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)} \tag{4.4}$$

Localization loss L_{loc} is a Smooth L1 loss between predicted, $\hat{o}$, and ground truth o offsets. This offsets are calculated using predicted, anchor and ground truth parameters, concretely using the center position (cx, cy), width w and h height. The regression is based on the strategy introduced by Faster R-CNN [43] and is defined as follows:

$$\begin{aligned}
\hat{o}_j^{cx} &= (l_j^{cx} - a_i^{cx})/a_i^w & \hat{o}_j^{cy} &= (l_j^{cy} - a_i^{cy})/a_i^h \\
\hat{o}_j^{w} &= \log\left(\frac{l_j^w}{a_i^w}\right) & \hat{o}_j^{h} &= \log\left(\frac{l_j^h}{a_i^h}\right) \\
o_j^{cx} &= (g_j^{cx} - a_i^{cx})/a_i^w & o_j^{cy} &= (g_j^{cy} - a_i^{cy})/a_i^h \\
o_j^{w} &= \log\left(\frac{g_j^w}{a_i^w}\right) & o_j^{h} &= \log\left(\frac{g_j^h}{a_i^h}\right) \\
L_{loc}(x, l, a, g) &= \sum_{i=1}^{N} \sum_{m \in \{cx, cy, w, h\}} x_{ij}^{p} \mathrm{smooth}_{\mathrm{L1}}(\hat{o}_i^m - o_j^m)
\end{aligned} \tag{4.5}$$

4.1.1 Adapting CNN: Panoramic BlitzNet

This base model needed to be adapted to work with panoramas, to create our proposed model: Panoramic BlitzNet. Firstly, a program was implemented to feed the network with our concrete dataset, for both training and testing, adapting the format of images and annotations.

After that, the objective was to modify the network to work correctly on rectangular images instead of square ones. This was important for the use of panoramas for our main goal of using the complete context for object recognition, but also for practical importance when using EquiConvs, as they work on spherical domain considering the image as projected on the complete sphere surface.

It initially seemed to be a simple adjustment because it is a Fully Convolutional model, which eliminates dependency between number of weights and dimensions of images. However, it was not a trivial task in the end. The initial tiling of the image for bounding box proposals had to be modified, as approach used was centered on pixels and dimensions of the initial image. It also should match dimensions with output of convolutions, whose reduction ratio had changed when using rectangular images. Our proposal bounding boxes are done by firstly converting image to a regular grid (of aspect ratio 1/2 to cover the whole rectangular image). Grid has different dimensions in each layer, from 128x256 to 1x2, related to the iteratively lower scale of the feature maps. In each grid cell 5 different proposals are created with 5 different aspect ratios: 1, 2, 1/2, 3 and 1/3 [shown in Figure 4.2], allowing objects to have different shapes.

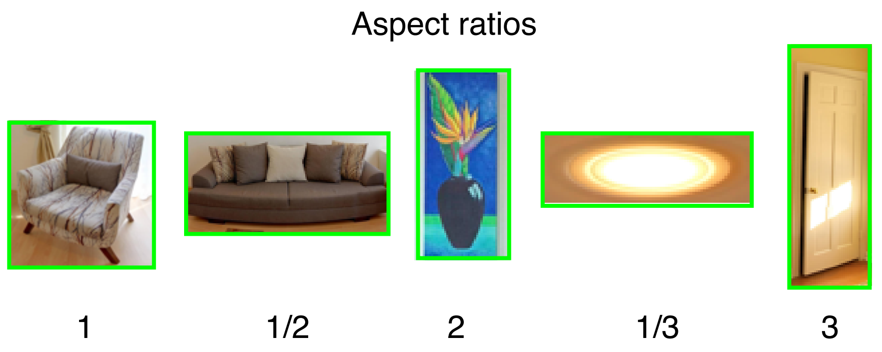


Figure 4.2: Aspect ratios used in proposals for bounding boxes seen on top of real cropped objects from our dataset.

Many other aspects had to be changed in the code and required much time of debugging, because it was implemented very *'ad hoc'* for square images, having a lot of different points where they were reshaping tensors to a square size. This aspects are not in depth explained as they are considered implementation details and are already documented on the source code.

4.1.2 Data augmentation

Data augmentation is a major characteristic of our system. As we had to deal with a non-massive dataset, conducting data augmentation has been very important to improve generalization and avoid overfitting. Apart from typical data augmentation which was already implemented in the base model, we have modified it, adding new techniques and deleting others, to adjust to the particularities of panoramic images.

In data augmentation, we first conduct some photometric distortions, affecting brightness, saturation or contrast to help the network learning under different light conditions. After that, we also apply horizontal mirror flips, to variate object positions. Random crops that were implemented in the network for conventional images, were eliminated as it does not allow to use the complete context, which is our intention, and also prevents from using EquiConvs. Additionally, SUN360 dataset images are all very similar in terms of camera pose, so this kind of data augmentation would only make it more difficult to learn, being detrimental for its performance.

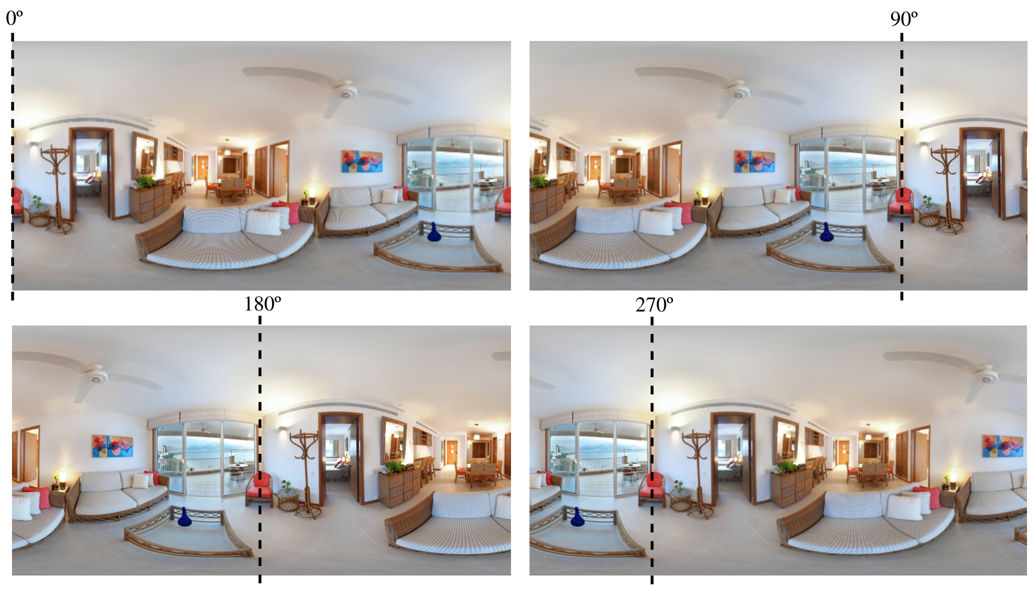


Figure 4.3: Horizontally rotation of the same equirectangular image along the sphere, used for data augmentation. Notice limits of the image and how different objects are cropped or not depending on the rotation.

Following CFL [14] ideas, we apply an important data augmentation that consists on rotating the panorama horizontally from 0° to 360° to cover all different positions in the sphere, as shown in Figure 4.3. Random erasing, is a different data augmentation technique to simulate occlusion by erasing a random patch on the image and is used in CFL work, as they needed to obtain corners that were not directly seen on the scene. However, it is not applied in this case as, although we should deal with many occlusions between objects and detect them even if they are partially visible, we are not interested in detecting their occluded part. It may be an interesting technique to apply when working on 3D reconstruction but not in our 2D semantic segmentation.

4.1.3 Pre-training and fine-tuning

Based on our dataset characteristics, we decided to use pre-trained models instead of randomly initializing the CNN. As datasets of panoramas are not massive and standard, we decided to use conventional images to pre-train the system and then fine-tune the network with our images. Benefits of this fine-tuning are commented on Chapter 5, together with the analysis of results. Firstly, we used a pre-trained model of ResNet50 backbone weights on ImageNet dataset, that was publicly available. With those weights, we then trained the whole network on SUN RGB-D dataset that had been already processed to have our same categories [See Section 3.3]. This pre-training was possible because, although it was conducted with the original base network adjusted for conventional images, having a Fully Convolutional model allows us to share weights with the model adapted for panoramas.

4.2 EquiConvs

Once we had the network built to work with panoramic images, EquiConvs [See Section 2.3.2] were integrated in our system, to study their impact on object recognition. To adapt it we had to modify all bottlenecks that conduct convolutions with kernel size > 1 (equirectangular convolutions are indifferent to $1x1$ convolutions), in both the encoder and the decoder. For the decoder, EquiConvs code does not have upconvolutions implemented, but it was not a problem as our base model used unpooling (with bilinear interpolation) and then normal convolutions.

With EquiConvs, our fine-tuning makes much more sense because in conventional images distortion is not present and weights are learnt based on that. Therefore, EquiConvs implicitly handle equirectangular distortion to be able to take advantage of previous weights as if they were learnt on the same kind of images. Aditionally, taking into account that our dataset does not include camera pose changes, there is a risk that standard convolutions overfit to the dataset and its distortion patterns, which does not happen when using EquiConvs that are invariant to those changes, as proved in [14].

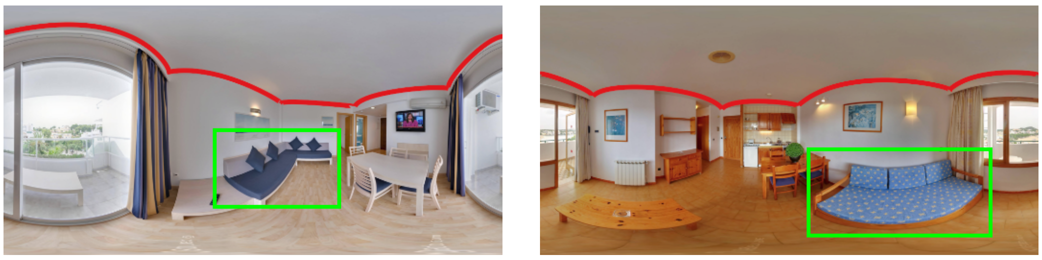


Figure 4.4: Differences between distortion patterns in layout, which are more similar among images (in red), and on objects such as a sofa here, which vary in many different ways (in green).

Apart from that, EquiConvs benefit the task of object recognition with a higher significance than the layout recovery task in CFL, as shown in results [See Chapter 5]. As seen in Figure 4.4, it can probably happen because corners and edges used in CFL, although being also distorted, follow a more uniform distortion pattern among different images than object shapes. This allows standard convolutions to rely on that image patterns, that together with a dataset where camera pose is uniform, makes it easier for their performance on equirectangular images. Anyway, EquiConvs prove to be a successful improvement to solve both object detection and semantic segmentation tasks on panoramic images.

4.3 Instance segmentation

The CNN in this work has been used to obtain semantic segmentation of images, what means, images are segmented pixel-wise classifying them into object classes. In order to get an initial approach for instance segmentation task, a simple post-processing has been implemented that takes into account detections together with semantic segmentation to build the result.

The key of the method is a bit naive: it takes each bounding box and treats them as gaussian distributions. As the predicted bounding box should be an enclosing rectangle, it is assumed that 99% of the object is contained on it, so standard deviation in each dimension is taken as $\sigma = \frac{dimension}{6}$, giving 3σ at each side of the mean, which is taken as the center of the bounding box. Finally, each pixel of the semantic segmentation is analysed with Mahalanobis distance to each gaussian distribution and then classified to the minimum distance distribution instance. When none of the distances to distributions exceeds the chi-squared test threshold, the pixel is left as the classification in the initial semantic segmentation. This proposal has shown the best results on experimental tests.

This assumption gives, consequently, more confidence to bigger objects, which are usually better segmented by the network. In addition to allow instance segmentation, it also improves semantic segmentation in many cases, as it gives more confidence to detections, being detection performance clearly better than segmentation [See Chapter 5]. A few examples of successful results of the method, and how it also improves initial segmentation are shown in Figure 4.5.

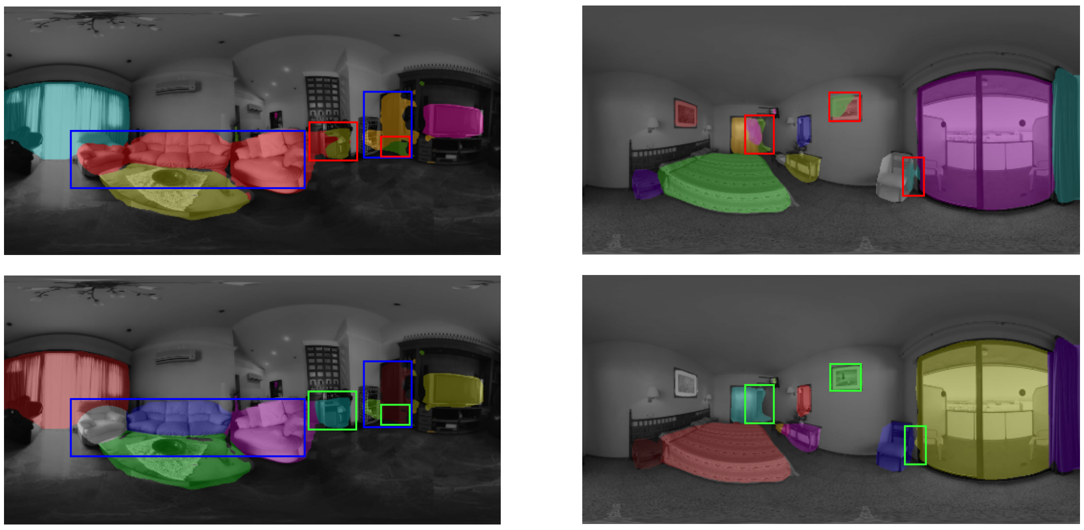


Figure 4.5: Successful instance segmentation results. Top is initial semantic segmentation result (output of CNN) and bottom is instance segmentation result (after post-processing). Notice that apart from correctly differentiate among instances (highlighted in blue) it improves original segmentation thanks to good detections (highlighted in red and green for bad and good segmentation respectively).

As weak points of the approach, it can be noticed that it makes the result segmentation worse when detections fail, and also when detecting the shape of small objects that are enclosed by a bigger bounding box, which is just predicted as a small gaussian shape [See Figure 4.6].

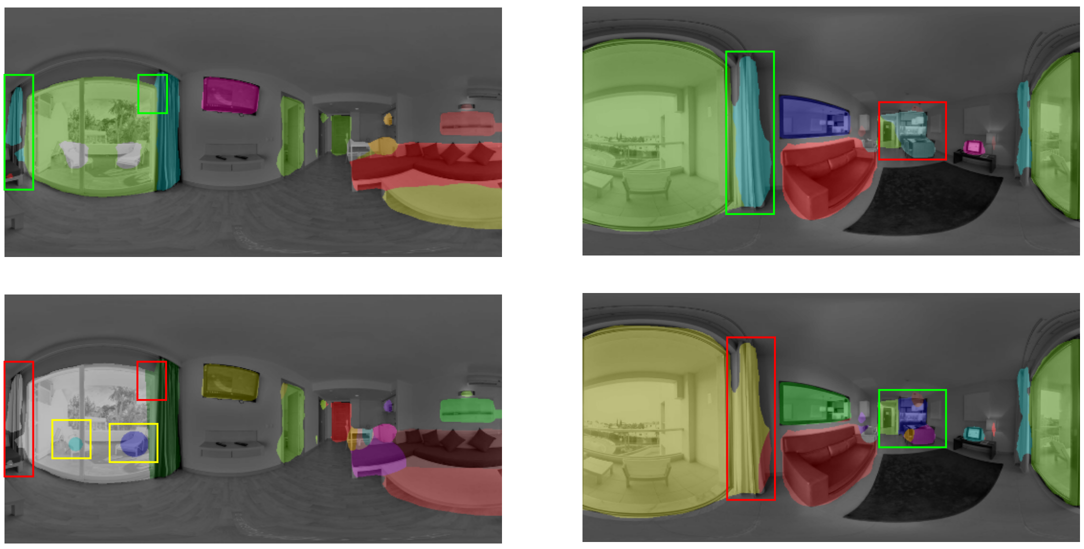


Figure 4.6: Failed examples of instance segmentation method. Top is initial semantic segmentation result (output of CNN) and bottom is instance segmentation result (after post-processing). Notice gaussian shape in small objects of left image (highlighted in yellow) and how it sometimes modify semantic segmentation incorrectly due to bad detections (highlighted in red and green for bad and good segmentation respectively).

The method has been implemented and integrated in the system, as an option to test program, which, if included, predicts both semantic segmentation and instance segmentation and saves the results.

To conclude, it is a simple first approach to the task that cannot be treated as a real contribution but it does help in understanding the difficulties of instance segmentation, and slightly improves results of the system. As clear future work, modifying the outputs of the network to adapt the loss for instance segmentation could be a good solution, having better ground truth instance segmentation data. Another option, again with the condition of having this information in the ground truth, could be to make the CNN predict some kind of data which could be post-processed to obtain a trusted instance segmentation. Other authors use that approach in [7] and [53], using a predicted vector for each pixel directed to the center of its instance bounding box.

4.4 Layouts

At the end of the project, and thinking of future work of combining object recognition with 3D layout recovery, a proof of concept of this idea has been developed. It was considered that, following the idea of simultaneous learning, another scene understanding task, in this case edge detection for layout recovery, could be added to the system and possibly benefit each other with object recognition. We based that on the intuition of continue using context, such as some specific object positions on indoor scenes. For example, paintings are always located on walls, furniture is always on the floor and usually on walls, or lights are most of the times on the ceiling.

To develop this idea, edge maps of SUN360 bedroom and living room sets were downloaded and added to our dataset [Figure 4.7]. Network was modified to implement this idea and some experiments were conducted, which are explained in this section, although consistent conclusions cannot be extracted up to now.

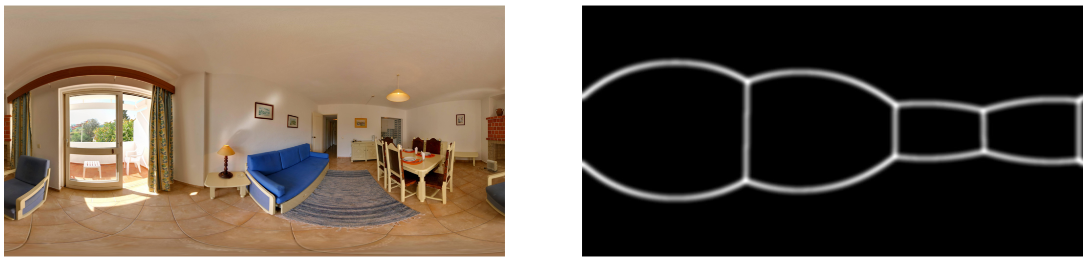


Figure 4.7: Example of an edge map and the corresponding RGB image. This kind of edge maps can be considered the 2D layout of the image and are used to get the 3D layout reconstruction of the room.

There were two different approaches considered:

- **Input layout:** Using layout as a new input of the network. This was an initial try to observe if layout and object recognition tasks can be considered as related problems, improving our predictions done from only the RGB image as single input. The implemented network was adapted to add this input, but in order to use initial ImageNet weights, a new channel was not added and edge map was introduced together with the three RGB channels. Results on SUN RGB-D dataset did not show any benefits, so this approach was not further studied. May be that edge maps should be introduced as a different input or that layouts on conventional images are not as descriptive as layouts on panoramas with a full view of the scene, so no further resolutions can be confirmed.
- **Output layout:** Adding layout prediction to the pipeline, as a new output of the network. To do so, a new head, which we called 'layout head', was firstly added to the architecture. It follows the idea of segmentation head, concatenating multi-scale layers and feeding them to a single convolutional layer. This layer performs pixel-wise segmentation in form of edge maps, with a single channel as it is a grey-scale image. Loss function is defined, exactly as in CFL work [14], as the binary cross entropy function between the ground truth and the predicted edge maps, considering a threshold to conduct that binarization and adding it to the global loss. Although these changes have already been implemented, experiments have not been conducted yet so the influence of joint learning of object recognition and layout recovery still remains as future work.

Chapter 5

Evaluation

Evaluation of the system was conducted by different experiments, which are detailed in this Chapter. For all the experiments, our dataset [See Chapter 3] was divided into two sets: 85% to train (566 images) and 15% to test (100 images). We did not specifically separate a set for validation as many important parameters of the architecture were taken directly from BlitzNet work [12]. When needed, a random subset of the train set was used for the validation of specific parameters or the training adjustment.

5.1 Evaluation metrics

Common evaluation metrics in object recognition literature and how they are used in this project to evaluate performance of the system are explained in this section, in order to better understand further results.

5.1.1 Semantic segmentation

Intersection Over Union (IoU) is the most typical metric used for segmentation and detection evaluation. It is measured based on Jaccard index, that evaluates the overlap between two areas. It requires a ground truth area A and a predicted area $\hat{A}$ and is given by their overlapping area divided by the area of their union.

$$IoU = \frac{A \cap \hat{A}}{A \cup \hat{A}} \tag{5.1}$$

In semantic segmentation, the most typical metric used in all public challenges and literature is known as **mean IoU**, which is computed as the average of the IoU along all classes, as follows:

$$meanIoU = \frac{1}{M} \sum_{i=0}^{M} \frac{A_i \cap \hat{A}_i}{A_i \cup \hat{A}_i}, \tag{5.2}$$

being M the number of classes (background of the image is considered as class 0) and considering A_i the area formed by all pixels classified with class i in the ground truth segmentation map, and $\hat{A}_i$ the ones classified with class i in the predicted segmentation map.

From information on a confusion matrix, *meanIoU* can be calculated as the diagonal vector (correct classified pixels of each class form the intersection) divided by, the vector of the sum along vertical dimension (ground truth set) plus the vector of the sum along horizontal dimension (predicted set) minus the diagonal vector (intersection), which is the definition of union following $A \cup \hat{A} = A + \hat{A} - A \cap \hat{A}$.

5.1.2 Object detection

For object detection, IoU is also used with areas of ground truth and predicted bounding boxes. Once IoU is computed for each pair of matching bounding boxes (following the corresponding matching strategy), we define the following concepts:

- **True positive (TP)**: A correct detection, *i.e.* a detection with $IoU \geq threshold$.
- **False positive (FP)**: A wrong detection, *i.e.* a detection with $IoU < threshold$.
- **False negative (FN)**: A ground truth that has not been detected, *i.e.* a ground truth with all $IoU = 0$.
- **True negative (TN)**: Does not apply in object detection. It would represent a corrected misdetection, but in an image we could consider infinite bounding box forms that should not be detected, so it makes no sense to consider them.

The detection threshold can be set to different levels, being typically bigger than 30% and lower than 95%. In PASCAL VOC challenges it started being fixed at 50% but changed in 2012 where they considered the average between performance at incremental thresholds. In this project we consider it as 30%, as, having precise semantic segmentation, we did not focus on precise localization in the detection task.

Precision is the ability of a model to identify only the relevant objects. It is then defined as the percentage of positive predictions among all predictions.

Recall is the ability of a model to find all relevant objects. It is then defined as the percentage of positive detected among all ground truth objects.

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN} \tag{5.3}$$

Both precision and recall are important for an object detection system. Precision is usually considered more significant for evaluating the performance of this kind of systems, although this significance is always dependent on the particular application.

Average Precision (AP) is a good metric to evaluate the performance of object detectors, as it combines both precision and recall. It is defined as the average of precision values calculated at all different possible recall levels. As precision-recall curves are usually a zigzag going up and down, interpolated precision (p_{interp}) is used to compute AP, which is considered as the highest precision found for any recall level bigger than the current one.

$$p_{interp}(r) = \max_{r' \geq r} p(r') \tag{5.4}$$

That average can be considered as a simple average at all possible recall levels (N) or as a weighted average taking into account the recall area that they represent, which better approximates the Precision-Recall curve. Both approaches are widely used in literature, depending on the importance they want to give to recall results and are both called AP. In this project, both metrics are used, so they will be referred as AP and AP^w (AP weighted) respectively.

$$AP = \sum_{i=1}^{N} p_{interp}(r_i) \qquad AP^w = \sum_{i=1}^{N} (r_i - r_{i-1}) p_{interp}(r_i) \tag{5.5}$$

considering $r_0 = 0$.

Finally, **Mean Average Precision (mAP)** is the most common metric to evaluate performance of object detectors and is computed as the mean of average precisions along classes. This metric is usually defined by a regular mean in all public challenges, such as PASCAL VOC or COCO, but in this project we define it as a weighted mean, weighting classes in correspondence with the number of detections in which that average precision is based. This is done to take into account that our dataset has a slight class imbalance, which although did not show to be a major issue, can affect performance.

$$mAP = \frac{1}{M} \sum_{i=1}^{M} \frac{d_i}{n} AP_i \tag{5.6}$$

being M the number of classes, d_i the number of detections of class i and n the total number of detections. AP_i could be replaced by AP_i^w in this formula and we will refer to it as mAP^w.

5.2 Preliminary experiments

First of all, two preliminary experiments were conducted on already trained models on conventional images, to prove the necessity of an specific system for panoramas.

When processing the dataset, we firstly conducted a failed experiment with Mask R-CNN, by feeding it with rectified crops of the panoramic image, which is further explained in Section 3.1.1. Apart from the disappointing results, we conclude that having a model that directly works on panoramas could be better and more efficient than having to crop and rectify from a single panorama to multiple conventional images with smaller field of view, and then merge the results.

Secondly, we used YOLO v3 [42] model, whose code is publicly available and implemented in C and CUDA [40] (they developed a whole neural network framework named Darknet). A Python existing interface was adapted to be able to execute it, with weights already trained on COCO dataset, and to use just the classes that both datasets had in common. We tried feeding the network directly with all our panoramic images, without any pre-processing. Results, which are shown in Table 5.1, are better than initially expected (specially qualitative ones), but still not enough satisfactory. Precision results are acceptable and it clearly stands out over recall, which is very low. This can be caused by some thresholds adjusted by the authors, or due to the fact that some objects in images are not almost affected by distortion (so with a good detector as YOLO are usually predicted correctly), but the ones that are distorted are often not detected, as it can be seen in Figure 5.1. Confusion matrix and further results can be found in Appendix B.

		Ground truth		
		✓	✗	
Prediction	✓	TP: 1013	FP: 670	**1683**
	✗	FN: 3677	–	–
		4690	–	

YOLO	Results
precision	0.602
recall	0.216
mAP^w	0.472
mAP	0.679

Table 5.1: Quantitative YOLO experiment results. Notice low number of detections (*recall*) but acceptable level of *precision* and mAP.

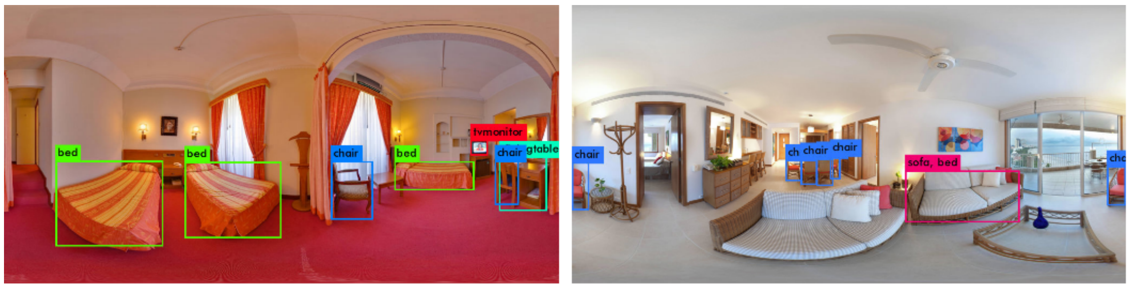


Figure 5.1: Qualitative YOLO experiment results. See very good qualitative results on left image and worse performance on right image, where more distorted objects (sofa, table...) are not detected. Notice how detections are correct in general (high precision) but many objects are not recognized (low recall).

5.3 Experiments

Evaluation of our model is done by five different experiments: First experiment proves the importance of initialization, comparing three different ways of initializing the weights. Second experiment compares performance of our Panoramic BlitzNet with original BlitzNet that uses square images. Third experiment shows the variation of performance given different values for the confidence threshold. Fourth experiment studies the importance of jointly training the two tasks: object detection and semantic segmentation. And finally, fifth experiment compares performance of Panoramic BlitzNet using both standard and equirectangular convolutions. As Panoramic BlitzNet with EquiConvs is our best method proposed, its results are considered the final evaluation of the system.

All our experiments have been conducted on the same dataset and with the same optimization strategy. Following BlitzNet [12] configuration, Adam [30] stochastic optimization algorithm is used, but this time with varying mini-batch sizes, which will be stated in each experiment. Learning rate is set to 10^{-4} and it is decreased twice during training. Some experiments were conducted by changing that learning rate without noticeable influence. All models have been trained until convergence, measured with a random validation subset.

5.3.1 Initialization

First experiment was conducted training with our dataset of panoramic images but using BlitzNet300 architecture, without modifying the base implementation, and therefore random cropping and resizing panoramas to a square shape 300x300. Batch size was set to 16 and it was trained until convergence. When training, different initializations of weights were executed to compare: random initialization (trained all from scratch), ImageNet initialization of ResNet50 and pre-trained SUN RGB-D initialization. ImageNet initialization converged faster, but it demonstrated higher overfitting than SUN RGB-D pre-training. It must be considered that training from scratch such a deep model with just $\sim$ 500 images will for sure not work well, but experiments were conducted to prove the influence of initialization. Results are compared in Table 5.2, which shows that pre-training with SUN RGB-D followed by fine-tuning the complete network with panoramic dataset, gives the best test results and generalizes better. However, they are still not very satisfying results, probably due to the fact that images are too much distorted, with random crops and resizes, to make them square.

	TRAIN mAP^w	TEST mAP^w	TRAIN $meanIoU$	TEST $meanIoU$	Convergence
From scratch	0.911	0.468	0.872	0.419	45k iter.
ImageNet (ResNet)	0.896	0.479	0.788	0.432	23k iter.
SUN RGB-D	0.728	**0.516**	0.742	**0.461**	95k iter.

Table 5.2: **Effect of initialization:** Results with original BlitzNet model trained on SUN360 train-seg and tested with SUN360 test-seg dataset. Notice how initializing with SUN RGB-D weights gives clear better test results, although convergence is bigger (it includes pre-training). Train results are also shown to observe the overfitting effect, which happens specially in trained from scratch model.

5.3.2 Square vs. Panoramic BlitzNet

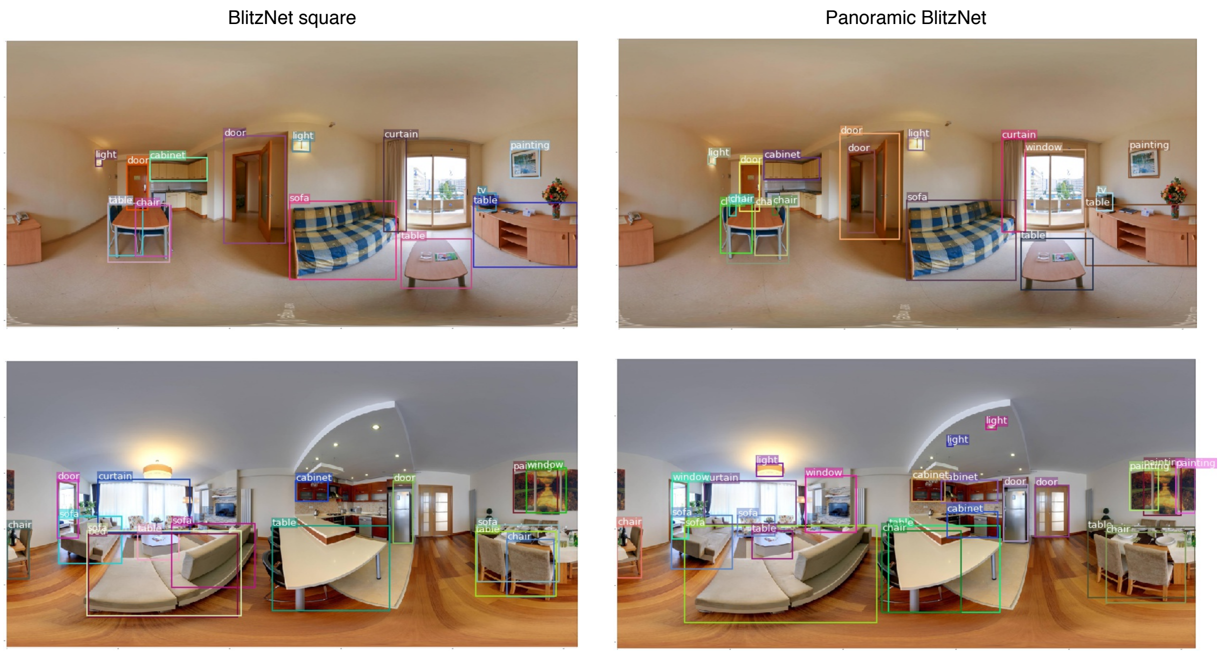


Figure 5.2: **Qualitative evaluation of object detection:** BlitzNet initial model (squared images), and Panoramic BlitzNet are compared in these two example images. Second image is complex as having many different objects and bigger distortions. Notice how detection task works well in both models but initial BlitzNet does forget some of the objects and is also less precise in localization of bounding boxes.

After adapting the CNN to use panoramas, experiments were carried out to evaluate the system and compare our Panoramic BlitzNet with the original model. Batch size was reduced to 4, for memory limitations in our GPU. As seen in Table 5.3, our adapted model demonstrates clear better results, improving performance by a wide margin and supporting our assumption of the important benefits of a concrete model for panoramas. Qualitative results also support this improvement, like detections in Figure 5.2 or specially when having a look at segmentation predicted maps, as shown in Figure 5.3.

	$meanIoU$	mAP^w	mAP	Convergence
BlitzNet (square)	0.461	0.516	0.688	95k iter.
Panoramic BlitzNet	**0.530**	**0.632**	**0.768**	102k iter.

Table 5.3: **Effect of adapting for panoramas:** Comparison between base initial and adapted model. Both models are pre-trained on SUN RGB-D. Notice it is the most significant improvement, as using the complete panorama is the key of our system.

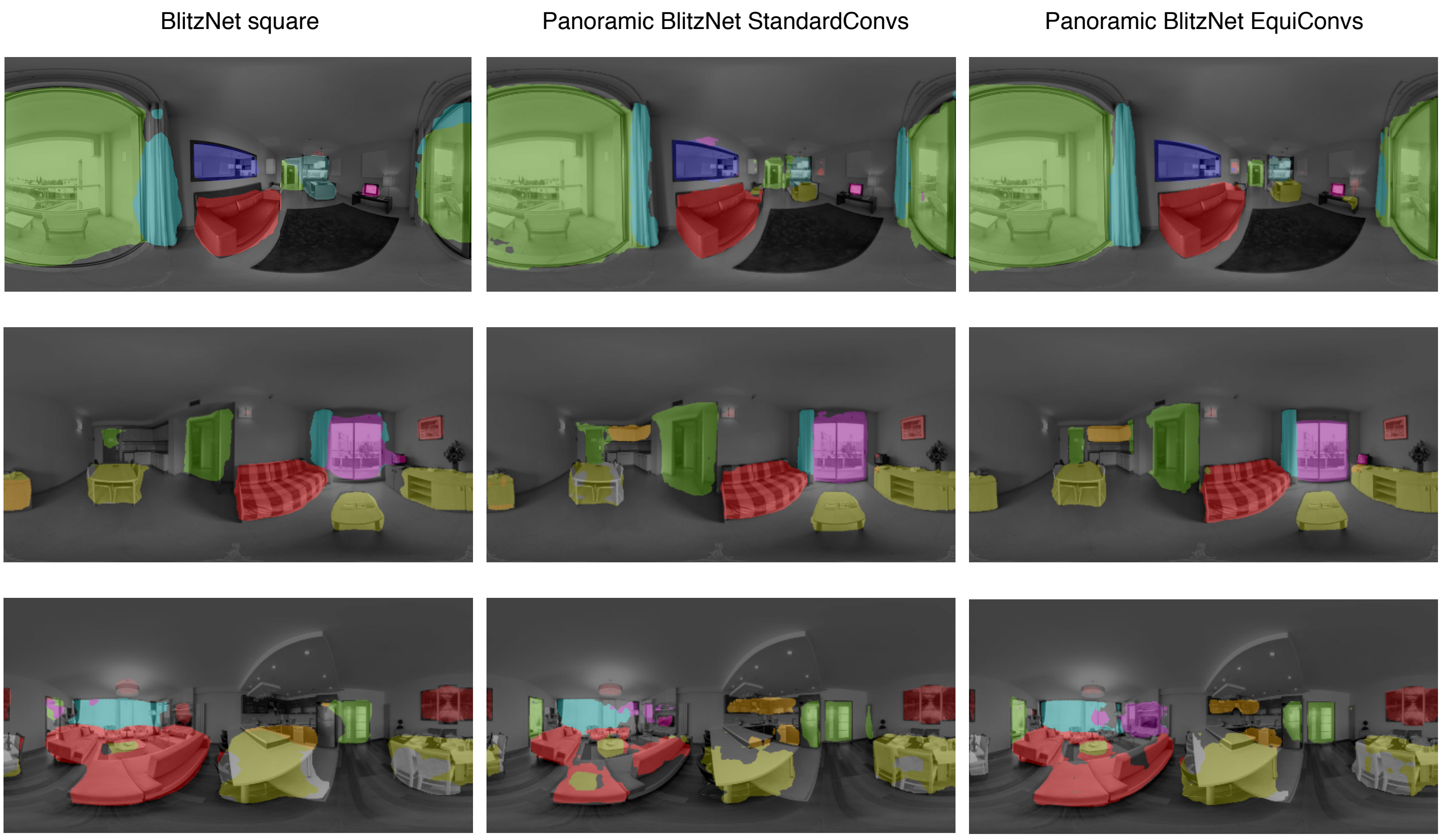

BlitzNet square
Panoramic BlitzNet StandardConvs
Panoramic BlitzNet EquiConvs

Figure 5.3: **Qualitative evaluation of semantic segmentation:** BlitzNet initial model (square images), and our Panoramic BlitzNet with both standard convolutions and EquiConvs are compared in these three example images. Third image is complex as having many different objects and bigger distortions. Notice how Panoramic BlitzNet performs better in most of the cases, specially when using EquiConvs. On the third image, bigger and non-precise segmentation from original BlitzNet performs acceptably well, better than Panoramic with StandardConvs that fails trying to be more precise, but worse than Panoramic with EquiConvs that adapts better to distortion.

5.3.3 Confidence threshold

When testing, a confidence threshold must always be set to decide which of the predicted bounding boxes are reliable enough to be proposed as detections. Changing this threshold is clearly a trade-off between precision and recall, as with a low threshold predictions are less reliable so many objects are detected but the number of false positive increases. Meanwhile with a higher one, detections are mostly correct but many objects are not detected so the number of false negatives increases. Evolution of these metrics depending on the threshold can be observed in Figure 5.4. If not stated differently, this threshold is set as 0.5 by default.

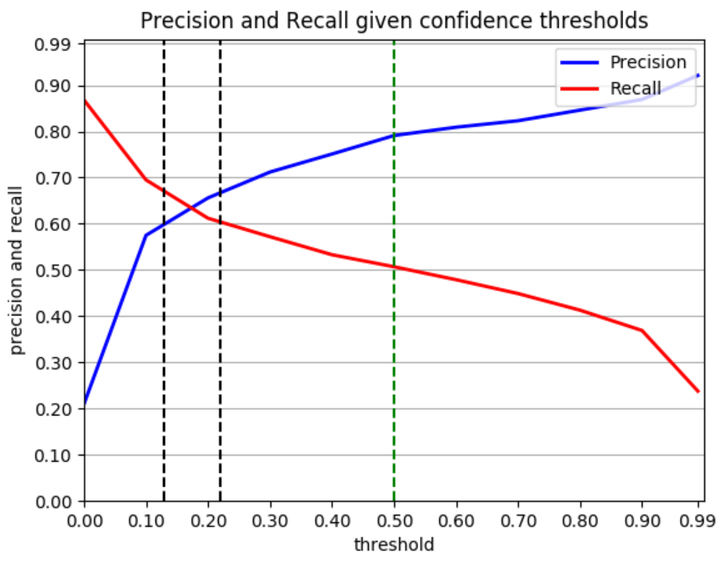


Figure 5.4: Evolution of global precision and recall with different confidence thresholds. For this evaluation our model, Panoramic BlitzNet with standard convolutions, initialized on SUN RGB-D, has been used. Notice optimal threshold (considering both precision and recall levels) would be in ~ 0.18, but we consider as default 0.5, giving more importance to precise detections, even if there is a small detrimental for recall, which is still maintained above 0.5.

5.3.4 Joint training

The effect of joint training is also an important aspect to analyze. In Table 5.4 results of our Panoramic BlitzNet, trained on ImageNet initialization are presented. Although results are lower with this initialization, SUN RGB-D is not used as it was pre-trained with both tasks together so it could deteriorate significance of this results. Differences are not very remarkable, but it can be seen that both tasks benefit from each other, as already stated by [12].

SUN 360	segment.	detect.	*mIoU*	mAP^w	*mAP*
Panoramic BlitzNet	✓	✗	0.476	–	–
Panoramic BlitzNet	✗	✓	–	0.525	0.719
Panoramic BlitzNet	✓	✓	**0.481**	**0.544**	**0.721**

Table 5.4: **Effect of joint learning:** All models are the same, trained on SUN360 train-seg with ImageNet initialization and tested on SUN360 test-seg, but trained only for segmentation, detection or both tasks together, respectively.

5.3.5 StandardConvs vs. EquiConvs

Finally, our best results have been reached by the use of EquiConvs. They took longer to converge but, thanks also to pre-training and data augmentation techniques, clearly help in avoiding overfitting, one of our main challenges. A curious result found with EquiConvs is related to confidence threshold and illustrated in Figure 5.5: EquiConvs make detections much more confident than standard convolutions, as recall is maintained higher than 0.5 with a threshold of 0.75 and higher than 0.4 with a threshold of 0.95, which did not happen with StandardConvs.

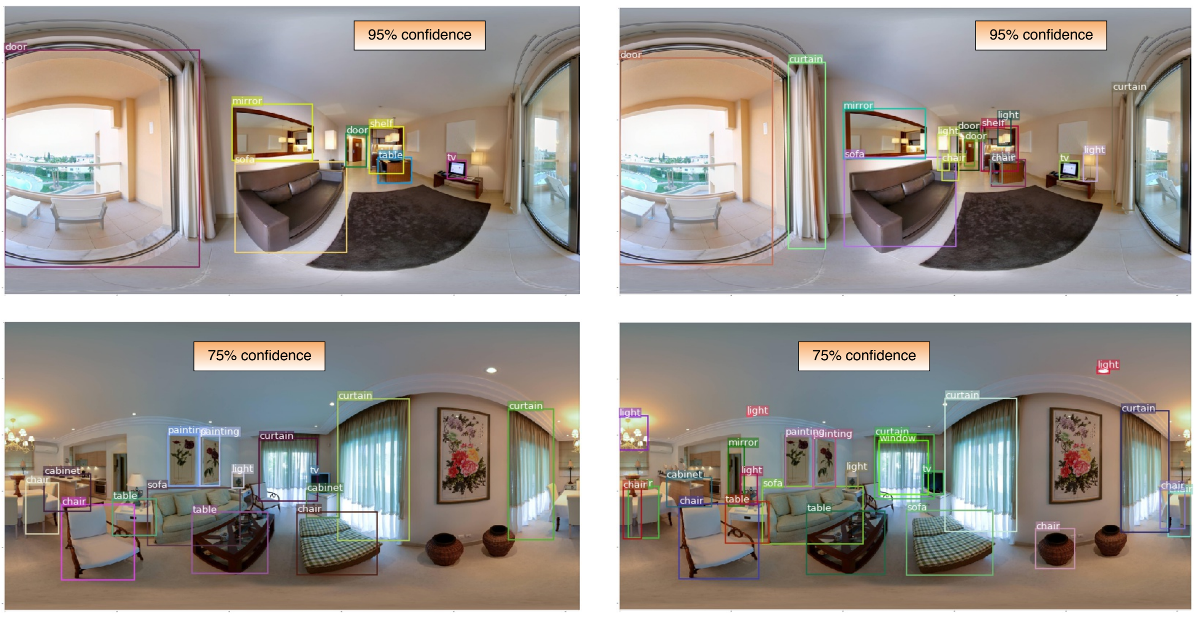


Figure 5.5: Qualitative results of object detection with both standard (left) and equirectangular (right) convolutions, with high confidence thresholds. Notice how EquiConvs detect many more objects (most of them correct) than StandardConvs given the same confidence threshold. Also EquiConvs are slightly more precise on localization bounding box, as seen on the main chair and table of bottom image.

Complete results per category, comparing both kind of convolutions, are shown in Tables 5.5 and 5.6. They show that EquiConvs perform slightly better in both object detection and semantic segmentation, although there are some differences among categories. Confidence threshold is set to 0.2 in these experiments to avoid giving advantage to EquiConvs, as (as already shown) they have clear better recall for high confidence thresholds.

It can be concluded that our Panoramic BlitzNet system with equirectangular convolutions, by managing inherent panorama's distortions and characteristics, achieve satisfactory results, with a final detection **mAP** of **0.778** and segmentation **meanIoU** of **0.544**, that outperforms state of the art.

DETECTION *mAP*	Panoramic BlitzNet StandardConvs (102k iter.)	Panoramic BlitzNet EquiConvs (111k iter.)
bed	0.949	**0.953**
painting	**0.850**	0.839
table	**0.833**	0.821
mirror	0.719	**0.762**
window	**0.722**	0.709
curtain	0.722	**0.759**
chair	0.719	**0.809**
light	0.350	**0.410**
sofa	**0.893**	0.854
door	**0.755**	0.725
cabinet	**0.579**	0.556
bedside	0.879	**0.914**
tv	0.911	**0.933**
shelf	0.305	**0.402**
TOTAL *mAP*	0.768	**0.778**

Table 5.5: **Effect of EquiConvs:** Detailed results, separated in categories, which show differences in object detection task (mAP) between using standard or equirectangular convolutions. Both are trained on SUN360 train-seg with SUN RGB-D initialization and tested on SUN360 test-seg with a confidence threshold of 0.2.

SEGMENTATION *meanIoU*	Panoramic BlitzNet StandardConvs (102k iter.)	Panoramic BlitzNet EquiConvs (111k iter.)
background	0.907	**0.913**
bed	0.617	**0.621**
painting	0.321	**0.612**
table	**0.752**	0.723
mirror	**0.423**	0.414
window	**0.558**	0.534
curtain	**0.540**	0.537
chair	0.551	**0.552**
light	**0.314**	0.265
sofa	**0.346**	0.329
door	0.636	**0.638**
cabinet	0.485	**0.511**
bedside	**0.407**	0.366
tv	0.522	**0.523**
shelf	0.574	**0.619**
TOTAL *meanIoU*	0.530	**0.544**

Table 5.6: **Effect of EquiConvs:** Detailed results, separated in categories, which show differences in semantic segmentation task ($meanIoU$) between using standard or equirectangular convolutions. Both are trained on SUN360 train-seg with SUN RGB-D initialization and tested on SUN360 test-seg.

Further qualitative and quantitative results can be found in the Appendix B.

5.4 Discussion of results

Results of the system arise many different discussions. Given a non-massive dataset, the usefulness of pre-training has definitely been confirmed. Training such a very deep model from scratch with a small dataset only leads to huge overfitting to training data and difficulties to converge. Therefore, the use of a massive dataset of conventional images to learn high level characteristics of the objects for both tasks was one of the keys for the success of the system. Clearly proved is also the importance of using the complete panorama at a higher resolution. Apart from avoiding distortions and crops to make it fit to a square shape, it also shows the benefits of using a wider field of view, which allows to consider the whole context of the room. This idea is a strong support for the potential of panoramic images, not only in object recognition but in many other visual tasks, at least in those related to indoor scene understanding problem.

Conducting two simultaneous tasks is also an interesting characteristic of our model. Experiments show that they benefit from each other, showing better performance results when training together than separately, although not with a clear evidence. When having a look at results, both quantitative and qualitative, detection demonstrates better performance than segmentation. Could be because pixel-wise segmentation is a more precise and complex task that takes more time to converge and would probably require a bigger dataset, and also because of our own-generated ground truth. Some differences between classes are remarkable, such as high detection accuracy in beds, without being among the most representative class in the dataset and having many distorted bed examples. The unexpected not significantly high accuracy in windows and doors is also particular, as they are not only highly represented but also have a very uniform and particular shape. However, it is true that they can be confused in many cases, such as balcony glass doors, and follow widely variate distortion patterns among images. Finally, low performance of shelves in detection could be due to a class imbalance problem as it is the most under-represented class in our data (it can be managed adjusting a special loss), but it surprisingly performs quite well in the segmentation task. It should finally be noticed that, as pre-training was conducted with SUN RGB-D dataset, its characteristics and the distribution of the classes could have also influenced these results in a significant way.

Finally, the use of equirectangular convolutions for object recognition has been one of the main originalities and contributions of the project. They have shown better accuracy results, with a higher significance than in other tasks [14] although not as significant to extract strong conclusions. However, this might be due to the particularities of our dataset, where overfitting does not dramatically damage test results. EquiConvs make pre-training more important and help in avoiding overfitting to training data, apart from having proved better quantitative and qualitative results.

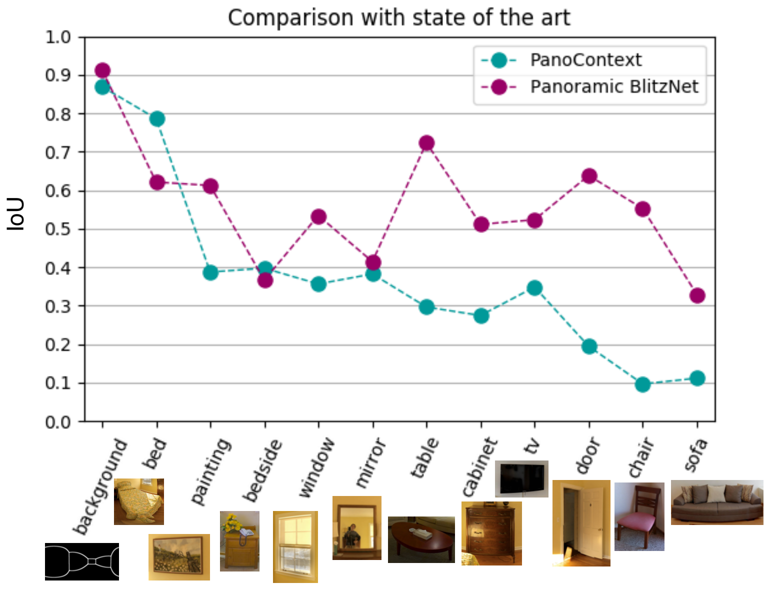


Figure 5.6: Comparison of semantic segmentation performance with state of the art. Results show that our final Panoramic BlitzNet model with EquiConvs totally outperforms state of the art, except from a pair of categories.
Mean IoU results among all common categories are **0.561** $>$ **0.375** [See Table B.3 on Appendix].

Our final model, with its semantic segmentation results, definitely outperforms state of the art in panoramic object recognition, which was obtained by PanoContext [58] in their 2D semantic segmentation experiments. Comparison to them is shown in Figure 5.6, where it can be noticed that our system outperforms them by a wide range. It must be still considered that PanoContext did not make use of deep learning techniques and semantic segmentation was not their main contribution.

In conclusion, results of the system are very satisfactory, proving the necessity of continue researching in specific object recognition models for panoramic images.

5.4.1 Efficiency

Related to efficiency, our Panoramic BlitzNet has a test execution time (on our GeForce GTX GPU) of less than 0.3 seconds per image. When adding EquiConvs, the efficiency decays to ~ 0.9 seconds per image. Instance segmentation post-processing is slower, as a pixel-wise task that was not implemented using Tensorflow optimized functions, but with Numpy library. It raises execution time up to more than 1 second per image, although it could be further optimized. Training times were not computed and are very related to number of iterations until convergence. As a conclusion, despite not being a main focus or objective of this project, the system can perform detection and semantic segmentation of panoramic images quite fast and close to real time.

Chapter 6

Conclusions

Developing a deep learning system for object recognition on panoramic images has been a challenging and interesting work. As we did not have previous knowledge about this field, a deep research on literature and analysis of state of the art has been conducted as the first step of the project, gaining the needed level of understanding.

A complete model for object detection and semantic segmentation has been developed, making use of BlitzNet work [12] but adapting it to work on panoramic images. This kind of images are very interesting because their wide field of view allows making use of contextual information, which is an important factor to successfully recognize objects in scenes. However, the extra complexity added by their particular distortions lead us to develop an specific deep learning model for indoor object recognition on panoramas that, from the best of our knowledge, had not been done before.

The model has been tested with SUN360 dataset, which was processed to create segmentation masks that were not initially available. Experiments conducted clearly state the importance of using the complete panorama, giving much higher performance results than in the original model. Besides, the advantages of simultaneous joint learning and the usefulness of pre-training deep learning models when dataset is not massive, as in our case, have been totally supported by experimental results.

Another main contribution, together with the use of the complete 360° image, has been the study of equirectangular convolutions impact on the task. This kind of convolutions, following the distortion patterns on the image, have proved satisfactory results in both detection and segmentation tasks. Additionally, they specially help in avoiding overfitting, the reason why they are promising to be tested on different panoramic datasets. As the last contribution, the developed proof of concept to integrate this system with the layout recovery task, allowing both problems to benefit from each other, is considered as an interesting idea to further research on the complete scene understanding problem.

Finally, due to the originality of the work and the satisfactory results that represent an improvement for other state of the art methods, there is an intention of writing and publishing a scientific paper about it.

6.1 Future work

Possible future work includes the development of better ground truth labels or the use of more massive and different datasets to continue evaluating the generalization capacity of the system.

Another idea would be to adapt the Panoramic BlitzNet model to support instance segmentation. Continuing with the approach already developed, the system could be modified to add a pipeline for instance segmentation task learning, which could be evaluated thanks to our already available binary masks.

Finally, and as the most promising future work, we should continue studying the influence of layouts on the system. Conduct experiments and evaluate the proposed model would arise the impact on jointly learning another scene understanding task. The objective would be to allow both object recognition and layout recovery to benefit from each other and possibly integrate them in a more complex system by adding objects to obtain a complete 3D reconstruction of indoor scenes.

Bibliography

[1] W. Abdulla. Mask r-cnn for object detection and instance segmentation on keras and tensorflow. `https://github.com/matterport/Mask_RCNN`, 2017.

[2] A. Andreopoulos and J. Tsotsos. 50 years of object recognition: Directions forward. *Computer Vision and Image Understanding*, 117:827–891, 08 2013.

[3] A.R. Jimenez, R. Ceres and J.J. Pons. A survey of computer vision methods for locating fruits on trees. *IEEE Transactions of the ASABE 43*, page 1911–1920, 2000.

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

[5] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.

[6] S. Chatterjee. Traditional versus deep learning algorithms in retail industry. `https://towardsdatascience.com/traditional-vs-deep-learning-algorithms-in-retail-industry-i-b7b7f86793d4`. Last accessed: 2019-06-10.

[7] L. Chen, A. Hermans, G. Papandreou, F. Schroff, P. Wang, and H. Adam. Masklab: Instance segmentation by refining object detection with semantic and direction features. *CoRR*, abs/1712.04837, 2017.

[8] T. S. Cohen, M. Geiger, J. Köhler, and M. Welling. Spherical cnns. *CoRR*, abs/1801.10130, 2018.

[9] W. contributors. Equirectangular projection — Wikipedia, the free encyclopedia, 2019. Last accessed: 2019-06-10.

[10] Convolutional Neural Networks for Visual Recognition course (CS231n). Standford University. `http://cs231n.stanford.edu/`. Last accessed: 2019-06-06.

[11] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. *CoRR*, abs/1703.06211, 2017.

[12] N. Dvornik, K. Shmelkov, J. Mairal, and C. Schmid. Blitznet: A real-time deep network for scene understanding. *CoRR*, abs/1708.02813, 2017.

[13] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, Jan. 2015.

[14] C. Fernandez-Labrador, J. M. Fácil, A. Perez-Yus, C. Demonceaux, J. Civera, and J. J. Guerrero. Corners for layout: End-to-end layout recovery from 360 images. *arXiv:1903.08094*, 2019.

[15] C. Fernandez-Labrador, A. Pérez-Yus, G. López-Nicolás, and J. J. Guerrero. Layouts from panoramic images with geometry and deep learning. *CoRR*, abs/1806.08294, 2018.

[16] C. Fernández-Labrador. Estimación del layout 3d en interiores a partir de imágenes panorámicas. Master's thesis, Escuela de Ingeniería y Arquitectura. Universidad de Zaragoza, 2017.

[17] C. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. DSSD : Deconvolutional single shot detector. *CoRR*, abs/1701.06659, 2017.

[18] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. G. Rodríguez. A review on deep learning techniques applied to semantic segmentation. *CoRR*, abs/1704.06857, 2017.

[19] R. B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[20] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[21] G. R. Giuseppe Gallus. A decisional model of recognition applied to the chromosome boundaries. *Journal of Histochemistry & Cytochemistry*, 22(7):546–553, 1974. PMID: 4850193.

[22] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[23] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *CoRR*, abs/1406.4729, 2014.

[24] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[25] L. Hoseong. A paper list of object detection using deep learning. `https://github.com/hoya012/deep_learning_object_detection`, 2019.

[26] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[27] A. O. J. Xiao, K. A. Ehinger and A. Torralba. Recognizing scene viewpoint using panoramic place representation. *Proceedings of 25th IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[28] R. K. Aggarwal and K.-S. Fu. A pattern classification system for the identification of irradiated chromosomes. *Biomedical Engineering, IEEE Transactions on*, BME-24:178 – 185, 04 1977.

[29] A. Kathuria. What's new on YOLO v3? `https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b`. Last accessed: 2019-06-10.

[30] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2014. Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

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

[33] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017.

[34] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[35] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.

[36] E. N. Malamas, E. G. Petrakis, M. Zervakis, L. Petit, and J.-D. Legat. A survey on industrial vision systems, applications and tools. *Image and Vision Computing*, 21(2):171 – 188, 2003.

[37] X. Meng, X. Zhang, K. Yan, and H. Zhang. Real-time detection and recognition of live panoramic traffic signs based on deep learning. 2018.

[38] Open Images Challenge. Google AI. `https://storage.googleapis.com/openimages/web/challenge2019.html`. Last accessed: 2019-06-20.

[39] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[40] J. Redmon and A. Farhadi. YOLO (You Only Look Once) Real Time Object Detection. `https://pjreddie.com/darknet/yolo/`.

[41] J. Redmon and A. Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016.

[42] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[43] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015.

[44] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[45] S. L. S. Song and J. Xiao. SUN RGB-D: A RGB-D Scene Understanding Benchmark Suite. *Proceedings of 28th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[46] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2014.

[47] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 09 2014.

[48] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic scene completion from a single depth image. *Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[49] SUMO Challenge dataset. `https://sumochallenge.org/`. Last accessed: 2019-05-10.

[50] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[51] C. Szegedy, S. E. Reed, D. Erhan, and D. Anguelov. Scalable, high-quality object detection. *CoRR*, abs/1412.1441, 2014.

[52] L. Uhr and C. Vossler. A pattern recognition program that generates, evaluates, and adjusts its own operators. In *Papers Presented at the May 9-11, 1961, Western Joint IRE-AIEE-ACM Computer Conference*, IRE-AIEE-ACM '61 (Western), pages 555–569, New York, NY, USA, 1961. ACM.

[53] J. Uhrig, E. Rehder, B. Fröhlich, U. Franke, and T. Brox. Box2pix: Single-shot instance segmentation by assigning pixels to object boxes. In *IEEE Intelligent Vehicles Symposium (IV)*, 2018.

[54] J. Uijlings, K. E. A. Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104:154–171, 09 2013.

[55] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. *CoRR*, abs/1611.05431, 2016.

[56] S. Yang, F. Wang, C. Peng, P. Wonka, M. Sun, and H. Chu. Dula-net: A dual-projection network for estimating room layouts from a single RGB panorama. *CoRR*, abs/1811.11977, 2018.

[57] W. Yang, Y. Qian, F. Cricri, L. Fan, and J. Kamarainen. Object detection in equirectangular panorama. *CoRR*, abs/1805.08009, 2018.

[58] Y. Zhang, S. Song, P. Tan, and J. Xiao. Panocontext: A whole-room 3d context model for panoramic scene understanding. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 668–686, Cham, 2014. Springer International Publishing.

[59] C. Zou, A. Colburn, Q. Shan, and D. Hoiem. Layoutnet: Reconstructing the 3d room layout from a single RGB image. *CoRR*, abs/1803.08999, 2018.

Appendix A

SUN360 pre-processing

One of the first tasks of the project was to download and pre-process SUN360 images and object 2D bounding boxes and labels.

When processing it, 273 different labels were identified, which are listed in Figure A.1. However, when having a quick look at them, it can be seen that the labelling process was not very careful, as there are many inconsistencies between names such as 'bedside table' vs. 'nightstand', many unmeaning differentiated categories, such as 'kitchen table' vs. 'table' vs. 'dining table' vs. 'coffee table', mixed labels such as 'chair and table', weird categories such as 'name' or 'cuboid' and even typo mistakes such as 'curtain' vs. 'cutrain'. There was also a distinction between objects detected inside of the room or outside (which could be seen though a door or window), a distinction that made no sense for our system.

Therefore, pre-processing was needed to try to unify and standardize categories by merging them in clusters. Simple text processing was conducted in Matlab, such as deleting non-alphanumeric symbols, unifying blank spaces, changing all to lowercase or identifying the main word in a group such as all labels containing 'table'. After that, we conducted a manual processing for unifying classes with orthographic mistakes or synonym names. Finally, the most significant classes (the ones with more number of occurrences) were chosen, as with less than 50 objects they were going to be too underrepresented in the dataset, selecting in the end the 14 categories used that are shown in Table 3.1.

On the other hand, 2D bounding boxes were not defined as the typical enclosing rectangle, but with point corners of the object, so the minimum enclosing rectangle was firstly obtained to define the bounding box four parameters: (x, y, w, h) being (x, y) the top left corner, w the width and h the height. The initial point annotations were mostly correct. Some incorrect ones were identified but not processed as we tried to avoid too many manual processes that were not the objective of the project.

'bed'	'door non-4pt-polygon'	‘fan:outside room '	'basket'	'oven'
'bed:outside'	'curtain'	'fan:outside room'	'switch'	'oven:outside room'
'bed:outside room'	'cutrain'	'heater'	'speaker'	'outside room oven'
'bed:outside room '	'curtai'	'heater:outside room'	'sounder'	'telephone'
'outside room bed'	'curtain:outside'	'screen'	'souder'	'pillow'
'baby bed'	'curtain:outside room'	'monitor'	'suitcase'	'faucet'
'painting'	'outside room curtain'	'tv'	' suitcase:outside room'	'faucet:outside room'
'paitning'	'shower curtain:outside'	'TV'	'drawer'	'paper towel'
'paint'	'chair'	'tv:outside room '	'barrel'	'towel'
'picture'	'chair:outside'	'TV set'	'fireplace'	'towel:outside room'
'picture: inside'	'chair: outside'	'computer'	'fireplace:outside room'	'blender'
'outside room picture'	'chair:outside room'	'laptop'	'TV stand'	'stereo'
'picture:outside room'	'chair:outside room '	'shelf'	'tv stand'	'book'
'picture: outside'	'chair: outside room'	'shelft'	'tv stand '	'ashtray'
'picture: outside room'	'outside room chair'	'shelf:outside room'	'rug'	'human'
'photo'	'deck chair'	'shelf:outside room '	'rug:outside room'	'socket'
'poster'	'deck chair:outside room'	'shelf: outside'	'outside room rug'	'plate'
'table'	'chair and table'	'book shelf'	'air condition'	'remote control'
'table:outside room '	'desk and chair'	'shelf: inside'	'air condition:outside room'	'toaster'
'table: outside room'	'light'	'ladder'	'flower'	'railings'
'table:outside room'	'lamp'	'stove'	'candle'	'railings:outside room'
'outside room table'	'lamp:outside room'	'stove:outside room'	'green plant'	'safe'
'round table'	'light:outside room '	'decoration'	'green plant:outside room'	'fire extinguisher'
'round table:outside'	'light:outside room'	'box'	'trash bin'	'toilet'
'dressing table'	'light stand'	'box:outside room'	'outside room trash bin'	'bottle'
'desk'	'bedside'	'clock'	'green'	'bag'
'desk:outside'	'beside'	'clock:outside room'	'stairs'	'toy'
'desk:outside room'	'outside room nightstand'	'microwave oven'	'outside room stairs'	'stroller'
'dining table'	'nightstand'	'microwave:outside room'	'stairs:outside room'	'condition'
'dining table '	'nightstand:outside'	'microwave: outside room'	' stairs:outside room'	'keyboard'
'dining table:outside ‘	'bedhead'	'bench'	'sink'	'fruits'
'dining table:outside’	'sofa'	'diagram'	'sink:outside room '	'kettle'
'dining table: outside’	'sofa:outside'	'glass'	'sink:outside room'	'shoes'
'outside dining table'	'sofa:outside room'	'cup'	'dishwasher'	'scale'
'console table'	'sofa:outside room '	'vase'	'dish washer'	'guitar'
'console table '	'outside room sofa'	'vase:outside room'	'dishwasher:outside room'	'beer'
'console table'	'cabinet'	'windowsill'	'microwave'	'newspaper'
'console table:outside'	'cabinet:outside'	'washtub'	'cloth'	'overhead projector'
'coffee table'	'cabinet:outside room'	'bathtub'	'piano'	'hat'
'coffee table:outside'	'cabinet: outside room'	'washing machine'	'water heater'	'stage'
'end table'	' cabinet:outside room'	'washing machine:outside '	'water heater:outside room'	'map: inside'
'bar table'	'outside room cabinet'	'trash can'	'water heater: outside '	'tel'
'bar table:outside room '	'wardrobe'	'stair'	' water heater:outside'	'target'
'kitchen table'	'wardrobe:outside'	'machine'	'outside room refrigerator'	'cubicle'
'mirror'	'aircon'	'board'	'outside room microwave'	'cubicle:outside room'
'mirror:outside room'	'refrigerator'	'wall'	'plant'	'rectangle'
'outside room mirror'	'refrigerator:outside room'	'player'	'black board'	'cuboid'
'window'	'refrigerator: outside room'	'printer'	'pillow:outside room'	'room'
'window:outside'	'blanket'	'fish tank'	'range hood'	'room non cuboid'
'window:outside room'	'cushion'	'cloth stand'	'outside room range hood'	'non-cuboid room'
'window:outside room '	'column'	'cloth hook'	'range hood:outside room'	'unnamed'
'window: outside room'	'colunm'	'safebox'	'coffee maker'	'unamed'
'outside room window'	'pillar'	'calendar'	'coffee maker:outside'	'name'
'door'	'pillar: outside room'	'bucket'	'white board'	
'doorway'	'fan'	'writing'	'overhead'	

Figure A.1: List of all 273 classes labels. Final 14 clusters used are marked in colors.

Appendix B

Further results

Extra quantitative and qualitative results of the system, that extend the ones presented on Chapter 5 are included in this Appendix.

Preliminary experiments

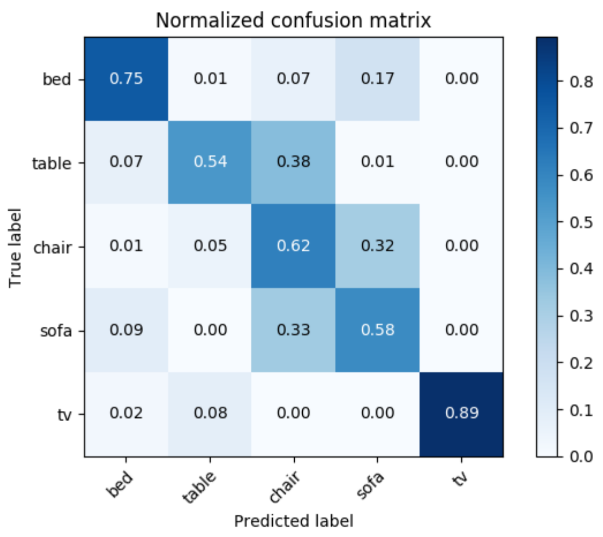


Figure B.1: YOLO preliminary experiment: Confusion matrix normalized, detailed for detections of different common categories. Notice how results from tv category are very reliable, but sofas, tables and chairs are often confused.

More Experiments

SUN360	BlitzNet Scratch mAP^w (45k iter.)	BlitzNet ImageNet mAP^w (23k iter.)	BlitzNet SUN RGB-D mAP^w (65k iter.)
bed	0.834	**0.854**	0.849
painting	0.662	0.674	**0.719**
table	0.575	0.558	**0.592**
mirror	0.421	0.399	**0.563**
window	0.389	0.368	**0.448**
curtain	0.378	0.324	**0.419**
chair	0.359	0.350	**0.416**
light	**0.209**	0.193	0.190
sofa	0.514	**0.654**	0.633
door	0.472	0.402	**0.495**
cabinet	0.287	**0.317**	0.300
bedside	0.574	0.619	**0.680**
tv	0.737	0.748	**0.796**
shelf	0.148	**0.245**	0.122
TOTAL	0.468	0.479	**0.516**

Table B.1: **Effect of initialization:** Detailed results [they extend Table 5.2] separated in categories of different initialized models. Trained SUN360 train-seg, and tested on SUN360 test-seg, with BlitzNet300 square model.

DETECTION *mAP*	Panoramic BlitzNet StandardConvs (102k iter.)	Panoramic BlitzNet EquiConvs (111k iter.)
bed	0.874	**0.863**
painting	**0.798**	0.777
table	**0.817**	0.781
mirror	0.768	**0.795**
window	**0.726**	0.707
curtain	**0.810**	0.746
chair	0.756	**0.779**
light	0.411	**0.432**
sofa	**0.836**	0.784
door	**0.745**	0.687
cabinet	**0.595**	0.505
bedside	0.859	**0.861**
tv	0.849	**0.852**
shelf	0.317	**0.411**
TOTAL *mAP*	**0.764**	0.739

DETECTION mAP^w	Panoramic BlitzNet StandardConvs (102k iter.)	Panoramic BlitzNet EquiConvs (111k iter.)
bed	**0.842**	0.833
painting	0.676	**0.692**
table	**0.630**	0.608
mirror	**0.334**	0.330
window	**0.523**	0.453
curtain	0.412	**0.475**
chair	0.408	**0.439**
light	0.123	**0.137**
sofa	0.712	**0.735**
door	0.404	**0.463**
cabinet	**0.356**	0.351
bedside	**0.716**	0.706
tv	**0.861**	0.817
shelf	**0.042**	0.033
TOTAL mAP^w	0.534	**0.548**

Table B.2: **Effect of EquiConvs on object detection:** Detailed results, separated in categories, which show differences in object detection task with both different *mAP* metrics, as explained in Section 5.1. Both are Panoramic BlitzNet model, trained on SUN360 train-seg with SUN RGB-D initialization and tested on SUN360 test-seg, but this time with a confidence threshold of 0.5. Remember mAP^w results, which are generally lower, are a better approximation of the Precision-Recall curve but *mAP* benefits models with high precision. With this given confidence level, standard convolutions have higher precision (higher *mAP*) but clear less recall than EquiConvs (lower mAP^w).

Qualitative results

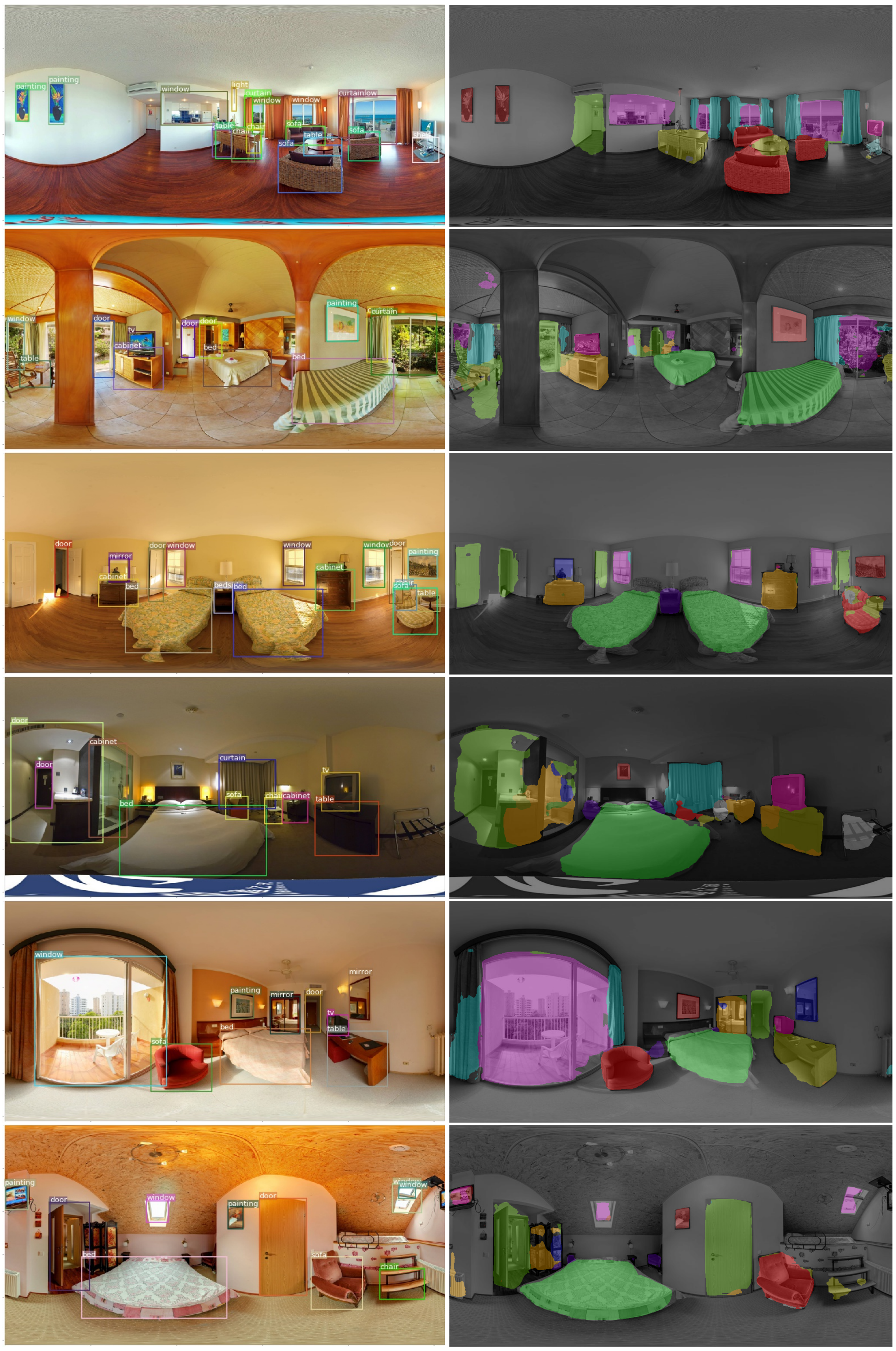


Figure B.2: **Qualitative results of our Panoramic BlitzNet:** Model trained with EquiConvs on SUN360 train-seg. Experiments conducted on SUN306 test-seg.

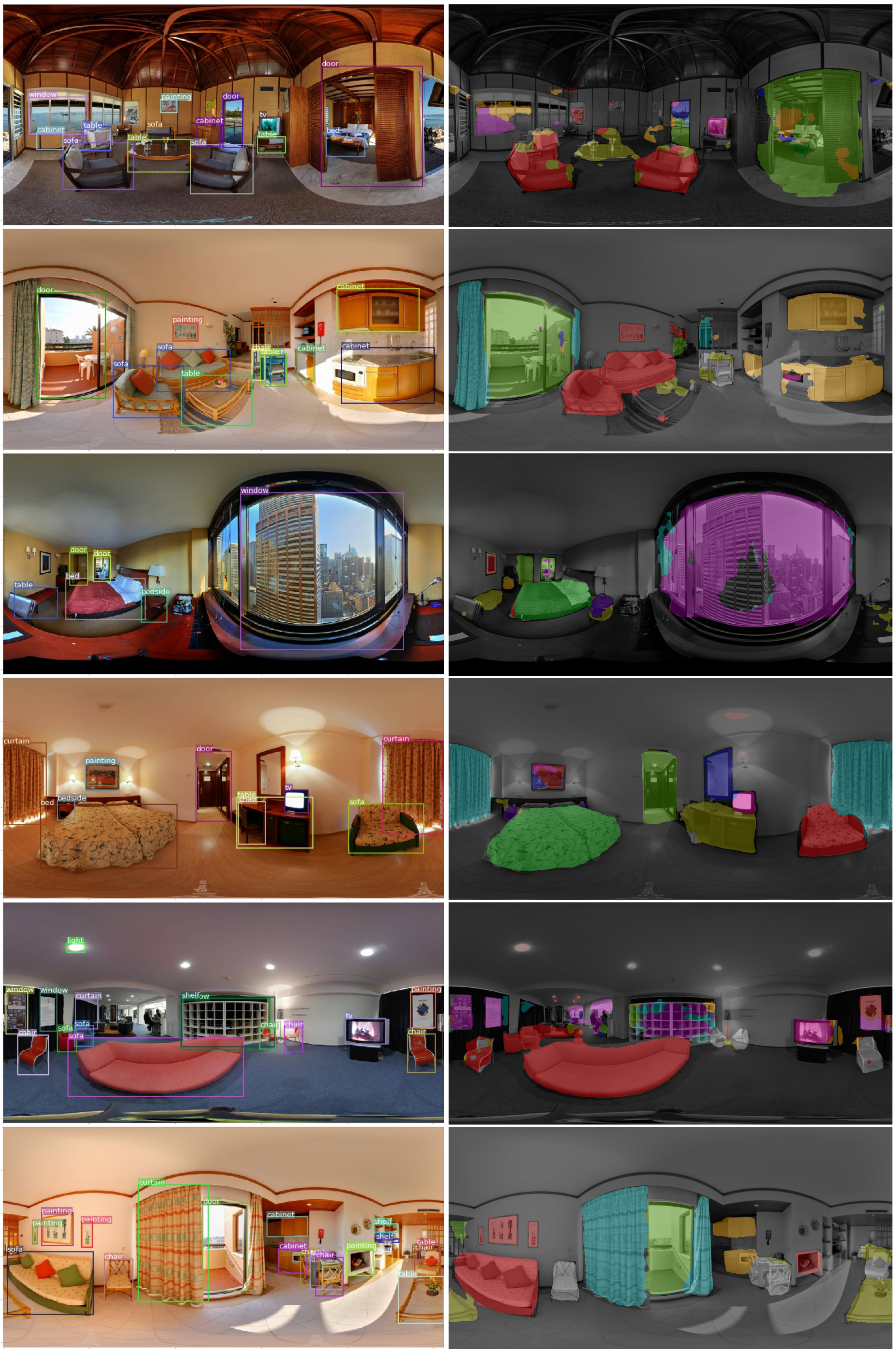


Figure B.3: **More qualitative results of our Panoramic BlitzNet:** Model trained with EquiConvs on SUN360 train-seg. Experiments conducted on SUN306 test-seg.

Comparison with state of the art

SEGMENTATION *meanIoU*	PanoContext	Panoramic BlitzNet EquiConvs
background	0.869	**0.913**
bed	**0.786**	0.621
painting	0.387	**0.612**
table / desk	0.296	**0.723**
mirror	0.382	**0.414**
window	0.356	**0.534**
chair	0.096	**0.552**
sofa	0.111	**0.329**
door	0.194	**0.638**
cabinet / wardrobe	0.274	**0.511**
bedside / nightstand	**0.397**	0.366
tv	0.348	**0.523**
TOTAL *meanIoU*	0.375	**0.561**

Table B.3: **Comparison with state of the art:** Detailed results, separated in categories, which compare differences in semantic segmentation task (*meanIoU*) between our Panoramic BlitzNet model using equirectangular convolutions and the state of the art from PanoContext [58] work. Only categories that they have in common have been selected.

Appendix C

Project management

C.1 Hardware and software tools

Different software tools have been used during the project, which are listed below specifying the version, to help future work.

Operating system was Linux Ubuntu 16.04, with CUDA 10.0, cuDNN 7.5, and Nvidia drivers 418.67. Initial processing was conducted in Matlab R2018-A and the rest of the implementation has been done in Python 3.5 using the Google Machine Learning framework, Tensorflow v1.13.1. Other common libraries such as Numpy (Version 1.16.4), OpenCV (version 4.0) or PIL (version 6.0) have also been installed and used.

Specific hardware was also required for training the CNN model. A Nvidia GeForce GTX 1080 GPU has been used for most of the experiments, except from the last ones with EquiConvs, which were trained on a borrowed Nvidia Titan V.

C.2 Workload and planning

Regarding project management, it has been developed within 'Robotics, Perception and Real Time' Group. The project was conducted individually, researching on literature and under the help of my directors Clara Fernández Labrador and Josechu Guerrero Campo, with whom we arranged periodical meetings to control the progress of the project.

The project started in October 2018 and finished at the end of June 2019, with a complete duration of nine part-time months. Planning of different activities within the project is summarized in the following Gantt diagram[C.1].

Hours of work were reported and sum a total of approximately 440 hours. The detailed workload of the main parts of the project is included in the table below[C.2].

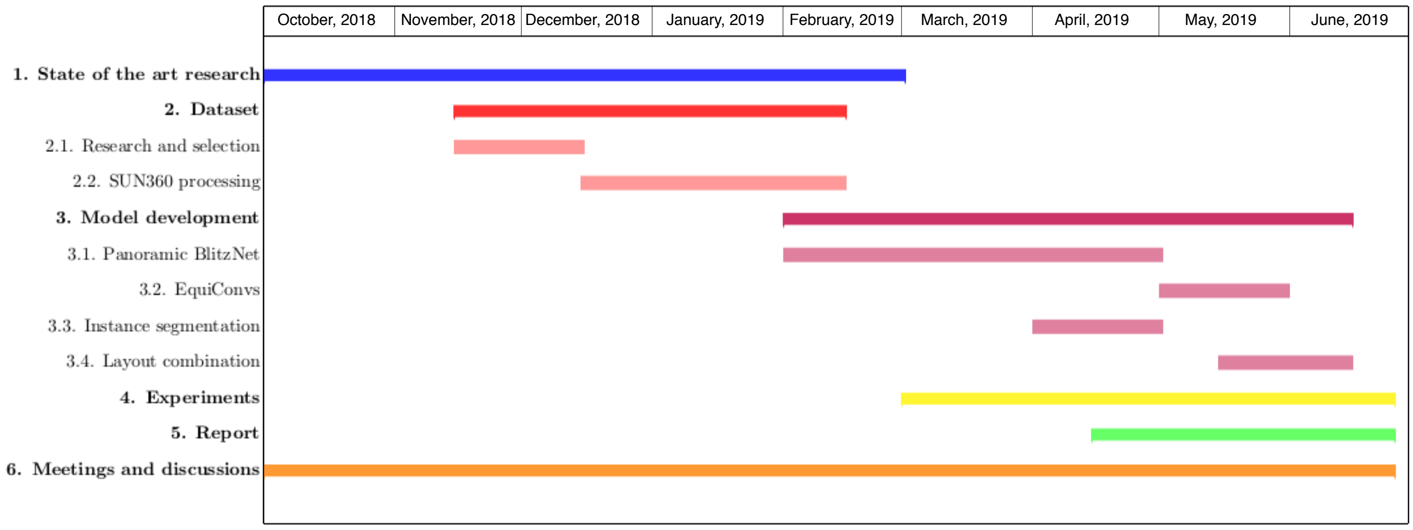


Figure C.1: **Project planning:** Gantt diagram showing the development of the project

Task	Time (hours)
State of the art research	90
Dataset selection and processing	40
Model development	125
- *Panoramic BlitzNet*	*60*
- *EquiConvs*	*25*
- *Instance segmentation*	*20*
- *Layout combination*	*20*
Experiments	70
Report	85
Meetings and discussions	30
Total	**440**

Figure C.2: **Project workload**