UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

MATHEUS ALAN BERGMANN

Upright Adjustment of Spherical images

Work presented in partial fulfillment of the requirements for the degree of Bachelor in Computer Science

Advisor: Prof. Dr. Claudio Rosito Jung
Coadvisor: Prof. Dr. Thiago Lopes Trugillo da Silveira

Porto Alegre
June 2021

ABSTRACT

Spherical cameras capture all the information around them with a field of view (FOV) of $360° \times 180°$. The underlying images, called spherical, omnidirectional or panoramic images encounter several applications such as immersive exploration when visualized using a Virtual Reality (VR) headset. For these and many other applications, the captured view should be aligned with the viewing subject. This means that the upright vector of the camera/image should be aligned with the gravity vector (which also means that the horizon of the panorama should be aligned with the horizon of the Earth). However, images or video sequences captured by spherical cameras might not be gravity-aligned, and feeding them directly to the headset might cause uncomfortable sensations to the user. In this work, we use Computer Vision and Machine Learning techniques to estimate the upright vector of a panorama, which can be used for gravity-alignment. We also show an application of our method by coupling the proposed upright-adjustment procedure with a single-panorama depth estimation method, and show that our pre-processing step helps improving depth estimates.

Keywords: Upright Adjustment. Spherical Images. Computer Vision. Machine Learning.

Ajuste do *upright* de imagens esféricas

RESUMO

Câmera esféricas capturam toda informação ao seu redor com um campo de visão de 360°×180°. As imagens resultantes, chamadas de esféricas, omnidirecionais ou panoramicas possuem diversas aplicações como exploração imersiva quando visualizadas através de um óculos de realidade virtual. Para essa e muitas outras aplicações, a vista capturada deve estar alinhada com objeto observado. Isso significa que o vetor *upright* da camera/imagem deve estar alinhado com o vetor da gravidade (o que também sigifica que o horizonte da imagem panoramica deve estar alinhado com o horizonte da Terra). Porém, imagens ou sequencias de vídeos capturadas por câmeras esféricas podem estar desalinhadas em relação à gravidade, e apresentá-las diretamente no óculos de realidade virtual pode causar desconforto ao usuário. Nesse trabalho é proposto o uso de técnicas de Visão Computacional e Aprendizado de Máquina para estimar o vetor *upright* de uma imagem panorâmica, que pode então ser usado para alinhá-la com a gravidade. Também é mostrada uma aplicação acoplando o procedimento proposto de ajuste do vetor *upright* com um métodos de estimativa de profundidade em uma única imagem, e mostra-se que esse passo de pré-processamento ajuda a melhorar as profundidade estimadas.

Palavras-chave: Ajuste do *upright*, Imagens Esféricas, Visão Computacional, Aprendizado de Máquina.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

CNN	Convolutional Neural Network
GCN	Graph Convolutional Network
DoF	Degrees of Freedom
FoV	Field Of View
MSE	Mean Squared Error
SGD	Stochastic Gradient Descent
SOTA	State-of-the-art
VP	Vanishing Point
VR	Virtual Reality

LIST OF SYMBOLS

δ	Accuracy
β	Pitch
γ	Roll
ϕ	Latitude
λ	Longitude
e	Angular error
$\boldsymbol{v}_{gt}$	Ground truth vector
$\boldsymbol{v}_{out}$	Output vector
$< \cdot, \cdot >$	Inner product.

CONTENTS

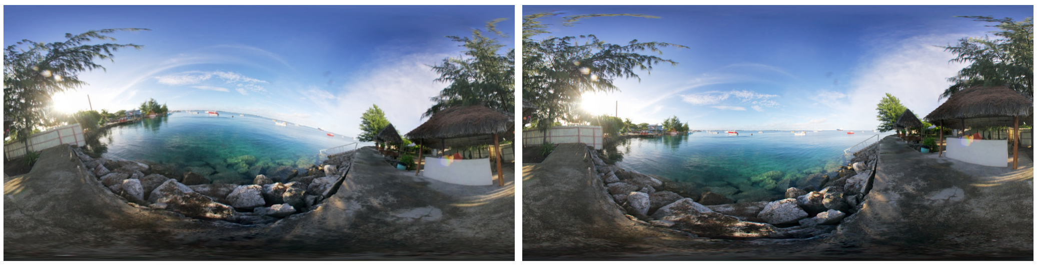
1 INTRODUCTION

Spherical images, also called omnidirectional, panoramas, or 360 images, are images captured by a camera with a full horizontal field of view of 360° and a full vertical field of view of 180°. In the last few years, these cameras have become cheaper and affordable by non-professional photographers who are more likely to acquire images with a tilted camera setup, a problem that is also common in perspective images. In spherical images, however, there is enough information to automatically correct the photo (since the horizon must be present unless occluded), as long as the rotation applied during the capture is known. And when such rotation is not known, it can be estimated. Figure 1.1 shows an example before and after the upright adjustment (or gravity-alignment).

Providing a full field of view (FoV) allows a series of applications. For example, panoramas have been widely used in the real state market, allowing buyers to have a comprehension of the whole ambient with a single shot. Websites like Facebook and YouTube have also added support for this media type, allowing users to consume omnidirectional content even without a Virtual Reality (VR) headset. However, to enjoy a more immersive experience, a VR headset is recommended because it provides the users the ability to look around the space just by moving their head, as long as a proper projection of a spherical sector to a narrower-FoV image is used. Panorama images shine when computer vision algorithms are used to extract data and process large volumes of data, creating applications such as Google Street View.

In this work, we present a deep learning-based approach to estimate the camera upright vector and rotate the image to generate a gravity-aligned version of the scene. It enables post-capture processing of the full-FoV image and can be used either to improve the image visualization or as a pre-processing step for rotation-sensitive methods, such as single-panorama depth estimation (ZIOULIS et al., 2018) and layout inference (ZOU et

Figure 1.1 – Example of a VR photo before (left) and after (right) the upright adjustment. In this case, the image in the left was synthetically rotated and the image in the right is the ground truth.



al., 2018).

It is important to mention that state-of-the-art (SOTA) methods that regress the upright vector suffer from high errors when the image is nearly aligned or upside-down (JUNG et al., 2019), so that the gravity-aligned images in these cases might be even more tilted than the input.

1.1 Goals

Our main goal in this work is to develop an upright vector estimation approach that produces roughly stable, acceptable errors for any input rotation. The following specific goals are defined to achieve this main goal:

1. Review existing methods for upright adjustment;
2. Evaluate the use of a classification model to detect aligned and upside-down images;
3. Evaluate an alternative representation with respect to SOTA methods of the rotation as the regression output of the model;
4. Evaluate the use of a spherical-adapted backbone to regress the upright vector;
5. Use the proposed method to improve single-view depth estimation.

1.2 Chapters organization

The rest of this document is organized as follows: Chapter 2 introduces a few important concepts about spherical images and upright adjustment. Chapter 3 presents an overview of related works in both spherical images and upright adjustment. Chapter 4 describes how we prepare our data and presents three machine learning models. Chapter 5 describes the results achieved by each of the models and shows how they can be used to improve other computer vision tasks (e.g., single-image depth estimation). Chapter 6 provides an overview of everything presented, evaluate progress toward our goals, and discusses future work.

2 THEORETICAL FOUNDATION

Spherical images present a full-FoV of 360°×180°. Thus, a 360° image can be seen as a mapping $i(\phi, \lambda)$ that relates a longitude $\lambda \in [0°, 360°)$ and a latitude $\phi \in [0°, 180°)$ Although the image i is defined on the unit sphere (Figure 2.1), it can be represented in several ways, including the equirectangular projection - a $[0°, 360°) \times [0°, 180°)$ planar representation and widely used by researchers and camera manufacturers (SU; GRAUMAN, 2017; SILVEIRA; JUNG, 2020) - (Figure 2.2), and the cube map projection (Figure 2.3). In immersive VR applications, one can prepare the image i for consuming in VR headsets - which considers two perspective crops with a small FoV and a small displacement (Figure 2.4), each one for each eye.

Ideally, the 3D unit vector related to $\phi = \lambda = 0$, which is related to the "up" vector of the camera, should be aligned with the earth gravity vector (and pointing up), so that the horizon of the panorama would match the horizon of the world. However, a rotation R might be accidentally applied during the capture and the resulting image ends up rotated, as in Figure 2.5. If we know the rotation $R(i, \boldsymbol{v})$ - that makes the up vector of the camera point towards $\boldsymbol{v}$ - applied during the capture the rectified image could be obtained by applying the inverse of R to the image $i_{rect} = R^{-1}(i, \boldsymbol{v})$, where R^{-1} is the inverse of R. This process of realigning the image is known as upright adjustment (LEE et al., 2013; JUNG et al., 2017; JEON; JUNG; LEE, 2018; JUNG; CHO; KWON, 2020).

Figure 2.1 – Image projected to a unit sphere. Extracted from Coors, Condurache and Geiger (2018)

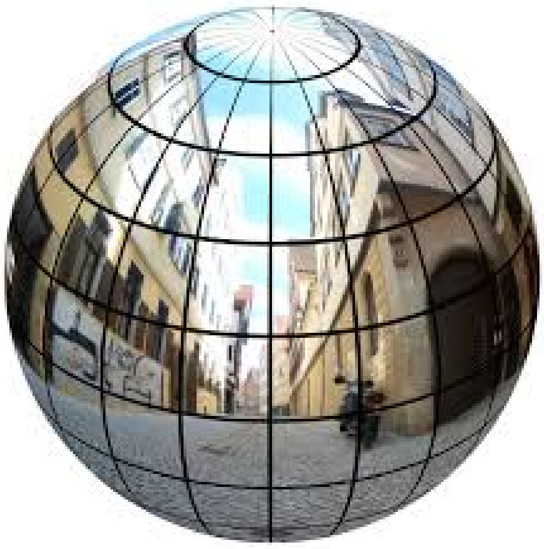


Figure 2.2 – Image using equirectangular projection. Extracted from SUN360 (J. Xiao et al., 2012)

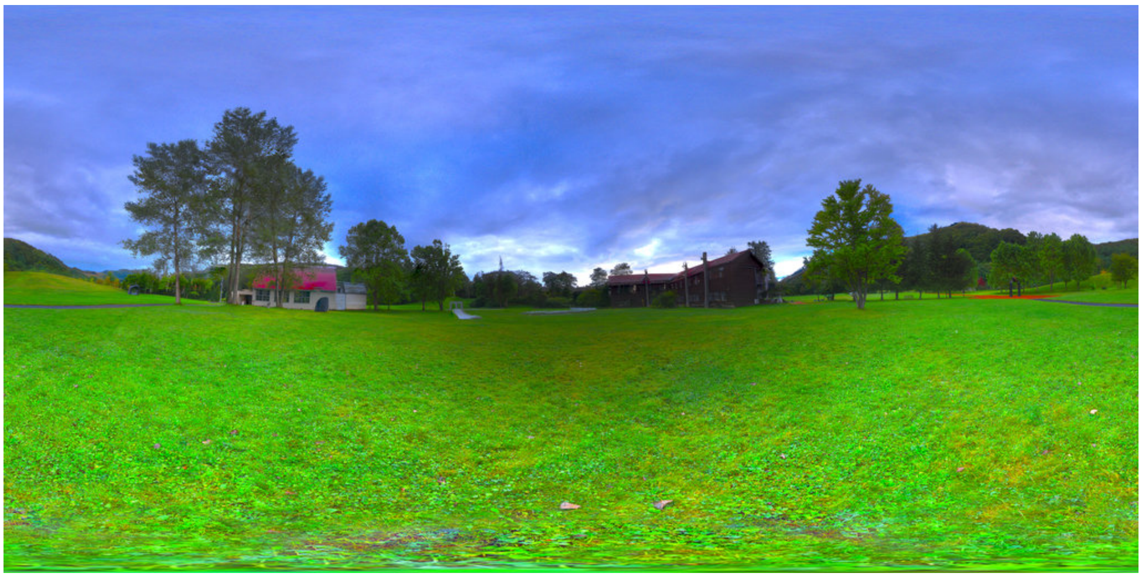


Figure 2.3 – Image projected to a cube (cube map). Extracted from Wan, Wong and Leung (2007)

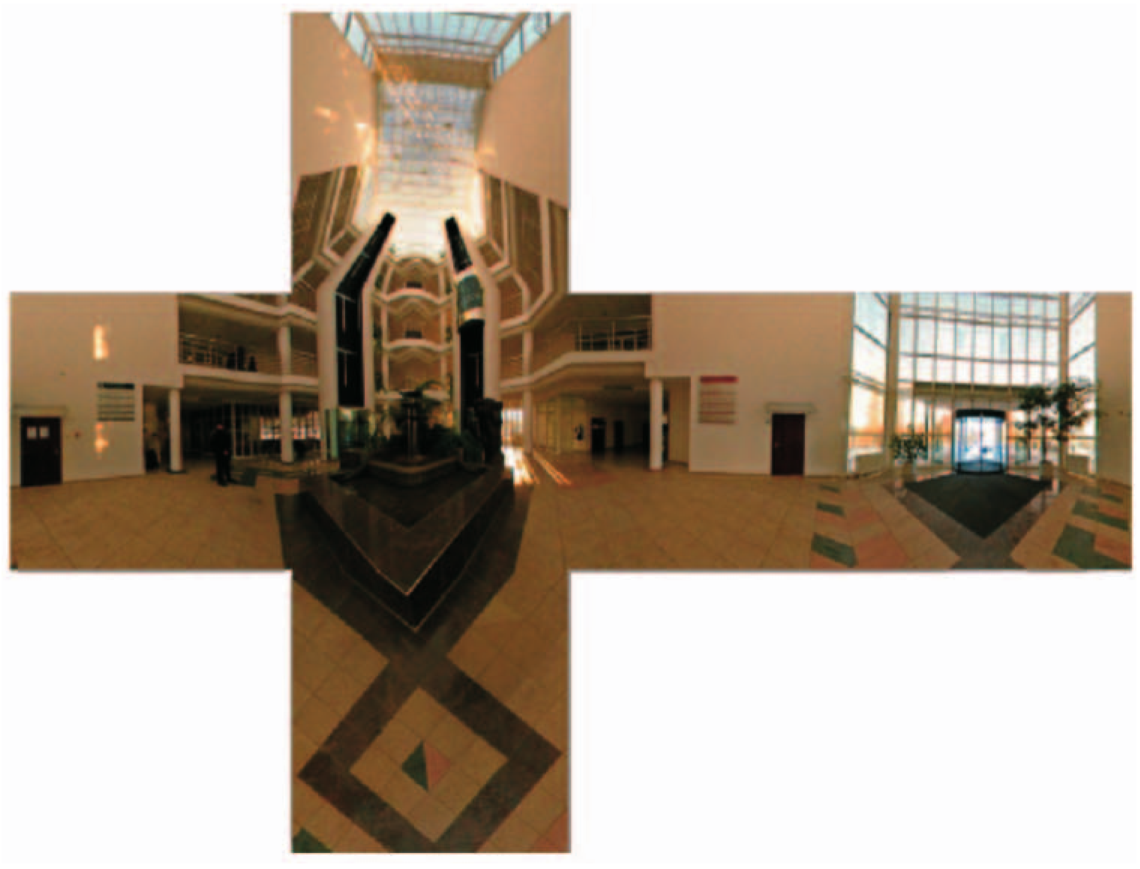


Figure 2.4 – Image ready to be used in a VR headset. Extracted from Jung et al. (2017)

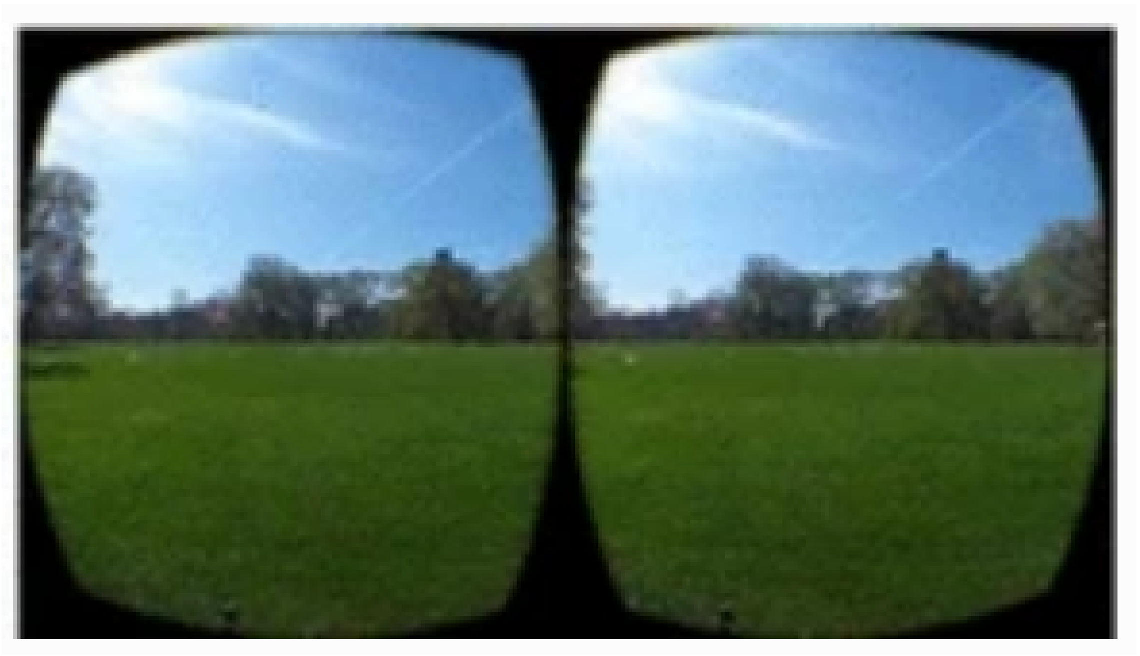
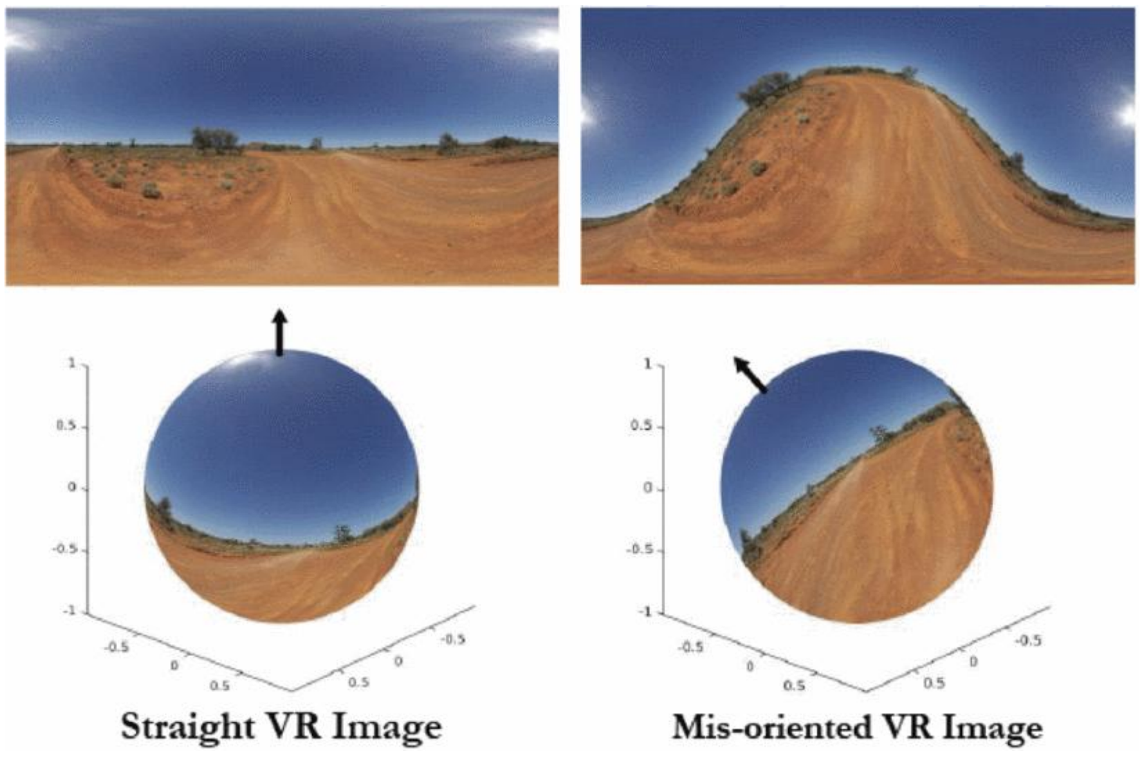


Figure 2.5 – Image showing an aligned and a tilted image extracted from Jung et al. (2019).



3 RELATED WORK

3.1 Spherical Images

As we have mentioned before, VR headsets are one way of visualizing panorama images, which involves projecting spherical data to the plane, generating a narrower-FoV image. These projections may allow the use of traditional visual computing methods (designed to work with perspective images), but working directly in image representations of the full sphere might fail due to distortions (EDER et al., 2019). The most common representation is the equirectangular projection (EDER et al., 2019; SILVEIRA; JUNG, 2019), which maps a unit sphere to a rectangular domain related to latitude and longitude angles. As such, this representation allows the use of various methods designed for traditional perspective images, including Convolutional Neural Networks (CNNs). However, the equirectangular projection generates non-uniformly sampled images, which causes small portions of data to occupy large portions of the projections (particularly near the poles) so that using a fixed convolutional kernel might be less efficient in the feature extraction process.

Many techniques were proposed to mitigate the effects of these distortions. They include converting the equirectangular image to a graph and using Graph Convolution Networks (GCNs) (KHASANOVA; FROSSARD, 2017), using dilated convolutions (YU; KOLTUN, 2015) to increase the size of the convolution kernel near the poles (SU; GRAUMAN, 2017), using spherical convolutions (COHEN et al., 2018; COORS; CONDURACHE; GEIGER, 2018) and transforming kernels from traditional perspective networks to the spherical domain (SU; GRAUMAN, 2019).

Another research direction is the use of other image representations, such as the cube map projection (ABREU; OZCINAR; SMOLIC, 2017; RUDER; DOSOVITSKIY; BROX, 2018) in which the panorama is projected to six planar regions (the faces of a cube). Even though this projection still has distortions, it reduces their effects, but introduces discontinuities in the image. This means a trade-off between less distortion and contextual information (SILVEIRA; DALAQUA; JUNG, 2018). There are also several other representations, including a few based on icospheres (EDER et al., 2019), but as far as we know they have not been used for any CNN-based upright adjustment application.

3.2 Upright Adjustment

As mentioned before, upright adjustment can be applied in perspective images as well. There are several methods (GALLAGHER, 2005; LEE et al., 2013) that automatically perform the correction in perspective images, including some based on deep learning (FISCHER; DOSOVITSKIY; BROX, 2015; JOSHI; GUERZHOY, 2017; HOLD-GEOFFROY et al., 2018; OLMSCHENK; TANG; ZHU, 2017; ZHANG et al., 2016). However, in this section, we will focus on those that perform upright adjustment in spherical images only.

3.2.1 Non-learning Approaches

Non-learning approaches usually explore geometric cues from the images, such as lines and vanishing points (VP). Computing the vertical VP indicates the upright direction. Even though it is possible to compute each VP independently (BAZIN et al., 2008; BAZIN et al., 2010), using geometric constraints as the Manhattan (BAZIN et al., 2012; BAZIN; POLLEFEYS, 2012; BAZIN; SEO; POLLEFEYS, 2012; ZHANG et al., 2016) or Atlanta (JOO et al., 2018; JUNG et al., 2017) worlds usually yield better results. One thing that is common to all these methods is that they depend on the ability to distinguish between vertical and horizontal vanishing points. These VP-based methods tend to work well in indoor and urban scenes, but tend to fail in natural images due to the lack of structural information.

Another way to estimate the upright vector is to find the horizon line (DEMONCEAUX; VASSEUR; PÉGARD, 2006a; DEMONCEAUX; VASSEUR; PÉGARD, 2006b). The horizon line can be estimated by maximizing a criterion for the sky/ground photometric separation. In general, these methods are more suited to natural views (particularly when there is a clear distinction between the sky and the ground) and tend to fail in urban and indoor scenarios.

3.2.2 Deep Learning Approaches

In recent years, deep learning (DL) methods have produced SOTA results in many computer vision tasks, such as image classification (HE et al., 2016), segmentation (RON-

NEBERGER; FISCHER; BROX, 2015) and object detection (REDMON; FARHADI, 2018; REN et al., 2015). However, the use of DL for upright vector estimation in a generic capture scenario (indoor, outdoor, urban and natural scenes) is relatively recent, particularly when using panoramas.

Jeon, Jung and Lee (2018) propose a CNN that predicts the rotation in common perspective images. To predict the upright in spherical images it generates several crops from a panorama and then aggregates all the results to predict the rotation to the 360 image, which means more pre- and post-processing than a direct regression using the full panorama as input.

Shan and Li (2019) propose two classification models to perform first a coarse alignment (classifying within bins of 10°) and after that a fine alignment (classifying within bins of 1°). Besides using two models, this method cannot fully describe the domain of the problem, generating error even when all the classifications are correct.

Davidson, Alvi and Henriques (2020) use VPs to help a CNN segment pixels near the vertical axis. This segmentation is used to find the vertical axis and to estimate the upright direction. Segmentation models usually have more layers and are slower than direct regression methods because after the feature extraction it has several up-scaling layers instead of a fully connected one.

A recent method (JUNG; CHO; KWON, 2020) proposes the use of a CNN to extract features and a GCN that finds a spherical representation of the input. Using a GCN is slower than a fully connected layer. However, it also does not address the circularity problem (explained in Chapter 4).

The Deep360Up model (JUNG et al., 2019) uses a DenseNet backbone with a fully connected layer to regress the two relevant angles and align the horizon, and serves as the basis for our method. We chose this method because it is simple to reproduce, applies direct regression, and therefore is faster than the other methods presented in this section. Despite the simple architecture and good overall results, the main issue in their technique is the high errors for nearly aligned images. This happens because of the circularity problem - which will be further discussed in Chapter 4 - which is tackled in our work. We also evaluate a backbone with spherical convolutions, but the tested implementation presented higher execution time and also higher errors than the backbone with traditional convolutions.

4 THE PROPOSED METHOD

Our solution is inspired by Jung et al. (2019), but focused on solving the large errors reported by the authors in nearly aligned and upside-down images, which we believe to be caused by the "circularity problem". The circularity problem is a consequence of the chosen rotation parametrization and happens when very distinct parameters lead to very similar rotations. For example, the parametrization using Euler angles does present this problem, since angles close to $0°$ and close to $360°$ lead to very similar rotations. In the context of deep learning, regressing such parameters directly might lead to instabilities or bad results, since samples that are visually very similar (with angles close to $0°$ or $360°$) generate similar values (because CNNs are continuous functions). Figure 4.1 shows examples of such images.

In this work we propose two alternatives to overcome this issue. One of them is by classifying edge cases and dealing with them separately. Another one is using a different rotation parametrization than the one used by Jung et al. (2019), Jung, Cho and Kwon (2020). In the following sections, we describe what we changed with respect to Jung et al. (2019). Meta parameters such as learning rate, batch size and image size were chosen empirically and, along with rotation parametrization, reduced the training time by a factor of 4. Section 4.1 sets some standards on data preparation that are used in all the models. Section 4.2 proposes the use of a classification model to detect edge cases. Section 4.3 suggests the use of a different parametrization to the rotation to overcome the circularity problem. Finally, Section 4.4 evaluates a backbone with spherical convolutions.

4.1 Data Preparation

The dataset is probably the single most important part of the development of deep learning models. To the best of our knowledge, there is no dataset with upright labels

Figure 4.1 – Two similar images with different parametrizations. Left image has roll of 5° and the right image a roll of 355°.

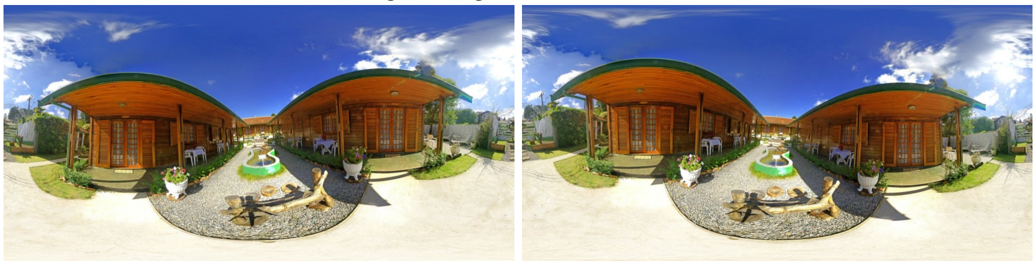


Figure 4.2 – Example of rotations. From top to bottom: original, roll of 60°, pitch of 60° and yaw of 60°, respectively. The original image is part of the SUN360 dataset and the others were synthetically rotated.

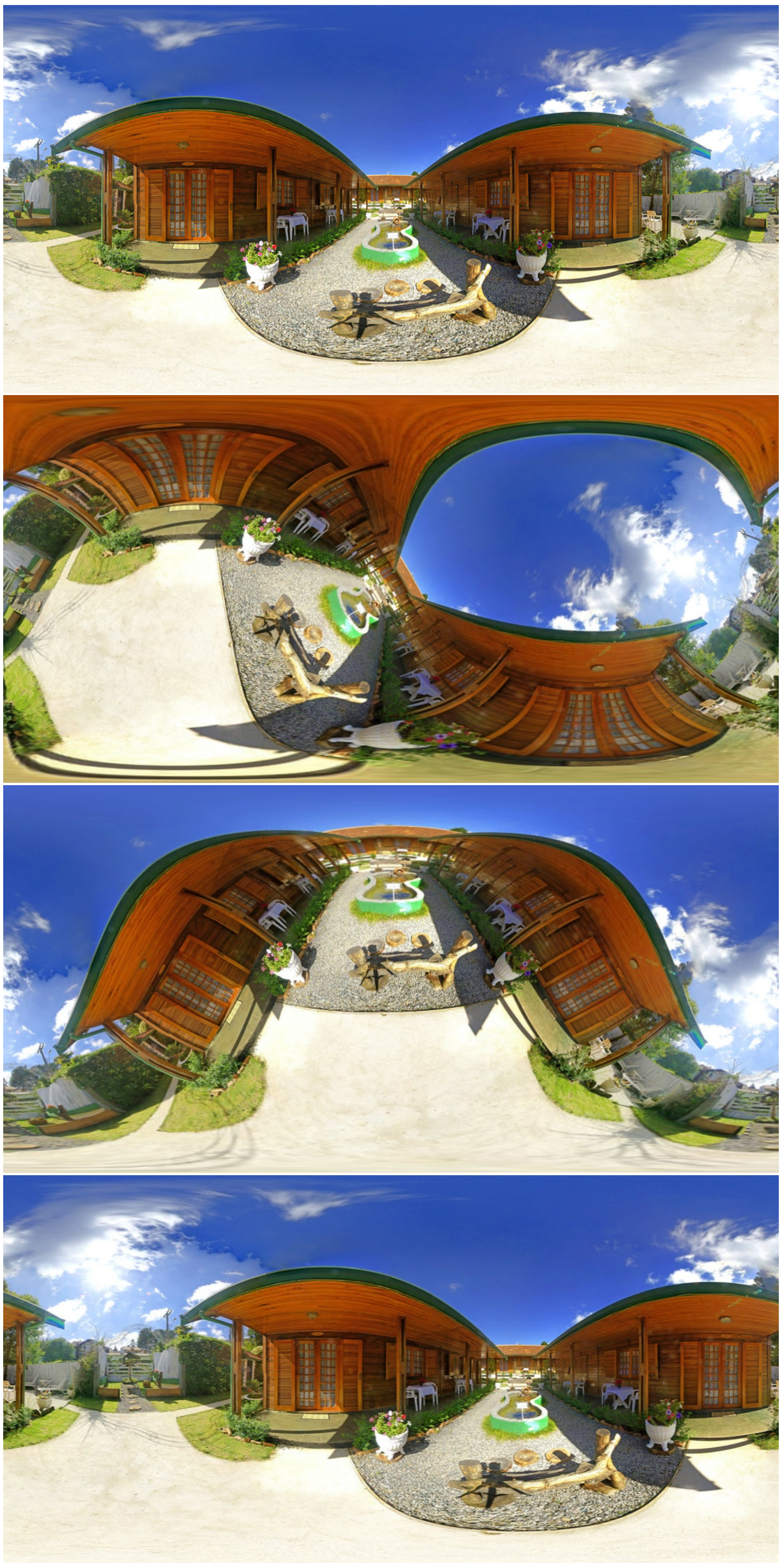
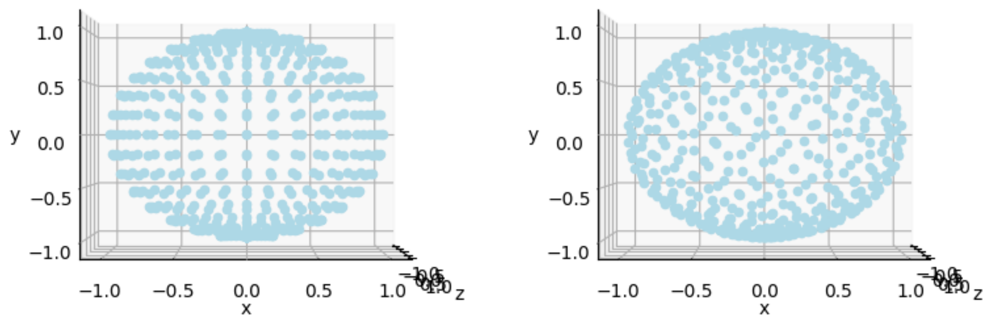


Figure 4.3 – In the left the vectors positions in the uniform distribution in a 2D grid. In the right the positions using Fibonacci Lattice. Both examples contain 512 points each.



that contains enough images to train a CNN. As previously done by Jung et al. (2019), Jung, Cho and Kwon (2020) we use a dataset from another task and assume all images are originally horizon aligned. To allow comparisons, we use SUN360 dataset (J. Xiao et al., 2012), the same as used in Jung et al. (2019), Jung, Cho and Kwon (2020). The dataset contains $67,569$ captures in both indoor and outdoor scenarios and the images are all roughly aligned.

There are two relevant angles in the upright adjustment problem - Pitch and Roll. The other angle, Yaw, only performs a rotation around the gravity direction, which translates to the image as a horizontal shift and keeps the horizon line unchanged. Figure 4.2 shows the result of varying each rotation angle independently. To create the variability necessary to train our CNN, we artificially rotate images. Generating random uniformly distributed values on a planar grid for both Pitch and Roll would create an oversampling of vectors near the poles. Therefore, we use a Fibonacci Lattice (STANLEY, 1975) to generate vectors as well distributed as possible on the sphere surface. Figure 4.3 shows the difference between sampling the upright vectors using a 2D uniform distribution and Fibonacci Lattice.

Applying rotation to low-resolution images produces artifacts that interfere in the learning process, as noted by Jung et al. (2019). However, high resolutions images (9104×4552) from SUN360 are not available anymore, and we only have access to 1024×512 images. In the attempt to prevent the CNN from learning these artifacts, we add noise to create random artifacts and blur to make the existing ones less visible. We add Gaussian

Noise in 40% of the images with a variance between 10 and 50 (255 is white) and blur in 50% of the images using a normalized box filter with a kernel size of 3, 5, or 7 (randomly chosen). Both noise and blur can be applied simultaneously to the same image.

We also performed a cleaning step in the data. Besides some corrupted files, several repeated images were removed from the SUN360 database. The resulting (cleaned) dataset has 57.011 images, 10.558 less than the original one.

Evaluating a model always boils down to splitting the available data into three sets: training, validation, and test (CHOLLET et al., 2018). In our work, we randomly split the images using the same proportion as Jung et al. (2019): 70% of the images as train, 15% as validation and 15% as test set. After the split, using a Fibonacci Lattice, one unit vector was generated for each image in each set and the upright vectors were adjusted to point in that direction.

Before being used by the network, all images are resized. The size depends on the model and will be described in the following sections.

4.2 Classifying aligned images

One major problem with SOTA methods that regress the upright vector is that they do not deal well with neither ground-aligned nor upside-down images. A naive solution would be to manually select the images that are roughly aligned or upside-down and do not pass them through the model. This would, however, be time-consuming, susceptible to errors, and an obstacle to automatically processing large amounts of data. During the course of this work, we evaluated the possibility of automatically classifying the images as aligned, upside-down, or rotated.

For this purpose, we explored a network with a ResNet50 backbone (HE et al., 2016). More precisely, we used pre-trained weights for the convolutional layers using ImageNet (DENG et al., 2009), whereas the fully connected (classification) layer was trained from scratch. We consider as aligned images with $-5^\circ \leq \beta, \gamma \leq 5^\circ$ and as upside-down images with $-5^\circ \leq \beta \leq 5^\circ$ and $175^\circ \leq \gamma \leq 185^\circ$, where β is the pitch and γ is the roll angle. The set of aligned images also includes the original image (without any rotation). To train the model, we used the same data splits as mentioned in Section 4.1 with a difference in how we generated the virtual camera poses (rotation only). To keep the data balanced, each image appeared three times: once aligned, once upside-down, and once with a random rotation.

The model was optimized for 100 epochs using the Stochastic Gradient Descent (SGD) (ROBBINS; MONRO, 1951; KIEFER; WOLFOWITZ et al., 1952) with a learning rate tailored for each layer: 0.001 in the pre-trained layers, and 0.01 in the classification layers, aiming to update the (initially untrained) classification layers faster. An L2 regularization of $\lambda = 0.001$ is applied to all layers.

In this model, input images have 224×112 pixels, and we selected a batch size of 32 images. We use the traditional Cross-Entropy (COX, 1958) as the loss function.

Even though this approach might help reduce the errors in nearly aligned images, it has some disadvantages:

- When the image is rotated, this method does not specify the upright vector. To overcome this problem we could use a second CNN to regress the pose, but this would increase the inference time;
- Even when the prediction of the model is correct, images with smaller rotations will contain error.
- The circularity problem is still present, slowing down the convergence of the upright regression model.

4.3 Direct Regression of the Upright Vector

Existing approaches for upright alignment, such as (JUNG et al., 2019), initially estimate the rotation of the input panorama and then apply the inverse rotation for gravity alignment. It is important to note that there are several ways to express a rotation, such as Euler angles, Angle-axis, Rotation Matrix, Unit Quaternions (HASHIM, 2019). Each parametrization has its peculiarities, with pros and cons. One characteristic that we are interested in is whether they suffer or not from the *circularity problem*

It is interesting to note that Euler angles were the chosen parametrization for the upright vector in (JUNG et al., 2019), and the authors report larger errors when the input image is already roughly aligned (i.e., the rotation angles are close to zero). To alleviate this problem, we model directly the upright camera vector instead of rotation parameters, mitigating the circularity problem. More precisely, the upright vector is expressed as $\boldsymbol{v}_0 = [x_0, y_0, z_0]^\top$ with $\|\boldsymbol{v}_0\| = 1$, which means that it presents 2 degrees of freedom (2-DoF). Note that such representation does not suffer from discontinuities.

As backbone, we used a DenseNet121 (HUANG et al., 2017) - same as used by Jung et al. (2019). To optimize the Mean Squared Error (MSE) loss, we used Adam optimizer with a learning rate set to 0.001 in the pre-trained layers and 0.01 in the randomly initialized layers. To preserve the fine-grained optimization in the late stages of the training process, we used a scheduler that reduces the learning rate in a factor of 0.1 after 10 epochs without improvements in the angular error. For this model, the size of the input images is 442×221, the batch size was 48 and the model was trained for 200 epochs.

4.4 Backbone with spherical convolutions

As mentioned in Section 3.1, there are several methods to improve feature extraction in spherical images, particularly dealing with the problem of non-uniform sampling. To evaluate if using such spherical adaptations indeed improve the results of upright adjustment, we decided to test one of these techniques. Among several existing possibilities, we decided to use SphereNet (COORS; CONDURACHE; GEIGER, 2018), which presents a spherical convolution and a spherical max pool layer. The main reasons for choosing this particular spherical adaptation strategy were the straight-forward layer-wise adaptation, the rotation *variance* of the convolution, and the existence of publicly available code at <https://github.com/ChiWeiHsiao/SphereNet-pytorch>.

In this new model, we use an adapted DenseNet121 (HUANG et al., 2017) as the backbone, as in the planar version of the network described before. Originally, DenseNet contains convolutions with kernel sizes 1, 3, and 7, but the adopted SphereNet implementation only allows convolutions with kernel size 3. Since convolutions with size 1 are not affected by the distortions in image, we kept them unchanged. The single planar convolution with kernel size 7 and all the other planar convolutions with kernel size 3 were replaced by spherical convolutions with a kernel size of 3. Also, all the planar max-pooling layers were replaced by a spherical max-pool, also available in SphereNet.

The spherical convolutions of SphereNet are slower and consume much more memory than conventional convolution. To compensate for this drawback and keep training time and memory usage similar to the values obtained in Section 4.3, we reduced the size of the input images from 442×221 to 320×160. It is also important to note that such a customized backbone does not present pre-trained weights. Hence, all layers were randomly initialized, and we extended the training phase from 200 to 300 epochs, applying the same learning rate (0.01) to all layers. The learning rate scheduler was also changed to reduce the learning rate in a factor of 0.1 after 40 epochs instead of 10. Aiming to keep the experiments as similar as possible, the loss function, optimizer, and batch size were kept the same as the planar network described in Section 4.3.

5 RESULTS

In this chapter, we discuss the experiments and results of the methods described in Chapter 4. We present both qualitative and quantitative results and compare our approach with state-of-the-art upright adjustment methods. All the metrics presented in this chapter were calculated using the entire test set described in Section 4.1, and all images used for illustration are part of this same set.

To evaluate the classification task we adopt the accuracy metric given by

$$acc = \frac{n_{correct}}{n_{total}}, \tag{5.1}$$

where $n_{correct}$ is the number of samples for which the predicted class is the same as the ground truth and n_{total} is the total of samples tested.

To compare the performance of different regressors, we adopt the angular error given by

$$e(\boldsymbol{v}_{gt}, \boldsymbol{v}_{out}) = \cos^{-1} \langle \boldsymbol{v}_{gt}, \boldsymbol{v}_{out} \rangle, \tag{5.2}$$

where $\boldsymbol{v}_{gt}$ and $\boldsymbol{v}_{out}$ represent the ground truth and estimated upward unit vectors, and $< \cdot, \cdot >$ denotes the inner product.

Besides presenting numerical values, we also categorize the results into *error bins*. For this purpose, we follow the thresholds proposed by Jung et al. (2019) after a user study, in which upright-corrected images with an angular error $e \leq 5°$ were considered very satisfactory, and images with $e \leq 12°$ are satisfactory, noting that the satisfactory bin includes images considered very satisfactory.

5.1 Classifying aligned images

In Section 4.2, we propose to classify images into three categories: Aligned ($-5° \leq \beta, \gamma \leq 5°$), Upside-down ($-5° \leq \beta \leq 5°$ and $175° \leq \gamma \leq 185°$) and rotated (all other angular configurations) using a ResNet50 (HE et al., 2016). Given a balanced dataset as described in Section 4.2, our classification method achieved an accuracy of $acc = 0.9886$. The confusion matrix in Table 5.1 indicates that the most common error is classifying aligned or upside-down images as rotated images.

As far as we know, there is no existing method that performs this kind of classification, which compromises the comparison with SOTA. To establish some sort of com-

		Predicted		
		Aligned	Rotated	Upside-down
Actual	Aligned	8489	89	15
	Rotated	28	8320	34
	Upside-down	9	117	8477

Table 5.1 – Classification Confusion Matrix

parison, Deep360Up (JUNG et al., 2019) reports a median error of 5.2° in upside-down images, which means that 50% of the upside-down images present an error $e \geq 5.2°$. Recall that Deep360Up takes advantage of high-resolution images for coupled rotation and resize operations – which reduces artifacts.

Table 5.2 presents a few examples of images that were incorrectly classified. We can see that most of the errors relate to images that are near the boundary between aligned/rotated or rotated/upside-down. A few other errors arise from more complex images that sometimes make the CNN confuse aligned and upside-down images (such as the last image).

5.2 Direct regression of the upright vector

This section evaluates the quality of the regressed upright vector using the approach described in Section 4.3. It is important to recall that existing methods (JUNG et al., 2019; JUNG; CHO; KWON, 2020) that use regression report high errors in nearly aligned and upside-down images, and one of our goals was to mitigate this problem. Our average angular error was $4.28°$ with a standard deviation of $10.78°$. The small error and high standard deviation might indicate that there are several outliers with higher errors, as shown in Figure 5.1. In fact, the median angular error, a metric that is less influenced by outliers, is $2.58°$.

Figure 5.2 shows the percentiles of the error distribution versus the angular error. The sharp increase of the plot close to the origin indicates that most of the error distribution presents a small angular error. 82.77% of the samples present an angular error smaller than $5°$, which is considered by people as a very satisfactory orientation, and 97.03% are smaller than $12°$, which is considered a satisfactory orientation according to the discus-

Figure 5.1 – Boxplot showing the distribution of the angular errors in degrees.

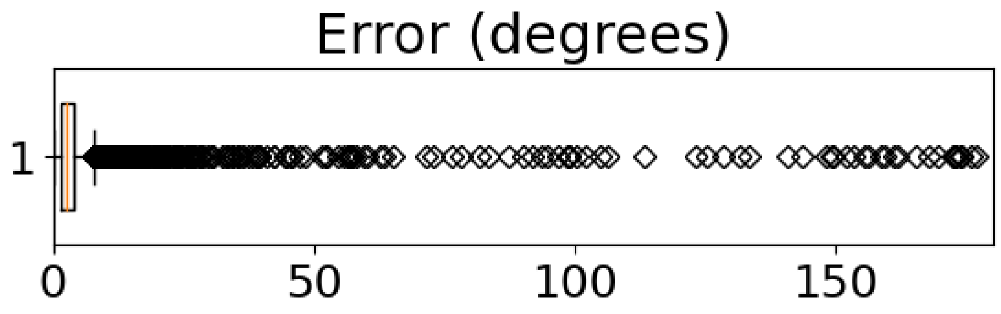


Figure 5.2 – Percentiles versus the angular error in degrees.

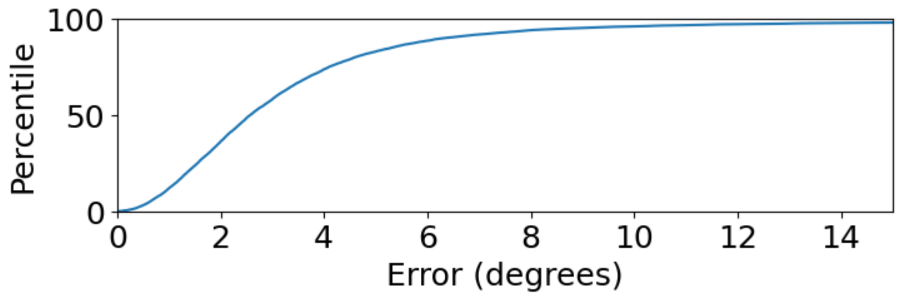


Figure 5.3 – Median error for each pitch and roll angles.

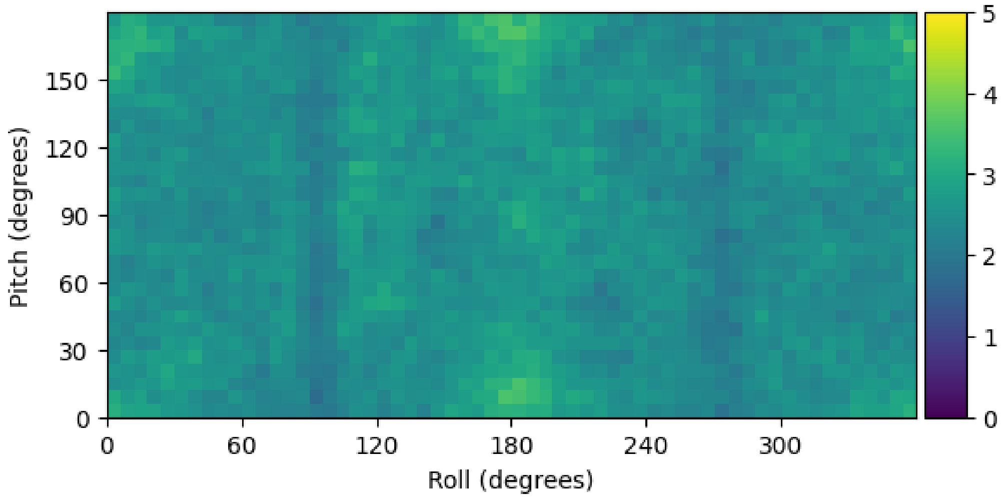


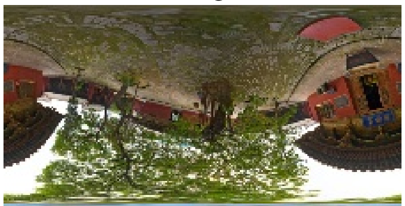
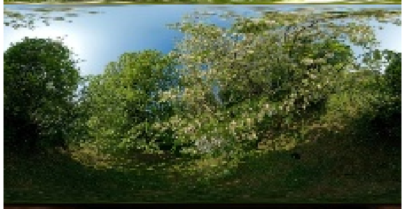
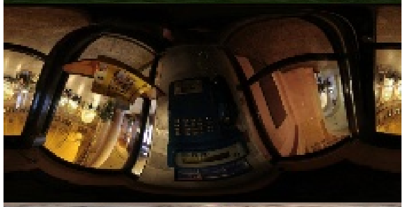
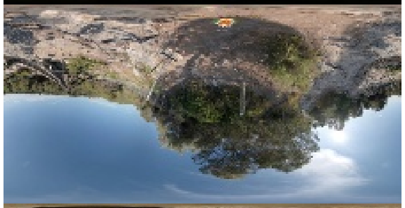
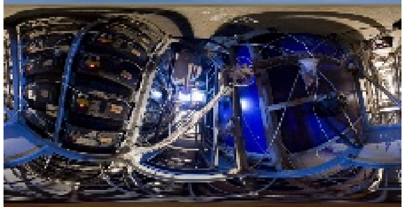
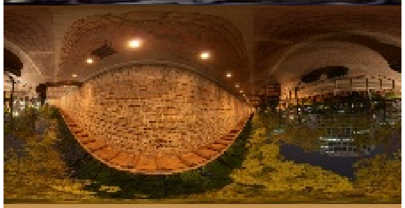
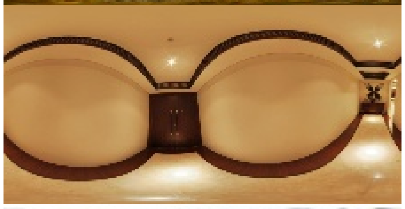
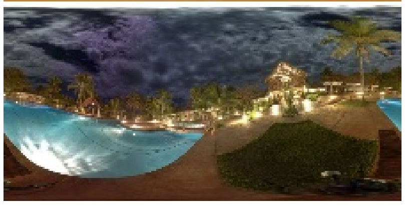
Image	Ground-truth	Predicted
	Rotated	Upside-down
	Rotated	Aligned
	Rotated	Aligned
	Rotated	Upside-down
	Upside-down	Rotated
	Upside-down	Aligned
	Aligned	Rotated
	Aligned	Upside-down

Table 5.2 – Wrong Classification Examples

sion in the beginning of this chapter. For the sake of comparison, Deep360Up presents fractions of 90.27% and 96.36% related to very satisfactory and satisfactory errors, respectively, as reported in Jung et al. (2019). Although our percentage of “high-quality” estimates was lower compared to Deep360Up, we were able to produce a higher percentage of “satisfactory” estimates which indicates a more homogeneous error distribution as a function of the input rotation angle. We also computed the median error for angular bins (varying the pitch and roll angles over all possible values), as shown in Figure 5.3. We can observe that our error estimates are indeed roughly homogeneous for all rotation angles. The maximum median error was $3.64°$, compared to $5.2°$ in Jung et al. (2019).

In Figure 5.4, we show a few example images (possibly rotated), the gravity-alignment obtained with our estimated upright vector, along with the corresponding angular error assuming the image in the SUN360 dataset (J. Xiao et al., 2012) was originally aligned. As we can see, most of the images are nearly aligned after our gravity-alignment process. The last image shows a common mistake in complex scenes, in which an “inverted” upright vector is regressed so that the “aligned” image is roughly upside-down.

5.2.1 Improving Monocular Depth Estimation

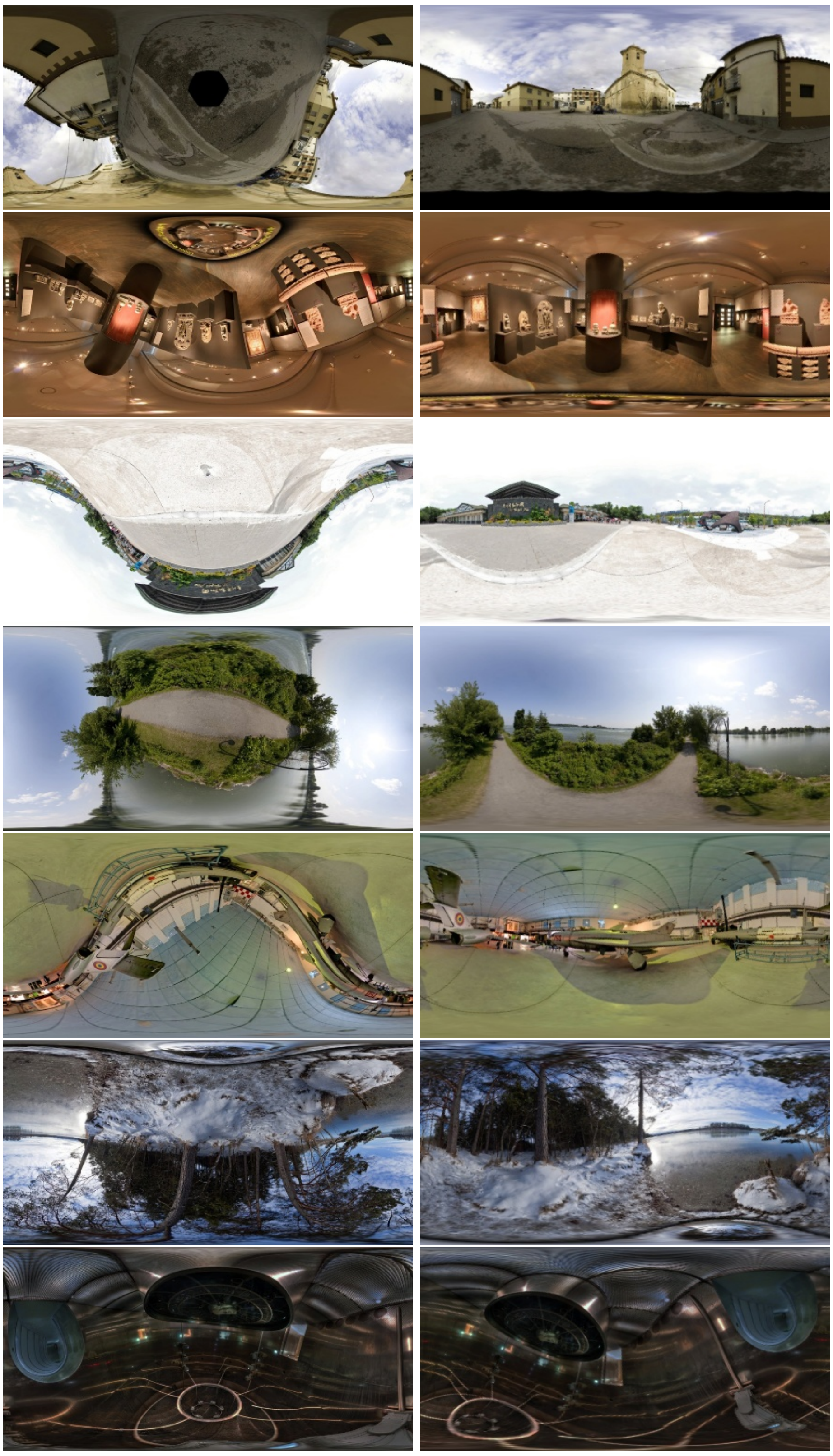
Several computer vision methods are susceptible to non-aligned images, particularly when dealing with panoramas (ZIOULIS et al., 2018; WANG et al., 2020; COORS; CONDURACHE; GEIGER, 2018). To evaluate the applicability of our method as a pre-processing step, we evaluate how it affects a single-image depth estimation model for panoramas. Here, for proof-of-concept, we select OmniDepth (ZIOULIS et al., 2018) since it has publicly available source-code and trained weights.

Let $R(I, \boldsymbol{v})$ denote the rotation of an equirectangular image I according to a unit upright vector $\boldsymbol{v}$. Also, let $f_u(I)$ denote the upright correction module, which outputs an estimate for $\boldsymbol{v}$, and $f_d(I)$ denote the depth estimation module (such as Zioulis et al. (2018)), which produces a pixel-wise depth estimate for the input image I. The complete pipeline consists of obtaining the upright vector for the input image, gravity-aligning it by applying a suitable rotation, estimating the depth, and then applying the inverse rotation to align the depth map with the original input image. The depth map D aligned with the input image I can be written as

$$D = R\left(f_d\left(R^{-1}\left(I, f_u(I)\right)\right), f_u(I)\right), \tag{5.3}$$

where I is the input color panorama. For some applications such as AR and VR, it might

Figure 5.4 – Example of inputs (left) rectified by the model (right). Errors are, from top to bottom, 0.8°, 1.8°, 2.18°, 3.26°,6.53°, 10.7°, 158.74°



be desirable to generate a 3D model of the captured scene aligned with the horizon. In that case, it is possible to use $R^{-1}(I, f_u(I))$ and $f_d(R^{-1}(I, f_u(I)))$ in the 3D modeling step, so that we obtain a gravity-aligned panorama with the corresponding depth map.

To evaluate our pipeline, we used images from the 3D60 dataset (ZIOULIS et al., 2018) assuming all images are roughly aligned and generated $10,000$ panoramas with random synthetic rotations. As proposed in Zioulis et al. (2018), we perform median depth scaling of the predicted depth map D_{pred} by multiplying it by a scaling factor s given by

$$s = \frac{\text{median}(D_{gt})}{\text{median}(D_{pred})}, \tag{5.4}$$

where D_{gt} is the annotated depth map. Using the scaled predicted depth map we can compute common evaluation metrics (EIGEN; PUHRSCH; FERGUS, 2014; da Silveira; DAL'AQUA; JUNG, 2018; ZIOULIS et al., 2018; YANG; LIU; KANG, 2018; EDER; MOULON; GUAN, 2019; TATENO; NAVAB; TOMBARI, 2018) (ignoring missing values in D_{gt}). The measurements are presented in Table 5.3 along with original OmniDepth results in the original, non-rotated dataset. We can observe a considerable improvement in all error metrics when correcting the image orientation. Even though we improve the results, our error is still higher than in the original dataset. Of course estimating depth in any image is a harder problem than estimating in any aligned input image. But besides this, we believe that the errors estimating the upright vector are one of the factors that cause this gap. Also, the artifacts created by the rotation of the image can also affect the result.

Framework	Orientation	Abs Rel	Sq Rel	RMS	RMSlog	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
OmniDepth	rotated	0.1797	0.1236	0.5972	0.2511	0.7126	0.9285	0.9783
OmniDepth + Upright	rotated	0.1156	0.0547	0.3735	0.1701	0.8677	0.9741	0.9928
OmniDepth	original	0.0641	0.0197	0.2297	0.0993	0.9663	0.9951	0.9984

Table 5.3 – Quantitative results for the baseline depth estimation method under rotation scenarios. Results on the original dataset are also provided.

To better understand how rotation affects in depth estimation, Figure 5.5 presents the depth inference error as a function of the rotation of the input panorama. We can see that our method improves the results and makes the method more robust to rotated images. The chosen baseline depth estimator (ZIOULIS et al., 2018) produces smaller errors when the input panoramas are roughly aligned, but the quality degrades as the rotation angle increases.

Figure 5.6 presents qualitative results of the depth maps produced by OmniDepth (ZIOULIS et al., 2018) in the rotated input images with and without the proposed upright adjustment

Figure 5.5 – Relative depth error varying the rotation angles (roll and pitch) without rotation correction (left) and with (right).

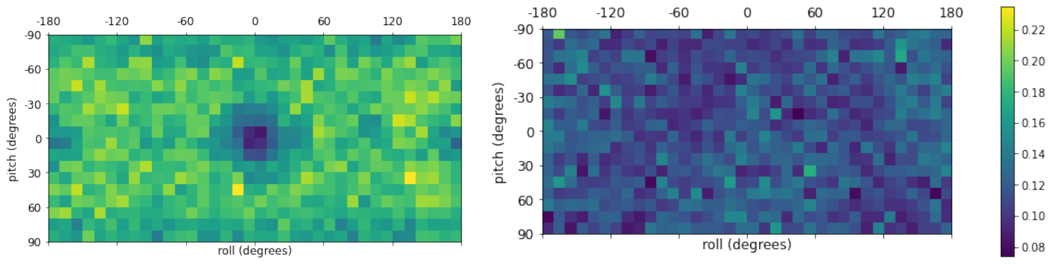
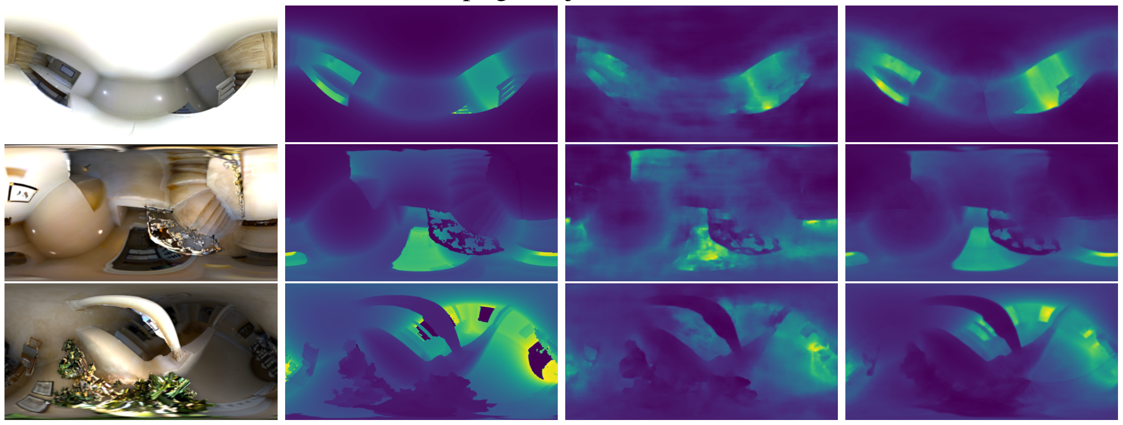


Figure 5.6 – Qualitative comparison of the Omnidepth method with and without the Upright module. The images are, from left to right input, ground-truth, Ominidepth and Omnidepth with upright adjustment.



module. We can observe that the direct application of OmniDepth to rotated images generates artifacts, whereas the pre-processed images present results similar to the groundtruth.

5.3 Backbone with spherical convolutions

Our model trained with spherical convolutions and poolings achieved an average angular error of $9.87°$ with a standard deviation of 20.76. This is more than twice the mean error achieved by the model with conventional convolutions and the standard deviation is twice the standard deviation achieved by the planar model. The median error was $5.05°$

Figure 5.7 – Percentiles versus the angular error in degrees using the spherical backbone.

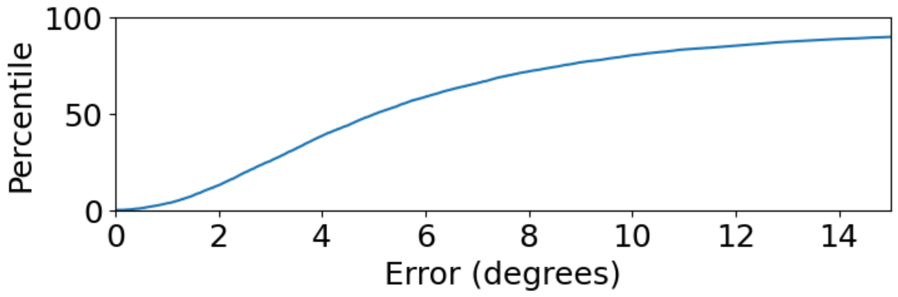


Figure 5.8 – Boxplot showing the distribution of the angular errors in degrees using the spherical backbone.

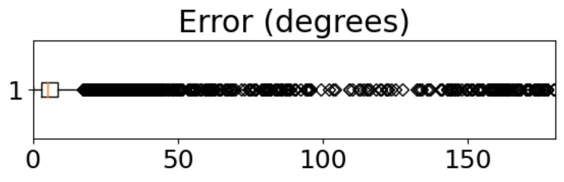
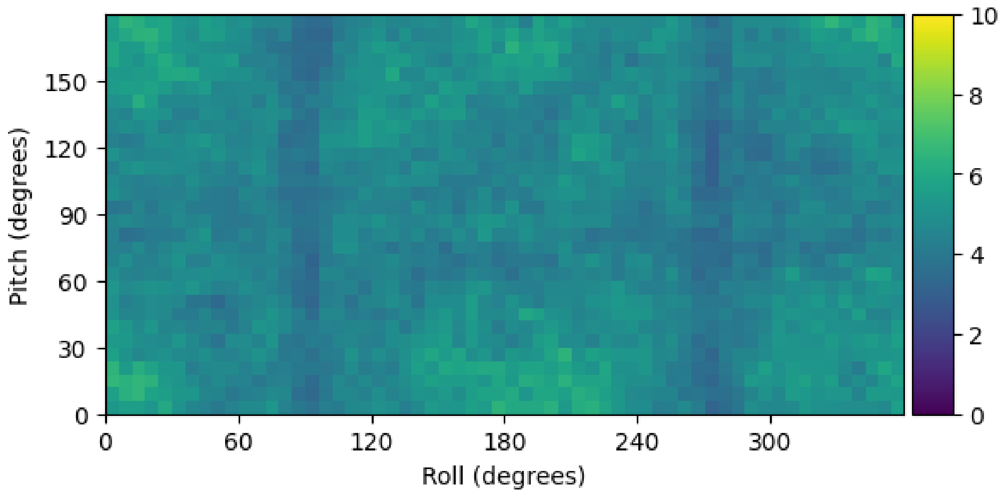


Figure 5.9 – Median error for each pitch and roll angles in the spherical model.

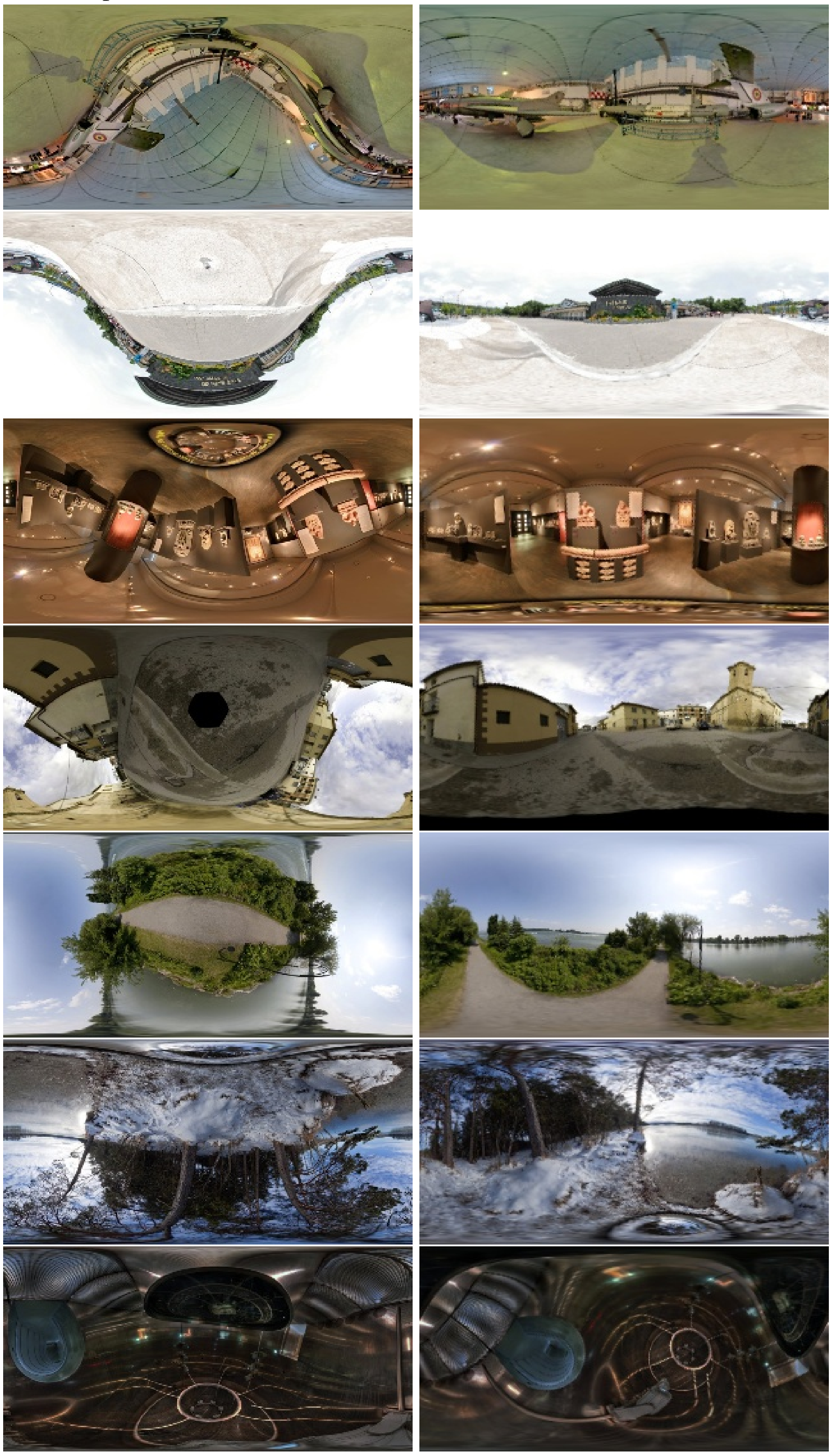


Following the classification discussed in the beginning of this chapter, 49.48% of the images were considered Very Satisfactory and 85.17% were considered Satisfactory. Figure 5.7 shows a slower increase of the percentiles compared to those presented in Section 5.2. In Figure 5.8, the boxplot shows that the error distribution is also similar, containing a lot of high error outliers.

Similarly to conventional convolutions, our error is well distributed between all the rotations, as show in Figure 5.9. Note that the scale is different that in Figure 5.3. The maximum median value is $6.60°$, higher than both our planar model and Jung et al. (2019).

Figure 5.10 shows some images before and after correction by the spherical model. Despite having higher errors in average, the spherical model had smaller errors than the planar in both first and last example images shown in Figure 5.10. Both of these images are more complex scenes with several objects and different textures. This might indicate that the model with spherical convolution performs better in those cases.

Figure 5.10 – Example of inputs (left) rectified by the spherical model (right). Errors are, from top to bottom, 1.78°, 2.66°, 2.76°, 4.89°, 4.91°, 15.23°, 112.16°



6 CONCLUSION

This work presented three deep models for the gravity-alignment problem in panoramas. One model focuses on classification, and categorizes an input panorama into thre classes: aligned, upside-down, and rotated. The other two models aim at regressing the upright vector (which can be used to gravity-align the images). Even though we have median errors slightly higher than state of the art, our method with planar convolutions is one of the fastest for performing upright adjustment. Also, our regression models are the only ones that deal with the circularity problem.

Even though the classification model does not specify where the upright vector points to for rotated images, it allows dealing separately with the cases with higher errors in Jung et al. (2019). It might be a good solution when most of the images are already aligned (only one model needed) but when the image is rotated another module is necessary to perform the adjustment. It is not useful in cases that prediction time is relevant, because two CNNs are necessary to perform the correction.

The regression method with a DenseNet as backbone is probably the best option presented in this work for most of the cases. Its robustness dealing with the entire range of rotation allows handling virtually any input with small errors. It is also recommended when efficiency is important because it uses only a single model and uses conventional 2D convolutions, which are faster than spherical convolutions.

We have also shown how our method can be used to add robustness to other computer vision tasks, using Omnidepth (ZIOULIS et al., 2018), a single-image depth estimation method, as an example. For this application scenario, high errors in nearly aligned images (present in SOTA regression methods) represent an issue, because the use of the upright adjustment module would make the resulting depth map worse for these cases. Our method, however, deals with the circularity problem making Omnidepth obtain even better depth estimations.

Finally, we presented a CNN using a DenseNet-like backbone but using spherical convolutions. In our tests, the spherical model achieved higher angular errors than the planar version. Therefore, our implementation is not a good match when there are time or memory limitations since spherical convolution is a computationally expensive operation. Without these limitations, it would be possible to use larger images that could present more information to the CNN and probably improve the results. The model has potential to achieve the same error as the planar model for simpler images, whilst being more robust in complex scenes. Ideally, there should also be a way to apply transfer learning, which usually reduces training time and improves the results of the model.

As future work, a spherical backbone could be trained to perform another task using a dataset with more images and after that perform transfer learning. Another important improvement would be the creation of a dataset that is specific to the task of upright adjustment. Ideally, this dataset should contain not only aligned images (which would remove the issue of the artifacts generated by the rotation) and the annotation should be more precise than assuming all images are originally aligned. Alternatively, it is possible to manually label the rotation of all images of an existing dataset. Finally, an interest test would be evaluate how the method behaves on spherical videos. Perhaps, additional steps would be necessary to ensure stability in the video.

REFERENCES

ABREU, A. D.; OZCINAR, C.; SMOLIC, A. Look around you: Saliency maps for omnidirectional images in vr applications. In: IEEE. **2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX)**. [S.l.], 2017. p. 1–6.

BAZIN, J.-C. et al. Motion estimation by decoupling rotation and translation in catadioptric vision. **Computer Vision and Image Understanding**, Elsevier, v. 114, n. 2, p. 254–273, 2010.

BAZIN, J.-C. et al. Rotation estimation and vanishing point extraction by omnidirectional vision in urban environment. **The International Journal of Robotics Research**, SAGE Publications Sage UK: London, England, v. 31, n. 1, p. 63–81, 2012.

BAZIN, J.-C. et al. Uav attitude estimation by vanishing points in catadioptric images. In: IEEE. **2008 IEEE International Conference on Robotics and Automation**. [S.l.], 2008. p. 2743–2749.

BAZIN, J.-C.; POLLEFEYS, M. 3-line ransac for orthogonal vanishing point detection. In: IEEE. **2012 IEEE/RSJ International Conference on Intelligent Robots and Systems**. [S.l.], 2012. p. 4282–4287.

BAZIN, J.-C.; SEO, Y.; POLLEFEYS, M. Globally optimal consensus set maximization through rotation search. In: SPRINGER. **Asian Conference on Computer Vision**. [S.l.], 2012. p. 539–551.

CHOLLET, F. et al. **Deep learning with Python**. [S.l.]: Manning New York, 2018.

COHEN, T. S. et al. Spherical cnns. **arXiv preprint arXiv:1801.10130**, 2018.

COORS, B.; CONDURACHE, A. P.; GEIGER, A. Spherenet: Learning spherical representations for detection and classification in omnidirectional images. In: **Proceedings of the European Conference on Computer Vision (ECCV)**. [S.l.: s.n.], 2018. p. 518–533.

COX, D. R. The regression analysis of binary sequences. **Journal of the Royal Statistical Society: Series B (Methodological)**, Wiley Online Library, v. 20, n. 2, p. 215–232, 1958.

da Silveira, T. L.; DAL'AQUA, L. P.; JUNG, C. R. Indoor Depth Estimation from Single Spherical Images. In: **IEEE ICIP**. [s.n.], 2018. p. 2935–2939. ISBN 978-1-4799-7061-2. Available from Internet: <https://ieeexplore.ieee.org/document/8451769/>.

DAVIDSON, B.; ALVI, M. S.; HENRIQUES, J. F. 360°camera alignment via segmentation. In: VEDALDI, A. et al. (Ed.). **Computer Vision – ECCV 2020**. Cham: Springer International Publishing, 2020. p. 579–595. ISBN 978-3-030-58604-1.

DEMONCEAUX, C.; VASSEUR, P.; PÉGARD, C. Omnidirectional vision on uav for attitude computation. In: IEEE. **Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.** [S.l.], 2006. p. 2842–2847.

DEMONCEAUX, C.; VASSEUR, P.; PÉGARD, C. Robust attitude estimation with catadioptric vision. In: IEEE. **2006 IEEE/RSJ International Conference on Intelligent Robots and Systems**. [S.l.], 2006. p. 3448–3453.

DENG, J. et al. Imagenet: A large-scale hierarchical image database. In: IEEE. **2009 IEEE conference on computer vision and pattern recognition**. [S.l.], 2009. p. 248–255.

EDER, M.; MOULON, R.; GUAN, L. Pano Popups: Indoor 3D Reconstruction with a Plane-Aware Network. In: **3DV**. IEEE, 2019. p. 76–84. ISBN 978-1-7281-3131-3. Available from Internet: <https://ieeexplore.ieee.org/document/8885718/>.

EDER, M. et al. Tangent Images for Mitigating Spherical Distortion. In: **IEEE/CVF CVPR**. [s.n.], 2019. p. 12426–12434. Available from Internet: <http://arxiv.org/abs/1912.09390>.

EIGEN, D.; PUHRSCH, C.; FERGUS, R. Depth map prediction from a single image using a multi-scale deep network. **arXiv preprint arXiv:1406.2283**, 2014.

FISCHER, P.; DOSOVITSKIY, A.; BROX, T. Image orientation estimation with convolutional networks. In: SPRINGER. **German Conference on Pattern Recognition**. [S.l.], 2015. p. 368–378.

GALLAGHER, A. C. Using vanishing points to correct camera rotation in images. In: IEEE. **The 2nd Canadian Conference on Computer and Robot Vision (CRV'05)**. [S.l.], 2005. p. 460–467.

HASHIM, H. A. Special orthogonal group so (3), euler angles, angle-axis, rodriguez vector and unit-quaternion: Overview, mapping and challenges. **arXiv preprint arXiv:1909.06669**, 2019.

HE, K. et al. Deep residual learning for image recognition. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 770–778.

HOLD-GEOFFROY, Y. et al. A perceptual measure for deep single image camera calibration. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2018. p. 2354–2363.

HUANG, G. et al. Densely connected convolutional networks. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2017. p. 4700–4708.

J. Xiao et al. Recognizing scene viewpoint using panoramic place representation. In: **IEEE CVPR**. [s.n.], 2012. p. 2695–2702. ISBN 978-1-4673-1228-8. Available from Internet: <http://ieeexplore.ieee.org/document/6247991/>.

JEON, J.; JUNG, J.; LEE, S. Deep upright adjustment of 360 panoramas using multiple roll estimations. In: SPRINGER. **Asian Conference on Computer Vision**. [S.l.], 2018. p. 199–214.

JOO, K. et al. Globally optimal inlier set maximization for atlanta frame estimation. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2018. p. 5726–5734.

JOSHI, U.; GUERZHOY, M. Automatic photo orientation detection with convolutional neural networks. In: IEEE. **2017 14th Conference on Computer and Robot Vision (CRV)**. [S.l.], 2017. p. 103–108.

JUNG, J. et al. Robust upright adjustment of 360 spherical panoramas. **The Visual Computer**, Springer, v. 33, n. 6, p. 737–747, 2017.

JUNG, R.; CHO, S.; KWON, J. Upright adjustment with graph convolutional networks. In: IEEE. **2020 IEEE International Conference on Image Processing (ICIP)**. [S.l.], 2020. p. 1058–1062.

JUNG, R. et al. Deep360up: A deep learning-based approach for automatic vr image upright adjustment. In: IEEE. **2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)**. [S.l.], 2019. p. 1–8.

KHASANOVA, R.; FROSSARD, P. Graph-based classification of omnidirectional images. In: **Proceedings of the IEEE International Conference on Computer Vision Workshops**. [S.l.: s.n.], 2017. p. 869–878.

KIEFER, J.; WOLFOWITZ, J. et al. Stochastic estimation of the maximum of a regression function. **The Annals of Mathematical Statistics**, Institute of Mathematical Statistics, v. 23, n. 3, p. 462–466, 1952.

LEE, H. et al. Automatic upright adjustment of photographs with robust camera calibration. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 36, n. 5, p. 833–844, 2013.

OLMSCHENK, G.; TANG, H.; ZHU, Z. Pitch and roll camera orientation from a single 2d image using convolutional neural networks. In: IEEE. **2017 14th Conference on Computer and Robot Vision (CRV)**. [S.l.], 2017. p. 261–268.

REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. **arXiv preprint arXiv:1804.02767**, 2018.

REN, S. et al. Faster r-cnn: Towards real-time object detection with region proposal networks. **arXiv preprint arXiv:1506.01497**, 2015.

ROBBINS, H.; MONRO, S. A stochastic approximation method. **The annals of mathematical statistics**, JSTOR, p. 400–407, 1951.

RONNEBERGER, O.; FISCHER, P.; BROX, T. U-net: Convolutional networks for biomedical image segmentation. In: SPRINGER. **International Conference on Medical image computing and computer-assisted intervention**. [S.l.], 2015. p. 234–241.

RUDER, M.; DOSOVITSKIY, A.; BROX, T. Artistic style transfer for videos and spherical images. **International Journal of Computer Vision**, Springer, v. 126, n. 11, p. 1199–1219, 2018.

SHAN, Y.; LI, S. Discrete spherical image representation for cnn-based inclination estimation. **IEEE Access**, IEEE, v. 8, p. 2008–2022, 2019.

SILVEIRA, T. L. d.; DALAQUA, L. P.; JUNG, C. R. Indoor Depth Estimation from Single Spherical Images. In: **2018 25th IEEE International Conference on Image Processing (ICIP)**. IEEE, 2018. p. 2935–2939. ISBN 978-1-4799-7061-2. Available from Internet: <https://ieeexplore.ieee.org/document/8451769/>.

SILVEIRA, T. L. da; JUNG, C. R. Dense 3d indoor scene reconstruction from spherical images. In: SBC. **Anais Estendidos do XXXIII Conference on Graphics, Patterns and Images**. [S.l.], 2020. p. 8–14.

SILVEIRA, T. L. T. da; JUNG, C. R. Dense 3D Scene Reconstruction from Multiple Spherical Images for 3-DoF+ VR Applications. In: **IEEE VR**. [s.n.], 2019. p. 9–18. ISBN 978-1-7281-1377-7. Available from Internet: <https://ieeexplore.ieee.org/document/8798281/>.

STANLEY, R. P. The fibonacci lattice. **Fibonacci Quart**, v. 13, n. 3, p. 215–232, 1975.

SU, Y.-C.; GRAUMAN, K. Learning spherical convolution for fast features from 360° imagery. In: **NIPS**. [S.l.: s.n.], 2017. v. 2, n. 3, p. 5.

SU, Y.-C.; GRAUMAN, K. Kernel transformer networks for compact spherical convolution. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2019. p. 9442–9451.

TATENO, K.; NAVAB, N.; TOMBARI, F. Distortion-Aware Convolutional Filters for Dense Prediction in Panoramic Images. In: **ECCV**. [S.l.: s.n.], 2018. p. 732–750.

WAN, L.; WONG, T.-T.; LEUNG, C.-S. Isocube: Exploiting the cubemap hardware. **IEEE Transactions on Visualization and Computer Graphics**, IEEE, v. 13, n. 4, p. 720–731, 2007.

WANG, F. et al. Bifuse: Monocular 360 depth estimation via bi-projection fusion. In: **IEEE/CVF CVPR**. [S.l.: s.n.], 2020. p. 459–468.

YANG, Y.; LIU, R.; KANG, S. B. Automatic 3D Indoor Scene Modeling from Single Panorama. In: **IEEE/CVF CVPR**. [S.l.: s.n.], 2018. p. 5430.

YU, F.; KOLTUN, V. Multi-scale context aggregation by dilated convolutions. **arXiv preprint arXiv:1511.07122**, 2015.

ZHANG, L. et al. Vanishing point estimation and line classification in a manhattan world with a unifying camera model. **International Journal of Computer Vision**, Springer, v. 117, n. 2, p. 111–130, 2016.

ZIOULIS, N. et al. Omnidepth: Dense depth estimation for indoors spherical panoramas. In: **Proceedings of the European Conference on Computer Vision (ECCV)**. [S.l.: s.n.], 2018. p. 448–465.

ZOU, C. et al. LayoutNet: Reconstructing the 3D Room Layout from a Single RGB Image. In: **2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition**. IEEE, 2018. p. 2051–2059. ISBN 978-1-5386-6420-9. Available from Internet: <http://arxiv.org/abs/1803.08999https://ieeexplore.ieee.org/document/8578317/>.