Viewpoint estimation in medical imaging

Mahouclo Anicet Hounkanrin

Supervisor: A/Prof. Fred Nicolls
Co-supervisor: Dr. Paul Amayo

A thesis submitted in fulfillment of the requirements for the degree of
Doctor of Philosophy

Department of Electrical Engineering
University of Cape Town

November 2023

Declaration

I, Mahouclo Anicet Hounkanrin, hereby declare that the work on which this thesis is based is my original work (except where acknowledgments indicate otherwise) and that neither the whole work nor any part of it has been, is being, or is to be submitted for another degree in this or any other university. I authorize the University of Cape Town to reproduce for the purpose of research either the whole or any portion of the contents in any manner whatsoever.

Signature:
M.A. Hounkanrin
November 2023

Abstract

In medical imaging, the appearance of a certain body part on a radiograph depends not only on the position but also on the orientation of the X-ray imaging system with respect to the patient. Given a 2D image of a 3D scene, the problem of viewpoint estimation aims to determine the position and the orientation of the imaging sensor that resulted in that view. We investigate methods to solve the viewpoint estimation problem for medical images, notably the determination of orientation parameters. Machine learning models, particularly convolutional neural networks (CNNs), are developed to predict a human subject's orientation in a radiograph. Since deep learning models require data for training, we first generate a dataset of digitally reconstructed radiographs (DRRs) from a set of computed tomography (CT) scans using Fourier volume rendering (FVR). The dataset of DRRs is then used to train CNN models for viewpoint regression and classification. A label-softening strategy is used to improve the performance of the classification models. Meanwhile, a geometric structure-aware cost function is used to account for the geometric continuity of the viewpoint space. Several 3D rotation methods such as Euler angle, axis-angle, and quaternions are investigated for viewpoint representation. The results demonstrate that viewpoint estimation in medical imaging can be effectively solved using CNN-based classification and regression models. The geometric structure-aware cost function proves to be essential to the success of classification models for viewpoint estimation. The regression-based models, on the order hand, appear to be sensitive to the type of parametrization used to represent the viewpoints. In particular, the unit quaternion representation of 3D rotations proves to be more effective than other representations for viewpoint regression with CNN models. Moreover, we extend the proposed method to perform viewpoint estimation for natural images. The performance on the PASCAL3D+ dataset indicates that the application of the methods presented is not restricted to medical imaging.

Acknowledgments

I would like to express my profound gratitude to my supervisors, Fred Nicolls and Paul Amayo, for their support and guidance throughout the course of this project. I would also like to thank my family and friends for their constant support and encouragement.

I acknowledge the financial assistance received from the University of Cape Town (UCT), and the German Academic Exchange Service (DAAD) via the African Institute for Mathematical Sciences (AIMS) South Africa.

In memory of Raphaël Cossi Koukpanou

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Two-dimensional images depict projections of objects in a scene. As such, they lack depth information that is necessary to gain a full understanding of a 3D scene. Although the human visual system is able to fill the gaps and make sense of 2D images, it is much harder for a computer to "understand" projected views from a 3D scene. An image detection or classification model could easily be fooled when seeing images from different viewpoints since nearby viewpoints could result in significantly different projections that are difficult for a computer system to understand. In computer vision, viewpoint estimation aims to help machine vision systems gain a better understanding of 3D scenes by learning to recognize the views of objects in an image. The most effective viewpoint estimation methods are based on deep learning [16, 25, 38, 49, 56, 79]. Although the effectiveness of deep learning methods is proven and widely recognized in the computer vision community, these methods rely on massive amounts of training data, which may not be readily available. In particular, supervised learning-based viewpoint estimation methods require not only large amounts of data but also data correctly labeled with viewpoint information. Since the number of datasets for viewpoint learning is quite limited, the number of works dealing with viewpoint estimation is very limited compared to other areas of computer vision.

The scarcity of training data for viewpoint estimation is even more severe in medical imaging. There is currently no publicly available dataset specifically dedicated to viewpoint estimation in medical imaging, leading to a lack of deep learning work addressing the problem despite the existence of potential applications in the field.

Computer-assisted surgery (CAS) relies on the creation of 3D patient models for pre-operative planning and intra-operative guidance. The models used for CAS are generally based on computed tomography (CT) and magnetic resonance imaging

(MRI) scans that give a 3D representation of the patient's body. These 3D models could be projected onto different viewpoints to provide a good visualization of the internal structure of the body. Thus viewpoint estimation could be used to identify specific views from projections of 3D models and help with better diagnosis.

1.1 Motivations

The motivations of this project are threefold: the development of novel methods for viewpoint estimation using CNNs, the application of viewpoint estimation methods to medical images, and the application of viewpoint estimation for computer-aided diagnosis (CAD).

1. Development of novel methods for viewpoint estimation using CNNs

   Estimating viewpoints from 2D images is a challenging task since projected 2D images lose critical information about the 3D scene. The most effective methods for this task in the literature are based on CNN models using supervised learning [16, 25, 38, 49, 56, 79]. These methods are either classification or regression-based. For the regression-based methods, the models are trained to predict the viewpoint of an object in a continuous viewpoint space, whereas classification-based methods require that the viewpoint space be discretized into partitions with each partition representing a class. The CNN model is then trained to predict the class of each test image. Although the classification-based methods seem intuitively less convenient for solving the viewpoint estimation problem, they generally perform better than the regression-based approaches. Nevertheless, the performance of the existing methods for viewpoint estimation in natural images seems to have plateaued over the last few years. Here we explore new methods for the viewpoint estimation task using CNNs with various representations of the viewpoint space.

2. Application of viewpoint estimation methods to medical images

   Most viewpoint estimation methods developed to date are applied to natural images. Due to the specificity of medical images, specific methods need to be developed to solve the problem of viewpoint estimation in the medical imaging domain. The lack of work on viewpoint estimation in medical imaging is certainly not due to the lack of applicability, but rather to the absence of suitable datasets that can be used to develop these methods. In medical imaging, several modalities such as CT and MRI are used to obtain a three-dimensional visualization of the human body. These imaging modalities help the radiologist or surgeon make better diagnoses and treatment planning. The

CT scans depict structural information of the hard tissues in the body whereas the MRI depicts soft tissues and functional information. Radiographs are essentially 2D projections of CT scans. Thus they are used to visualize hard tissues on a 2D plane. Taking projections from different viewpoints around the CT volume results in different 2D X-ray images depicting different parts of the body. Given a projected X-ray image, one could use viewpoint estimation to determine where, in a 3D space around the CT volume, the X-ray image was projected from. Thus viewpoint estimation could, for example, be used in this case for patient positioning in computer-assisted surgery.

3. Application of viewpoint estimation to computer-assisted diagnosis

   With the growing usage of imaging in the clinical routine, either for diagnosis or treatment planning, the size of medical image datasets increases every day. One can imagine that at some point in time, these datasets would need to be sorted in order to be useful for medical diagnosis. In this case, viewpoint estimation can be used to index the database so that the medical practitioner can have access to the most relevant images for their case.

In this project, we aim to fill this gap in the literature by generating suitable datasets and proposing effective methods for viewpoint estimation in medical imaging.

1.2 Problem statement

The problems investigated in this project are broken into five points:

- Development of an efficient method to generate a dataset of digitally reconstructed radiographs (DRRs) from a limited set of CT volumes.
- Development of an effective model for 1D viewpoint estimation.
- Accounting for the circular distribution of the viewpoint space in the classification model for 1D viewpoint estimation.
- Representation of viewpoints using quaternions on a 3-sphere for 3D viewpoint estimation.
- Application of the proposed methods to viewpoint estimation for natural images.

1.3 Overview of the methodology

In this work we focus on the development of deep learning models for viewpoint estimation in medical imaging. To assess the generalization capability of the proposed methods, we extend these methods to perform viewpoint estimation on natural images. Given the complexity of the viewpoint estimation problem, we divide the task into sub-tasks which are solved individually. Thus we propose an efficient method to generate a dataset with viewpoint annotation. The dataset generated is then used to train deep learning models for 1D viewpoint estimation. The one-dimensional case is a simplification of the general viewpoint estimation problem. The performance of the models proposed for the 1D viewpoint estimation task is further improved using a geometry-aware loss that deals with the continuity of the viewpoint space. The proposed methods are then extended to solve 3D viewpoint estimation. Several viewpoint parametrization methods were investigated. The unit quaternion representation of 3D rotations proves to be the most effective viewpoint representation method. We finally extend the method to viewpoint estimation in natural images.

In this section, we present the different methods proposed in this work.

1.3.1 Data generation

To develop learning-based viewpoint estimation methods, we need a dataset with accurate viewpoint annotation. Given the lack of such a dataset for viewpoint estimation of medical images, our first task is to generate a dataset of medical images with viewpoint annotation that can be used to train viewpoint estimation models. The data generation methods are presented in Chapter 3. We generate a dataset of digitally reconstructed radiographs (DRRs) from a set of computed tomography (CT) scans. For simplicity, we only use orthogonal projection methods to generate 2D DRRs from the CT scan volumes. Two DRR generation methods are used: additive projection and Fourier volume rendering.

Additive projection (AP) is a simple way to generate DRRs from a CT scan. We employ this method to generate the dataset used for one-dimensional viewpoint estimation. The dataset is generated by rotating the CT volume around the longitudinal axis of the human body at 1-degree increments, and projecting orthogonally the volume onto a plane. This projection is done by summing up all voxel values along a ray going through the CT volume and perpendicular to the projection plane. Although the AP method is easy to implement, it is computationally expensive since it requires the rotation of the entire 3D volume before the projection of a DRR. This makes it difficult to integrate the AP method into an online data generation pipeline since it would make the training process computationally inefficient. Consequently, it becomes difficult to perform certain types of data augmentation such as random out-of-plane rotation during training. To integrate the data generation step into the model training pipeline, we use an alternative DRR generation method: Fourier volume rendering.

Sometimes called frequency domain volume rendering, Fourier volume rendering (FVR) is a DRR generation method based on the Fourier projection-slice theorem. To generate DRRs using the FVR method, the Fourier transform of the CT scan volume is first computed. The DRR corresponding to each viewpoint is obtained by computing the inverse Fourier transform of the slice in the direction of that viewpoint. In contrast to the AP method, which requires the rotation of the whole 3D volume before the generation of each DRR, the Fourier volume rendering (FVR) method only requires the extraction of a 2D slice from the 3D volume. The time complexity of FVR is $O(N^2 \log N)$, compared to the $O(N^3)$ for the AP method. Since FVR requires interpolation in the frequency domain, we investigate the interpolation methods that can be used for this task. The spline interpolation method proves to be the most effective method. We use the spatial domain zero-padding technique to suppress the replication of the signal due to the sampling in the Frequency domain. Moreover, we compare the DRRs generated using FVR and AP. The results show that the two methods generate similar DRRs. The FVR method has the advantage that it is more computationally efficient than the AP method. Thus FVR can be used for online data generation during training.

1.3.2 1D viewpoint estimation

To simplify the problem, we first investigate one-dimensional viewpoint estimation in Chapter 4. Thus the problem of viewpoint estimation considered in this chapter consists in determining the rotation angle around the longitudinal axis of a human body. This simplification allows us to develop baseline methods, to which more elaborate methods are compared. We investigate regression and classification methods for viewpoint estimation.

Given the continuity of the viewpoint space, regression methods seem to be the most suitable to solve the problem. We propose a regression model based on a convolutional neural network (CNN). This model is based on a model pre-trained on ImageNet [42] which was initially developed for a classification task for natural images. Thus we use transfer learning to adapt the model for viewpoint estimation in medical imaging. The base model is an Inception-V3 [81] model pre-trained on ImageNet. The classification block of this pre-trained model (made of full-connected layers) is removed and replaced by a regression block made of fully connected layers.

The whole network is fine-tuned end-to-end by minimizing a loss based on the distance between the ground truth angle and the predicted angle. Two different metrics were used to evaluate the performance of the trained model on a test set. The first evaluation metric is the median error, which measures the distance (in degrees) between the ground truth angle and the predicted angle. The second evaluation metric is the accuracy at θ (Acc_θ), which measures the proportion of images for which the predicted viewpoint and the ground viewpoint are within a threshold of θ. The threshold Acc_θ is often taken to be $\theta = \frac{\pi}{6}$. The regression model for 1D viewpoint estimation has an accuracy $Acc_{\frac{\pi}{6}} = 30.83\%$ and a median error $MedErr = 45.38$. Thus, although the regression approach seems to be a natural fit for the viewpoint estimation problem, the results show that they do not perform well at the one-dimensional viewpoint estimation task. We then explore classification methods for the 1D viewpoint estimation task.

Since classification models require a finite set of classes, the viewpoint space is discretized into bins of 1-degree intervals. This leads to a total of 360 classes. We use the nearest neighbor (NN) classifier as a baseline for the classification models. The NN classifier is used to investigate the effectiveness of a simple classification model on the viewpoint estimation task, and to establish a baseline against which other classification models can be compared. We use a convolutional autoencoder to reduce the images to lower-dimensional feature representations. Thus the convolutional autoencoder is used to compute the feature vectors of all training images. Given a test image, its feature vector is computed using the convolutional autoencoder. The feature vector of the test image is then compared to the feature vectors of the images in the training set using the Euclidean distance. The viewpoint label of the training image corresponding to the smallest feature vector distance is returned as the predicted viewpoint. This classification method yields an accuracy $Acc_{\frac{\pi}{6}} = 57.78\%$ and a median error $MedErr = 23.50$. Thus nearest neighbor classifier gives better results than the regression model. However, the performance remains relatively poor.

Moreover, we investigate a CNN-based classification model. Like the CNN-based regression model, we leverage transfer learning to build the classification model. We use the Inception-V3 model which was pre-trained on ImageNet as a base model. This base model extracts useful features from the images. The features extracted are then passed through a classification block made of fully-connected layers. The output layer is a fully-connected layer with 360 neurons corresponding to the number of viewpoint classes. The whole classification model is fine-tuned on the training data of DRRs by minimizing a cross-entropy loss. This CNN-based classification model gives similar results to the NN classifier with an accuracy $Acc_{\frac{\pi}{6}} = 56.11\%$

and a median error $MedErr = 19$. One limitation of this model is that it classifies the viewpoints independently, which does not take into account the continuity of the viewpoint space.

1.3.3 Geometry-aware classification

The CNN-based viewpoint classification model proposed in Chapter 4 is trained on a cross-entropy loss that requires the viewpoint classes to be one-hot encoded. The one-hot encoding labels assign a weight of 1 to the ground truth class and 0 to all the other classes, as in a normal classification scheme. However, the continuity of the viewpoint space implies that some viewpoint classes are closer than others. This specificity of the viewpoint estimation problem is not considered in the cross-entropy loss with one-hot encoded class labels. In Chapeter 5 we use a soft label encoding strategy where the weight of the ground truth class is set to 0.2, the weights of the eight nearest neighbors to the ground truth class (i.e. within 4° to the ground truth class) are set to 0.1, and the weights of all the other classes are set to 0. We trained a CNN-based classification model with a weighted cross-entropy loss, where the weights of the loss are the entries of the soft label vector. We use a pre-trained model on ImageNet as a base model to extract image features. The base model is completed with a classification block made of fully-connected layers. The entire model is fine-tuned end-to-end until convergence. The performance of the model evaluated on the test set shows the effectiveness of the soft label encoding with an accuracy $Acc_{\frac{\pi}{6}} = 76.39\%$ and a median error $MedErr = 9$. Although the soft label encoding performs well compared to the one-hot encoding, the fact that the softening of the label is explicitly defined makes the method application-specific. This approach might not work well in other cases where the similarity between viewpoint classes requires a different type of label softening. One alternative to this explicit softening of the labels is to use a geometry-aware cost function that implicitly encodes the softening of the labels based on the similarity between the viewpoint classes.

The geometry-aware loss is a weighted version of the usual cross-entropy cost function where the weights are defined by the distance between classes. The label weights are an exponential decay of the angular distance between classes. The hyperparameter in the geometry-aware cross-entropy loss controls the weights of the viewpoint classes. We trained the CNN-based classification model using the geometry-aware loss. This model gives a similar performance to the model trained with the soft-label encoding with an accuracy $Acc_{\frac{\pi}{6}} = 75.28\%$ and a median error $MedErr = 8$.

The CNN-based classification models defined previously are trained using small random translation and in-plane rotation for data augmentation. Since we generate the training from a CT scan volume, we could also augment the training data with small out-of-plane rotations as well. However, due to space constraints, it is impractical to augment the training data with out-of-plane rotations when the training data is generated offline. Instead, we opt for an online approach where the training data is generated on the fly before each training iteration. This allows us to implement the out-of-plane rotation data augmentation without having to store all the augmented images. Given that the data generation is coupled with the model training, we require a fast data generation method to keep the training process tractable. Hence we use the FVR method for the online data generation. We train the geometry-aware classification model with the online data generation method and out-of-plane rotation data augmentation. The results show that the out-of-plane data augmentation improves the performance of the model with accuracy $Acc_{\frac{\pi}{6}} = 81.94\%$.

In the previous experiments, the training data was generated from a single CT volume. We use a different CT scan to generate the test data. To improve the generalization power of the model, we extend the training data by generating the DRRs from multiple CT scans. Since these CT scans are taken from different persons who might lie on the imaging table in slightly different orientations, the viewpoint labels become noisy. In fact, the same viewpoint class might correspond to a slightly different orientation from one patient to another. Since the patients have different shapes and sizes, they might appear shifted in the images compared to other patients. This increases the variability in the dataset. To capture the shifting in the images and the varying patient size, we implement translational cropping data augmentation with different translation ranges. The effect of input image size on the model's performance is also investigated. Despite the noisy labels, the model trained using data generated from the multiple CT scans performs well with an accuracy $Acc_{\frac{\pi}{6}} = 80.45\%$.

In addition to the augmentation done during training, we also investigate a test-time data augmentation strategy where each test image is augmented with small horizontal and vertical translations. The model predicts a candidate viewpoint for the original test image as well as its transformed versions. The predicted viewpoint is derived from the candidate predictions via aggregation or a confidence scoring process. We use an average scoring method proposed by Krizhevsky et al [42]. We compare this aggregation technique to a couple of confidence scoring methods based on the entropy and an autoencoder confidence score. The entropy-based confidence scoring method proves to be the most effective approach with an accuracy $Acc_{\frac{\pi}{6}} = 83.68\%$.

1.3.4 3D viewpoint estimation

We extend the viewpoint estimation task to full 3D rotation estimation in Chapter 6. Several 3D rotation representation methods are investigated, particularly the Euler angle and the quaternion representations. The Euler angle representation is made of three independent rotation angles around each axis of the Cartesian coordinate system. For the 3D viewpoint estimation, we train a CNN-based classification model that predicts the three rotation angles simultaneously. The results show that the rotation angle around the longitudinal axis is the most difficult for the model to predict. A better alternative to the Euler angle representation is the quaternion representation. Unit quaternions are used to represent 3D rotation efficiently. The quaternion representation allows a nice interpolation of 3D rotation, which can be beneficial to a regression model. Hence we investigate the performance of the viewpoint estimation models using regression and classification models.

We propose a CNN-based regression model using a quaternion representation. This model uses a pre-trained CNN model for image feature extraction. The extracted features are passed through a regression block made of fully-connected layers. The output of the model is an FC layer with four neurons representing each of the four entries of the unit quaternion vector. The whole model is fine-tuned end-to-end by minimizing a loss that measures the distance between the predicted quaternions and the ground truth quaternions. The model is trained on a set of DRRs generated at sampled quaternions on the viewpoint space. We investigate the influence of the number of sampled quaternions on the performance of the viewpoint regression model. The results show that the performance of the model improves with a higher number of sampled quaternions. However, the performance plateaus after a certain number of samples. The quaternion representation significantly improves the performance of the viewpoint estimation model, with an accuracy $Acc_{\frac{\pi}{6}} = 91\%$ for 5000 sampled unit quaternions.

We also investigate a CNN-based classification model using the quaternion representation. We define the classes by sampling unit quaternions uniformly on the viewpoint space. In general, a high number of viewpoint classes leads to better performance since the viewpoint space gets denser, which reduces the misclassification error. However, as the number of classes increases the model parameters also increase, which can make the model difficult to train. We experiment with different numbers of viewpoint classes. The classification model is fine-tuned on the dataset of DRR using a geometry-aware loss that takes into account the distance between viewpoint classes. This classification model gives a comparable performance to the regression model, with an accuracy $Acc_{\frac{\pi}{6}} = 92\%$.

1.3.5 Viewpoint estimation for natural images

We investigate how the proposed viewpoint estimation methods for medical images perform on natural images in Chapter 7. The PASCAL3D+ dataset, made of twelve object categories with viewpoint annotation, is used to train CNN-based classification and regression models for viewpoint estimation. Several data augmentation techniques such as pose jittering [56] and synthetic data are used. The PASCAL3D+ is initially labeled using an Euler angle representation. We convert the Euler angles to unit quaternions. We then train CNN-based classification and regression models using the unit quaternion representation.

The regression model uses a VGG-16 model pre-trained on ImageNet for image feature extraction. The features extracted are passed through a regression block. The entire network is fine-tuned on the PASCAL3D+ dataset. The results show that the CNN-based regression model gives comparable performance to the baseline method. Moreover, we train a classification model on the PASCAL3D+ dataset using the quaternion representation. We define the viewpoint classes by sampling uniformly unit quaternions on the viewpoint space. Each training image is assigned to the closest viewpoint class. We investigate two types of classification models: a category-specific model and a global model. For the category-specific models, we train one CNN model for each object category. On the other hand, the global model is trained to predict the viewpoint regardless of the object category. The results prove that the category-specific model performs better than the global model. However, the category-specific model has a higher computational cost than the global model.

1.4 Contributions

The main contributions of this project can be summarized as follows:

- An efficient implementation of the Fourier volume rendering (FVR) to generate a dataset of digitally reconstructed radiographs that are used to develop viewpoint estimation methods for medical images.
- A geometry-aware classification model for viewpoint estimation from 3D CT scans.
- A combination of a quaternion representation and geometry-aware classification for 3D viewpoint estimation.
- Publications:

1. **A. Hounkanrin**, P. Amayo, and F. Nicolls. "Content-Based Medical Image Retrieval Using a Class Similarity-Aware Cross-Entropy Loss". In: Pillay, A., Jembere, E., Gerber, A. (eds) Artificial Intelligence Research. SACAIR 2022. Communications in Computer and Information Science, vol 1734. Springer, Cham. DOI: 10.1007/978-3-031-22321-1_2.

2. X. Nkwentsha, **A. Hounkanrin**, and F. Nicolls, "Automatic classification of medical X-ray images with convolutional neural networks", 2020 International SAUPEC/RobMech/PRASA Conference, 2020, pp. 1-4, DOI: 10.1109/SAUPEC/RobMech/PRASA48453.2020.9041052.

1.5 Thesis outline

In Chapter 2 we give a brief background theory of the concepts that are necessary for a good understanding of the methods presented in this thesis. We first discuss the theory of neural networks. In the second part of the chapter we present the literature review, where the most relevant works in the literature are discussed. We particularly focus on work on learning-based viewpoint estimation.

Chapter 3 presents the methods used to generate the dataset for viewpoint estimation from a set of CT scan volumes. A dataset with accurate viewpoint annotation is required to train the viewpoint estimation models. Given the lack of such a dataset, we generate a dataset of digitally reconstructed radiographs (DRRs) with viewpoint annotations from the CT volumes. To simplify the data generation process, we only consider orthogonal projection methods instead of fan-beam projection. Two data generation methods are used: additive projection (AP) and Fourier volume rendering (FVR). Additive projection provides an effective technique to generate DRRs with viewpoint labels from CT volumes. However, this method is computationally expensive since it requires the rotation of a 3D volume before the 2D image corresponding to a given viewpoint can be generated. Fourier volume rendering offers a more computationally efficient alternative.

In Chapter 4 we present the methods proposed for 1D viewpoint estimation. Since the viewpoint space is continuous by nature, we first investigate regression-based methods. We develop a regression model based on a convolutional neural network (CNN). The regression model does not perform well for the 1D viewpoint estimation task. We then propose classification-based viewpoint estimation methods. To establish a baseline, we use a k-nearest neighbor classification model. We then investigate CNN-based classification models. In general, the classification models perform better than the regression ones. In particular, the CNN-based classification models give the best performance on the 1D viewpoint estimation task.

In Chapter 5 we present a geometry-aware classification method to deal with the continuous nature of the viewpoint space. The circular distribution of the viewpoints makes some classes closer than others. For the classification models presented in Chapter 4, the images are classified independently without regard to the similarity between classes. In this chapter, we first propose a label softening strategy whereby similar classes to the ground truth viewpoint are assigned non-zero weights according to their proximity to the ground truth viewpoint. This approach significantly improves the performance of the classification-based viewpoint estimation model. We then use a more general method that incorporates label softening into a geometric structure-aware cost function. This approach gives a similar performance to the label softening strategy and has the advantage that the label softening can be adjusted depending on the application.

In Chapter 6 we generalize the problem to three-dimensional viewpoint estimation. We explore different representation methods for 3D rotations, notably the Euler angle, axis-angle, and quaternion representations. The quaternion representation proves to be more effective for 3D viewpoint estimation than other methods. We propose CNN-based classification and regression models to predict 3D viewpoints. In particular we combine the discriminative ability of CNN models and the efficient viewpoint representation of unit quaternions to develop CNN-based viewpoint classification models using the unit quaternion representation.

In Chapter 7 we investigate how the methods proposed for viewpoint estimation in medical imaging generalize to natural images. We use CNN-based regression and classification models to solve viewpoint estimation on the PASCAL3D+ dataset. The performance of the proposed methods on PASCAL3D+ shows that their application is not limited to medical imaging.

We conclude the thesis in Chapter 8 with a summary of the contributions of the work and a proposition of possible research avenues.

Chapter 2

Background and related work

This work investigates various methods for viewpoint estimation in medical imaging. In particular, we develop deep learning models to predict the viewpoint of 2D projections of a human body from 3D CT scans of the body. The proposed methods rely on the successful application of deep learning models to viewpoint estimation in medical imaging. It is therefore necessary to present the relevant theoretical notions used. The first part of this chapter is dedicated to the presentation of the background information. Moreover, to give some context to the project, we review the related work.

Since the methods proposed are primarily based on deep learning models, we present a theory of neural networks in Section 2.1. A mathematical formulation of the viewpoint estimation problem is presented in Section 2.2. This is followed by the presentation of the related work. Section 2.3 is dedicated to the methods proposed for viewpoint estimation in medical imaging. We present a literature review on 2D/3D image registration in Section 2.4. We then discuss the state of the art in viewpoint estimation for natural images in Section 2.5. We conclude the chapter in Section 2.6 with a summary of the work presented.

2.1 Neural networks

In recent years, predictive models based on neural networks have become the de facto method for solving computer vision problems. The growing popularity of these methods is due to the fact that, in many instances, neural network methods (and convolutional neural networks in particular) have proved to be more effective than classical computer vision methods that rely on hand-engineered feature extraction. It is therefore not surprising that we opt for CNN-based methods in this work. In this section, we present the foundations of neural networks from the earliest architectures to state-of-the-art methods based on CNN models. We present the general architecture of a CNN model as well as a few examples of CNN models that are known to be effective at solving computer vision problems. We also discuss the optimization methods that are used to train a CNN model. We finally discuss the evaluation methods and machine learning libraries used to implement CNN models.

2.1.1 Multi-layer perceptron

The multi-layer perceptron (MLP) is the simplest neural network and is made up of at least three layers: an input layer, one or more hidden layers, and one output layer. The MLP is a feedforward neural network, i.e. the data flow in the network is unidirectional: from the input layer to the output layer. One particularity of the MLP model is that all these layers are fully connected. Thus, each neuron in a fully-connected (FC) layer is connected to all the neurons of the preceding layer, resulting in a dense connection of neurons. For this reason, the FC layers are also called dense layers. Each layer of the MLP model is a parametric function that maps its inputs to some output values. Since the architecture of an MLP model is organized such that the output of a given layer is used as the input to the following layer, the mapping from one layer to the following layer can be regarded as an intermediate function. The function mapping the inputs to the last layer is the composition of the successive intermediate functions. Figure 2.1 shows the architecture of an MLP model with two hidden layers.

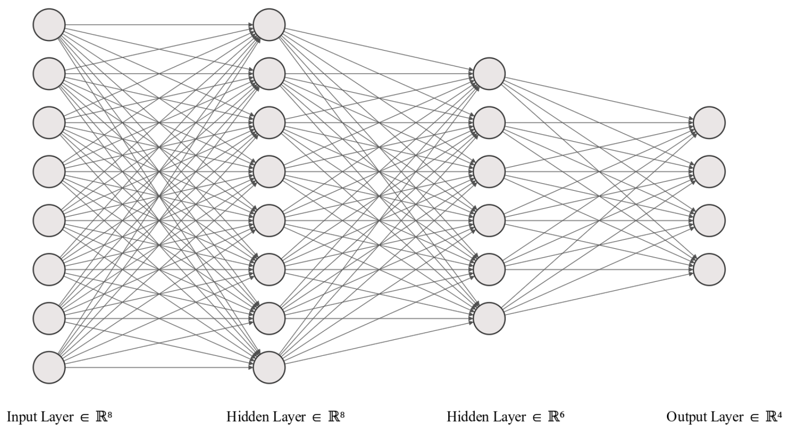


Figure 2.1: Architecture of an MLP model with four layers.

This architecture has a directed and weighted graph structure, where the nodes represent the neurons of the model and the weights of the edges are the parameters of the MLP model. If all the intermediate functions of the model presented in Figure 2.1 were linear mappings, the resulting function would also be linear since the composition of linear maps is again a linear map. Consequently, the hidden layers would become useless and the model would reduce to just two layers: an input and an output layer. To avoid this problem, the outputs of the hidden layers are passed through non-linear activation functions.

Non-linear activation functions

The raw output from a given neuron in the MLP model is the weighted sum of the outputs of all the nodes in the preceding layer. Since this is a linear map, the raw output of the neuron is passed through a non-linear function, called an activation function. The presence of this non-linear activation function is necessary to preserve the architecture of the model. The logistic functions are the most common activation functions for MLP models. Figure 2.2 shows two examples of these activation functions: the hyperbolic tangent (tanh), and the sigmoid function.

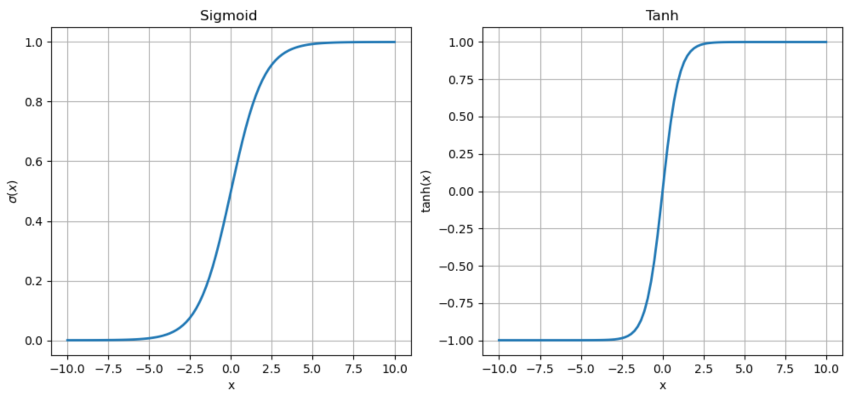


Figure 2.2: Sigmoid activation (left) and hyperbolic tangent (right) activation functions.

The tanh activation function is defined by

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \tag{2.1}$$

and is in the range $(-1, 1)$. The sigmoid activation function is defined by

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \tag{2.2}$$

and has range $(0, 1)$. Thus, in addition to preserving the architecture of the model, these activation functions constrain the output values to bounded ranges. One major inconvenience of this is that the neurons are subject to saturation, i.e. the output of the activation function becomes asymptotic for large and small input values.

The non-linear activations allow the model to learn non-linear maps, to separate data that are not linearly separable. Furthermore, the presence of non-linearity ensures that the model has hidden layers, which makes it more powerful according to the universal approximation theorem [32]. In essence, the universal approximation theorem states that any continuous function can be approximated, to arbitrary precision, with a feedforward neural network with at least one hidden layer and a sigmoid activation function.

Training an MLP model involves fitting it to the input data by adjusting its parameters. This training is done by gradually updating the model's parameters such that some error function is minimized. Since neurons in an MLP model are densely connected, the number of parameters to train grows exponentially with the number of layers. It becomes difficult to train such models since the optimization takes longer and requires more data to learn from. Convolutional neural networks offer a more efficient alternative to MLP models.

2.1.2 Convolutional neural networks

Convolutional neural networks (CNNs) have become the most common method used to solve computer vision problems. A CNN has similar functioning principles to the MLP model. However, CNN architectures are in general more efficient than fully dense networks thanks to the parameter sharing strategy used in CNN models. Similar to MLP models, the architecture of a CNN model is organized such that the output of one layer is the input to the following layer. Figure 2.3 shows the general architecture of a CNN model. CNN models mainly consist of three types of layers: convolutional layers, pooling layers, and fully-connected layers.

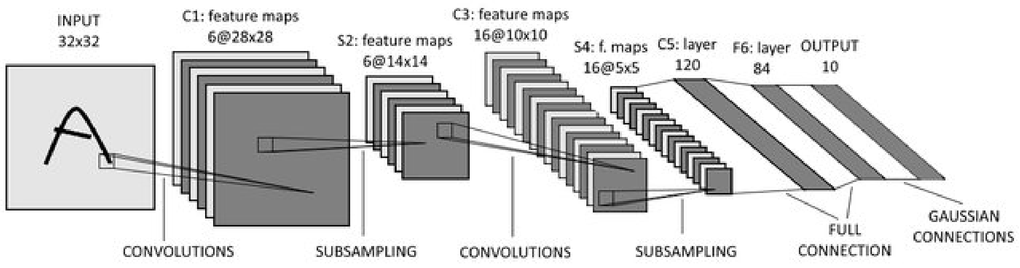


Figure 2.3: Architecture of LeNet-5 [43].

Convolutional layers

The convolutional layer is the main layer of the CNN model. This layer extracts features from an input image using a set of convolution filters also called kernels. In general, convolution filters are 2D arrays of parameters that are convolved with the input image to generate a feature map. In convolutional layers, convolution kernels are applied to inputs to extract feature maps. These feature maps can be passed to another convolution layer to extract higher-level features. A feature map is generated by moving a convolution kernel across the input and performing the convolution operation at each location of the kernel. The convolution operation computes a weighted sum of the input feature map at each location of the kernel. Since this operation is linear, convolutional layers are usually followed by a non-linear activation function to introduce some non-linearity into the model and allow it to learn non-linear maps.

Pooling layers

The outputs of the convolution layers are feature maps with relatively high dimensions. Since the convolution kernel is applied to the feature map of the preceding layer, a high-dimensional feature map increases the computational cost of the convolutional operation. To reduce the size of the feature maps, the convolutional layers are usually followed by pooling layers. Pooling layers are non-parametric layers using a pooling kernel to extract salient features and aggregate features from the input feature map. The most common types of pooling layers are the max-pooling and the average-pooling layers. The max-pooling layer uses a sliding kernel to extract the maximum value from a patch in the input feature map. Thus max-pooling allows the extraction of the most salient features from a local region in the input feature map. Similarly, average-pooling uses a sliding window to extract the average value from local patches in the input feature map. This results in a feature map made of local aggregation of the input feature map. Thus, the pooling layers not only make the CNN models more computationally efficient but also extract more powerful features from the convolutional layers.

Fully-connected layers

The last part of a CNN model architecture is often made of fully-connected (FC) layers, also called densely-connected layers. The FC layers have a similar structure to the MLP model. An FC layer is made of a set of neurons that are connected to all the neurons in the previous layer. The purpose of the FC layers is usually to classify/regress the features extracted by the convolution layers and the pooling layers. The output of each FC layer is passed through a non-linear activation function.

Activation functions

Like in MLP models, the tanh and the sigmoid activation can be used. However, the saturation of these activation functions (asymptotic for large and small input values) is a big inconvenience for deep CNN models. In effect, the saturation of neurons prevents the flow of the gradients through the network. This is known as the vanishing gradient problem, whereby the gradients of lower layers become negligible. The vanishing gradient problem prevents the model from learning the weights of the lower layers since their gradients are close to zero, and the weights are hardly updated during training. To avoid the vanishing gradient problem, other non-linear activation functions are used for CNN models. The most common non-linear activation function for CNN models is the rectified linear unit (ReLU) activation function [64] defined by

$$f(x) = \max(0, x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0. \end{cases} \tag{2.3}$$

This function sets the negative feature values to zero and applies an identity map to the positive values. The ReLU is a non-saturating activation function for positive input values, which mitigates the vanishing gradient problem. Other variants of this activation function are proposed in the literature. Most notably, the Leaky-ReLU [55] is defined by

$$g(x) = \max(\epsilon x, x) = \begin{cases} \epsilon x & \text{if } x \leq 0 \\ x & \text{if } x > 0, \end{cases} \tag{2.4}$$

where $\epsilon \ll 1$. The Leaky-ReLU is a modification of the ReLU which prevents the negative feature values from being discarded. Figure 2.4 shows the graphs of the ReLU activation function and the Leaky-ReLU activation function for $\epsilon = 0.05$.

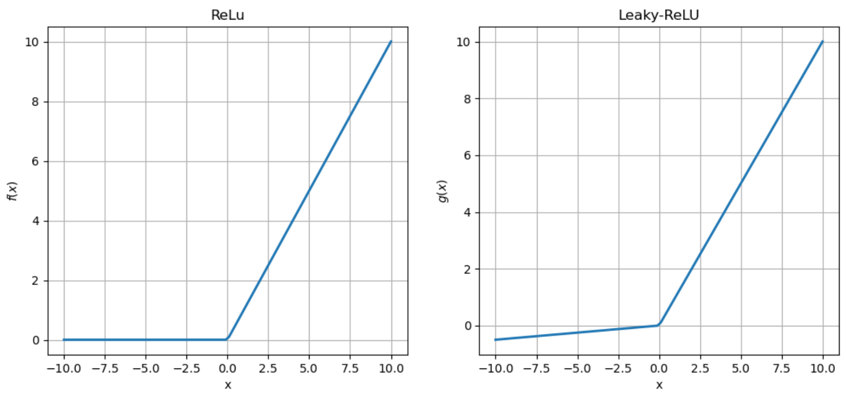


Figure 2.4: ReLU (left) and Leaky-ReLU (right) activation functions.

2.1.3 Optimization of CNN models

CNN model parameters are often initialized randomly. These parameters need to be adjusted to fit the training data through an optimization process.

Objective functions

The objective function of a CNN model is called a cost or loss function. The loss measures the distance between the predicted output and the expected output. In a supervised learning setting, the expected output is given by the labels of the input. There are several loss functions that can be used to train a CNN model, depending on the specific application. For classification tasks, one of the most common cost functions is the cross-entropy loss. The cross-entropy loss is defined by

$$CE = -\sum_{i=1}^{N} y_i \log(p_i), \tag{2.5}$$

where N is the number of classes, $y = (y_i)_{1 \leq i \leq N}$ is a one-hot-encoded label vector, and $p = (p_i)_{1 \leq i \leq N}$ is the output probability vector. This cost function measures the distance between the input and the output labels, assimilated to probability distributions. The mean square error is another common loss function that is used to optimize regression models. This loss function is defined by

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2, \tag{2.6}$$

where y_i and $\hat{y}_i$ are the expected and the predicted output values respectively. The MSE loss measures the average distance between the target value and the predicted value. These cost functions serve at guiding the CNN model to make better predictions using an optimization algorithm.

Optimizers

Optimizers are used to train a CNN model. Most of these optimizers are variants of the stochastic gradient descent (SGD) [39] algorithm. The SGD optimizer minimizes the cost function by iteratively updating the model's parameters in the negative direction of their gradients. Let W_i be the state of the model parameters at iteration i. Then the state of the parameters at the next iteration using a vanilla SGD optimizer is obtained by

$$W_{i+1} = W_i - \eta \nabla L_i, \tag{2.7}$$

where L_i is the loss at iteration i and η is the learning rate. Thus the SGD optimizer relies on the computation of the gradient of the loss function with respect to each parameter in the model. With the large number of parameters in a neural network, this computation could become computationally prohibitive without an efficient method for calculating the gradients. The backpropagation algorithm [72] is used to compute the gradients efficiently. Backpropagation uses the chain rule for differentiation to compute the gradients from the output layer to the input layer. Thus the gradients are propagated backward through the network. There are other optimizers used to train neural networks such as the Adam optimizer, the Adagrad optimizer, the Adadelta optimizer, and the RMSprop optimizer. Although some of these optimizers are more effective in specific applications than others, they all share the same basic principle as the SGD optimizer.

Training

Training a neural network is the process of minimizing the cost function by iteratively updating the parameters of the model. The training process starts with the initialization of the model's parameters. This initialization can be done randomly to train the model from scratch or from a pre-trained model for transfer learning. At each iteration, the network makes predictions on the input, and the distance between the prediction and the ground truth is evaluated by the cost or loss function. The gradient of the loss function with respect to each trainable parameter is then calculated via back-propagation and the parameters are updated using a gradient descent optimizer. The data is often divided into three sets: the training set, the validation set, and the test set. The training set is used to update the model parameters, while the validation set is used to monitor the performance of the model during training. The validation set also serves to select the best model after training. The generalization ability of the model selected model is then evaluated on the test set.

Over/under fitting

The success of the training process depends not only on the model (architecture) but also on the data. When the model does not have enough parameters, it cannot learn patterns in the dataset. The model is said to underfit the training data. On the order hand, a model that has a high number of parameters would be able to learn from the data. However, a model with too many parameters risks to overfit the training set. Overfitting occurs when the model learns not only the discriminative patterns in the data but also the noise, in an attempt to minimize the loss function. A classical method to prevent the model from overfitting is the early stopping strategy, whereby the training is halted as soon as the performance of the model on the validation set stops improving. Other approaches to prevent overfitting include the use of regularization techniques to constrain the optimization.

Regularization

Regularization techniques are methods used to constrain neural network learning and prevent it from overfitting the training data. Regularization is often achieved by modifying the loss function (L1/L2 regularization), by implicitly changing the model architecture during training (dropout regularization), or by changing the target label encoding (label smoothing regularization):

- L2 regularization

  This regularization is achieved by appending an additional term to the loss function. The additional term is often a multiple of the L2 norm of the model parameters. Given a model with weights W, the cross-entropy loss can be modified with L2 regularization as

  $$L = -\sum_{i=1}^{N} y_i \log(p_i) + \lambda \|W\|, \tag{2.8}$$

  where $\|W\|$ is the L2 norm of the weights and λ is a hyperparameter that controls the regularization term. With this regularization, the loss increases with the norm of the model weights. Thus the effect of this regularization is to favor weights with small values at the expense of weights with large values. In other words, this regularization implicitly reduces the complexity of the model, which reduces the risk of overfitting.

- Dropout regularization

  Dropout reduces the complexity of the model during training by dropping out a random set of neurons in a given layer at each training iteration. This improves the robustness of the model to noise in the data. However, the dropout is not applied during inference in order to use the trained model at full "capacity". Using dropout during inference would introduce randomness into the prediction process, which is not desirable.

- Label smoothing regularization

  This regularization is often used in classification models. Label smoothing regularization (LSR) [81] replaces the one-hot-encoded target vectors with a noisy version of these labels. In effect, when trained with one-hot-encoded labels, the model tends to become "over confident" with its prediction (by assigning 100% prediction score to one class and 0 to the other classes). Let y be a one-hot-encoded label vector for a N-class classification task. We can apply LSR to y to obtain a smoother version of y defined by

$$y' = (1-\epsilon)y + \frac{\epsilon}{N}. \tag{2.9}$$

With this regularization, the probability of the ground truth class is $1-\epsilon+\frac{\epsilon}{N}$. This prevents the model from overfitting by scattering logit scores across many classes.

2.1.4 Evaluation metrics

After training, the performance of the CNN model is evaluated on the test set. There are several evaluation metrics such as accuracy, precision, recall, and F1-score. The choice of the evaluation metric depends on the application. Consider a binary classification problem. We can use the confusion matrix and the accuracy as evaluation metrics:

- Confusion matrix

  For a binary classification problem the confusion matrix has two rows and two columns. Using the rows to represent the ground truth and the columns for the predictions, the confusion matrix is presented in Table 2.1.

Table 2.1: Confusion matrix for binary classification. TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negative instances.

	Positive	Negative
Positive	TP	FP
Negative	FN	TN

- Accuracy

  The accuracy measures the proportion of correct predictions. The accuracy is defined by

$$Acc = \frac{TP+TN}{TP+FP+TN+FN}. \tag{2.10}$$

  In general, the accuracy is effective at measuring the performance of the model when the dataset is balanced. However, when the data is unbalanced the accuracy could be misleading. In effect, a model could have a high accuracy by predicting the dominant class correctly and ignoring the rare class. Other evaluation metrics such as precision, recall, and F1-score are often used as an alternative to the accuracy when the data is unbalanced.

2.1.5 State-of-the-art CNN models

Training a CNN model from scratch requires a lot of data which is not always available. In applications such as medical imaging, where the size of the training data is limited, it is beneficial to use the transfer learning strategy. For transfer learning, model weights are initialized with the weights of a pre-trained CNN model. There are several state-of-the-art CNN models that can be used to initialize a new model for transfer learning. In this section, we review the architectures of a few CNN models in the literature.

AlexNet

AlexNet was proposed by Krizhevsky et al. [42] for object classification on the ImageNet challenge in 2012. The architecture of the AlexNet model is presented in Figure 2.5. This model uses a set of five convolution layers with a ReLU activation function. Some of the convolution layers are followed by a pooling layer to reduce the resolution of the feature maps. Three fully-connected layers are used to classify the features extracted by the convolution layers. AlexNet outperformed competing methods at the ImageNet challenge by a significant margin. The success of AlexNet inspired many researchers who proposed improved CNN models using the same principles as AlexNet.

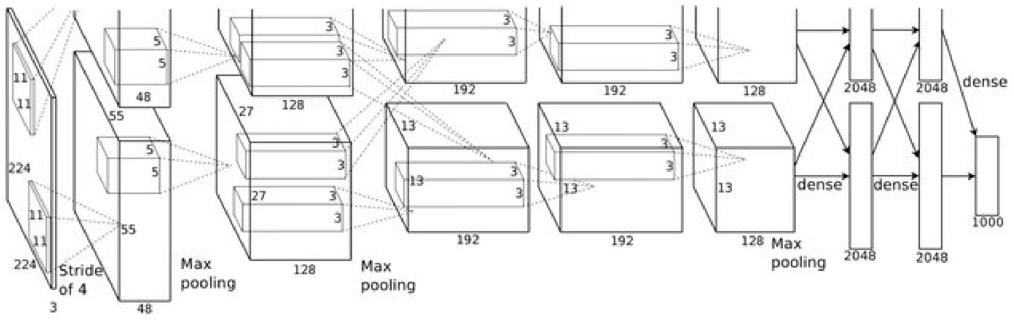


Figure 2.5: Architecture of AlexNet [42].

VGG

The VGG model was proposed by Simonyan and Zisserman [78]. There are several variants of the VGG model. The architecture of VGG-16 is illustrated in Figure 2.6, and is similar to the AlexNet model. However, VGG-16 uses smaller kernel sizes and a deeper network made of 16 layers.

Inception

Szegedy et al. [80] proposed the Inception model for object recognition. The particularity of this model is the use of a stack of modules for feature extraction. Figure 2.7

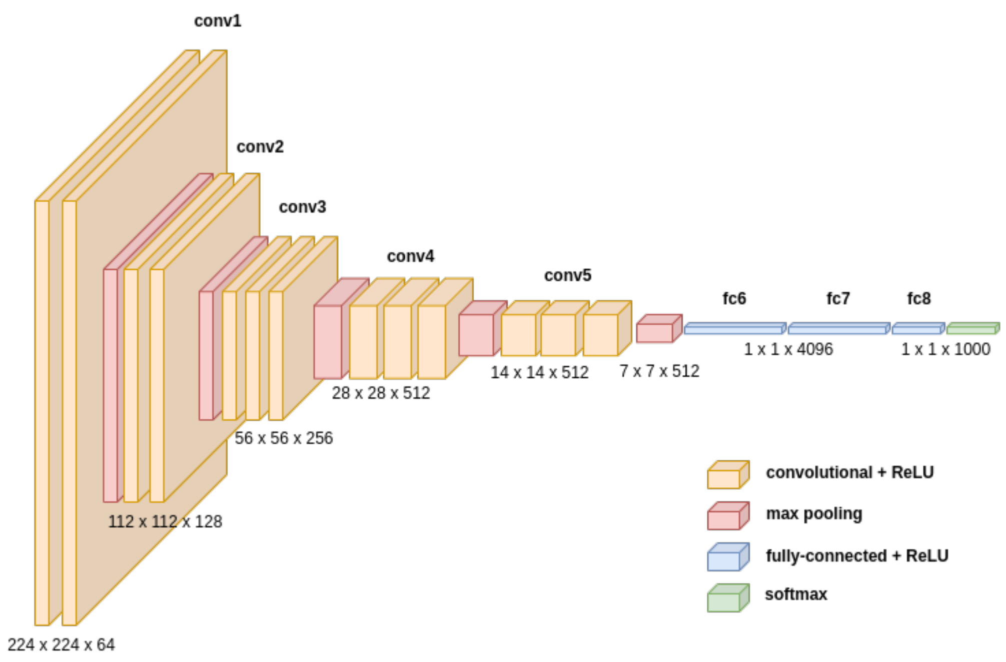


Figure 2.6: Architecture of VGG-16.

illustrates the architecture of inception modules. The stack of inception modules is followed by max-pooling layers to reduce the resolution of the feature maps.

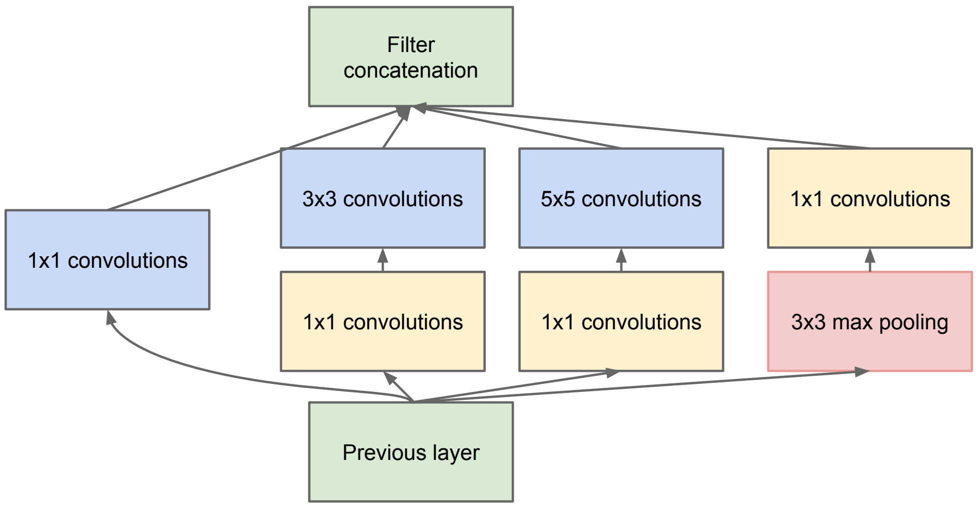


Figure 2.7: Architecture of the Inception module [80].

2.1.6 Machine learning libraries

Machine learning (ML) models such as CNN are usually implemented using libraries specifically developed for machine learning. Currently, the two most widely used libraries in ML and deep learning, in particular, are TensorFlow [1] and PyTorch [66]. In this work we use the TensorFlow library to implement the methods proposed. However, the methods proposed are not dependent on the library used, and any of the other ML libraries could have been used.

2.2 The viewpoint estimation problem

A 2D image of an object is essentially a projection from a three-dimensional space to a two-dimensional space. The image resulting from this projection depends on the position of the camera and the direction of the projection. In particular, in medical imaging, a 2D X-ray is a projection of a three-dimensional body part into a 2D image plane. The pose of the body part in the X-ray image obtained depends not only on the position of the X-ray source with respect to the body but also on the orientation of the body. Given an image resulting from the projection of a 3D object, the viewpoint estimation task aims at predicting the parameters of the camera (position and orientation). In a Cartesian coordinates system, the position is defined by the coordinates (x, y, z) of the camera and the orientation is defined by the angles $(\theta_x, \theta_y, \theta_z)$, representing the rotation angles around the x, y, and z axes, respectively. Thus the viewpoint of the object is defined by six parameters $(x, y, z, \theta_x, \theta_y, \theta_z)$. In this work, we focus on the estimation of the orientation parameters $(\theta_x, \theta_y, \theta_z)$. Moreover, we assume that the 2D images are generated from a 3D volume using orthogonal projection.

2.3 Viewpoint estimation from medical images

Early methods proposed in the literature for viewpoint estimation in medical imaging focus on the identification of standard X-ray image views (frontal and lateral) or very coarse in-plane rotation orientations. An illustration of the frontal and lateral chest radiograph views is illustrated in Figure 2.8. This simplifies the viewpoint estimation problem to a binary classification problem. This formulation of the viewpoint estimation problem ignores the intermediate views.

Boone et al. [5] used a neural network-based classification model to recognize chest X-ray image views by discretizing the viewpoint space into four bins. They used the frontal X-ray image as a reference view, which was rotated at 90 degrees intervals to generate three more views. Furthermore, each of the four views obtained by rotating the reference view was mirrored to generate four additional views. They then formulated the viewpoint estimation problem as an eight-class classification task. The proposed neural network is an MLP model with one hidden layer. The MLP model takes hand-engineered feature vectors of length 62 obtained by projecting the X-ray image pixel values horizontally and vertically. This method performs well on the chest radiograph dataset used (with an accuracy of 99.4%). However, this result

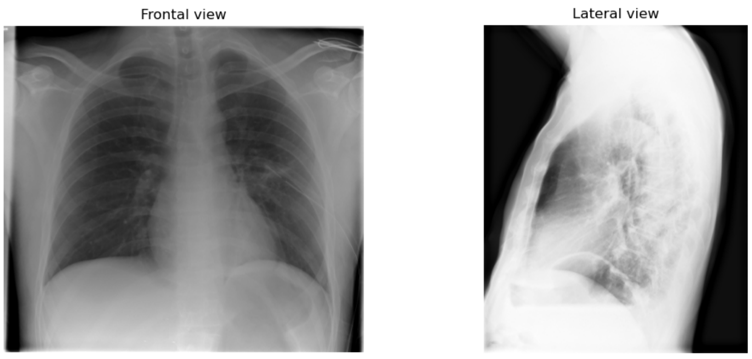


Figure 2.8: Frontal and lateral views of a chest X-ray image.

cannot be generalized since the viewpoint bins are very coarse, which makes the different classes very distinct and easy to classify. In addition, the method proposed by Boone et al. only applies to in-plane rotations of the X-ray images.

Similar to the method proposed by Boone et al. [5], Pietka and Huang [69] proposed a technique to predict the orientation of chest radiographs by identifying the orientation of the spine in the images. They used four different orientations of the chest X-ray images, and their mirrored images. A pixel profile analysis of the images was used to identify the orientation of the spine, which was subsequently used to determine the orientation of the image. The image was then rotated to match the orientation of the closest of the eight standard views. The authors reported an accuracy of 95.4% on their test dataset.

An automatic identification of chest radiographs method was also proposed by Arimura et al. [2]. They used a template matching technique to classify frontal and lateral views of chest X-ray images in a picture archiving and communication system (PACS). They defined nine templates (three for frontal views and six for lateral views) for patients of different sizes. Each template was obtained by averaging all images of the same view from patients of similar size in the training set. During inference, the correlation between a test image and all nine templates is calculated. The test image is then assigned the view of the template with the highest correlation. The authors reported an accuracy of 100% on the test set. However, the proposed method only solves the binary classification of frontal and lateral chest images. This is arguably the simplest version of the viewpoint estimation problem since it only deals with two classes. In addition, the template matching method used identifies the image views in steps, which is time-consuming and undesirable, particularly in medical applications where real-time inference is preferable.

Lehmann et al. [45] proposed a simplified and more computationally efficient method than the one proposed by Arimura et al. They used a k-nearest neighbor (kNN) classifier to identify frontal and lateral chest radiographs in the IRMA dataset [46]. They defined template images for the frontal and lateral views by averaging corresponding views in the training data. At inference time, the distance between a test image with the template was evaluated using several distance metrics such as the Euclidean distance, the correlation coefficient, the correlation function, and the tangent distance. Each of these distance metrics was then used to find the nearest neighbor to the test image among the standard views. The proposed method achieved a classification accuracy of 99.6% on the test set. Although more efficient than the approach of Arimura et al. [2], the method proposed by Lehmann et al. [45] only focuses on the standard views and does not deal with the intermediate views.

Boone et al. [4] used a multi-layer perceptron (MLP) model for a binary classification of lateral and frontal chest X-ray images. The proposed MLP model has one hidden layer with a sigmoid activation function. They reported a classification accuracy of 98.8% on the test set.

Kao et al. [35] used image projection profiles to identify frontal and lateral chest X-ray images. They computed the body symmetry index and the image background index from the projection profiles. These two indices were then used to classify the frontal and the lateral views in a dataset of chest radiographs. They reported an area under the receiver operating characteristic (ROC) curve of 0.993.

Luo et al. [53] used Bayesian inference to classify frontal and lateral chest X-ray images. In contrast to previous methods where features are extracted from the whole image, Luo et al. proposed to extract the features from regions of interest (ROI) in the images. The use of ROIs in place of the whole image makes their method more robust to the diversity in patient sizes. The authors reported a recognition accuracy of 98% on their test set.

A method to distinguish between the two types of frontal views (anteroposterior and posteroanterior) was proposed by Kao et al. [36]. They used three features to classify the anteroposterior (AP) and posteroanterior (PA) chest views: the index of radiolucence in the lung, the index of the tilt angle of the clavicle and the index of the tilt angle of the scapula. A linear combination of these three features was found to be the most effective feature, with an area under the ROC curve of 0.979.

Xue et al. [91] used a support vector machine (SVM) classifier to distinguish frontal views from lateral views in chest X-ray images. Features based on image profile, body size ratio, and pyramid of orientation gradients [6] were used as input for the SVM classifier. The proposed method was reported to have a classification accuracy of 99%.

Fang et al. [20] extended the standard view identification to other body parts. They proposed a CNN-based classification model pre-trained on ImageNet to classify X-ray images into lateral, frontal, or oblique views for several body parts (chest, ribs, ankle). They reported an accuracy of 90% on the test set. Similar to previous work in the literature, the method proposed by Fang et al. does not take into account intermediate views for general viewpoint estimation.

Salehi et al. [75] proposed a more general pose estimation method. They proposed a CNN-based regression model for 3D pose estimation of fetal brain MRI scans. They parametrized the 3D pose using the axis-angle representation, where the 3D rotation is represented by a vector and a rotation angle around the vector. The CNN model is trained using a loss composed of the mean square error (MSE) and the geodesic distance between rotations. The estimated pose from the regression model is subsequently used as initialization for image registration. This pose estimation method is more general than the other methods proposed in the literature since it is not restricted to particular views. In this work we propose a more general method to the one proposed by Salehi et al. that estimates the viewpoint of the human body from a computer tomography (CT) scan. We investigate regression as well as classification methods for the viewpoint estimation problem. Moreover, different viewpoint representation methods are investigated.

2.4 2D/3D image registration

The problem of viewpoint estimation is closely related to the one of image registration, which aims to find the geometric transformation that aligns a moving image to a reference image. An image registration system has two types of inputs: a moving image and one or more reference images. The transformation bringing the two input images to the same reference frame is typically determined by an iterative optimization process. This process starts with an initial guess of the transformation that is applied to the moving image, and the resulting image is compared with the reference image using some similarity metric. The transformation is then updated iteratively until the moving image is aligned with the reference image, or the similarity of the two images is within an acceptable threshold. In the particular case of 2D/3D registration, the reference image is often an intra-operative X-ray image and the moving image is a pre-operative CT volume. The goal of the 2D/3D image registration task is then to find the best geometric transformation that aligns the 3D volume with the X-ray image for better patient positioning in image-guided interventions. In this section we review some relevant works in 2D/3D image registration. We review the conventional method where the registration problem is solved as an iterative process, followed by the learning-based methods which attempt to solve the problem at once.

2.4.1 Conventional image registration

Russakoff et al. [73] proposed an algorithm to align a 3D pre-operative CT scan with a pair of 2D X-ray images. The registration method proposed aims to find the optimal rotation and translation parameters that bring the CT volume and each X-ray image to the same reference frame through a nearest neighbor optimization process. At each iteration of the registration process, DRRs corresponding to the current transformation parameters are generated using the attenuation fields method. The similarity of the DRRs generated with the corresponding X-ray images is measured using the mutual information similarity metric. The transformation parameters are then updated such that the similarity between DRRs and X-ray images is increased. This process is repeated until convergence and the transformation parameters corresponding to the optimal pose of CT volume are returned. The optimal transformation is then applied to the CT volume to align it with X-ray images. To reduce the registration time, only a region of interest (ROI) of size 200×200 in the X-ray image and the DRRs are used during the optimization process. The reported results show the effectiveness of the proposed method. However, the result is dependent on the accuracy of the choice of the ROI, which is done manually and is prone to errors. Also, given the nearest neighbor optimization strategy used, the convergence of the algorithm is dependent on the initial pose estimate.

Wein et al. [88] proposed a more efficient intensity-based 2D/3D image registration method using the gradient of the CT volume. They searched for the pose that best aligns a CT volume with a 2D X-ray image using an iterative procedure. At each iteration, the similarity of the DRR corresponding to the current pose of the CT volume with the X-ray image is computed using the gradient correlation (GC) similarity measure, which is based on the normalized cross-correlation (NCC) of the gradients of the DRR and the X-ray image. For computational efficiency, they used a volume gradient correlation technique where the gradient images are computed from the gradient of the CT volume. Similar to Russakoff et al. [73], they use an iterative optimization method based on the best neighbor search strategy. The evaluation of the proposed method using phantom and patient data show the effectiveness and efficiency of the method. However, the iterative pose optimation process requires an initialization close to the ground truth pose (within 20 mm for the translation parameters and 20 degrees for rotation parameters). Hence, the convergence of the optimization algorithm is not guaranteed when the initial pose is far from the ground-truth pose or when the ground-truth pose is unknown.

Fu and Kuduvalli [22] proposed an intensity-based 2D/3D image registration method for patient positioning during image-guided cranial radiosurgery. The authors separate the 3D transformations into two groups: in-plane transformation and out-of-plane rotations. A different search method is used to find the parameters of each group of transformations. To accelerate the registration process, DRRs are generated offline for predefined out-of-pane rotations that cover the range of patient orientation during patient positioning. The DRRs generated offline are used as references for out-of-plane rotations during registration. A multi-phase registration is used where the in-plane transformation and the out-of-plane rotations are searched separately using different similarity measures. The in-plane transformations are initialized using multi-resolution matching while the out-of-plane transformations are initialized using a one-dimensional search. During the registration process, the initial transformations are iteratively updated using steep descent minimization for the in-plane transformations and one-dimensional interpolation for the out-of-plane transformations. The authors used the sum of squared difference (SSD) as a similarity measure at the early stages of the registration process for its computational efficiency. The pose estimate using SSD is further refined with another stage of iterative registration where an optimized pattern intensity, based on the gradient of the X-ray and the DRR image difference, is used as a similarity measure. The proposed registration method is used to position a head-neck phantom. The performance of the registration method is evaluated using the result of a fiducial-based registration as a reference. The results show that the proposed method performs well when the transformation range is relatively small ($\pm 2°$). However, the accuracy of the registration deteriorates when the transformation range increases.

A single-plane 2D/3D registration method was proposed by Pickering et al. [68]. The 3D transformations (translations and rotations) that best align an X-ray image to a CT volume are determined iteratively using a gradient image registration algorithm. At each iteration, the estimated transformations are applied to the 3D CT volume, and a DRR is generated from that pose by summing the voxels of the transformed volume. Furthermore, a Laplacian-of-Gaussian (LoG) filter is applied to the X-ray image and the DRR for better edge detection. The sum of conditional variances (SCV) is used to measure the similarity between the filtered X-ray image and the DRR. This similarity measure is minimized using the Gauss-Newton optimization algorithm where the transformations are iteratively updated in the negative direction of the gradient of the SCV. The registration is performed for various LoG sizes to increase the displacement of the initial pose from the ground-truth pose for which the optimization algorithm will converge to the global minimum. This method is used to register a CT volume of a tibia and femur with synthetic fluoroscopy generated from the same CT volume. The results indicate that the proposed method is more efficient than non-gradient-based registration methods.

Fotouhi et al. [21] proposed an automatic re-initialization method for 2D/3D image registration using a pose-aware C-arm. An RGB-D camera is rigidly attached to the C-arm, and a vision-based tracking system is used to track the pose C-arm. The estimated C-arm pose is subsequently used to initialize the iterative registration process. At each iteration, a DRR corresponding to the current pose is generated from a CT volume. The DRR is then compared to the X-ray image acquired with the C-arm using normalized cross-correlation as a similarity measure. The rigid-body transformation is determined by maximizing the NCC in an iterative procedure using the bound-constrained optimization algorithm [70]. The performance of the automatic re-initialization technique is evaluated using a CT scan of a femur-pelvis phantom. The results show that automatic re-initialization performs significantly better than random initialization.

Bier et al. [3] proposed a method for anatomical landmarks detection in X-ray images acquired from arbitrary viewpoints, which was subsequently used for pose estimation and initialization of intensity-based 2D/3D image registration. They used a sequential prediction framework organized as a multiple stages network. At each stage, a belief map of the anatomical landmarks of interest is predicted. The final landmark detection is obtained by averaging the belief map from all stages. The proposed method is evaluated on both DRRs and real X-ray images. The reported results show that the method can accurately detect anatomical landmarks on X-ray images. The pose derived from the landmark detection result is used to initialize the registration of 2D X-ray images and CT volume of the pelvis with manually labeled anatomical landmarks. The authors reported a good registration performance using the proposed initialization technique when anatomical landmarks are clearly visible in the X-ray images and in the absence of surgical instruments in the image. A drop in performance is observed when images include surgical instruments, and when landmarks are hidden.

2.4.2 Learning-based image registration

In contrast with common computer vision tasks where deep neural networks have been largely adopted and are reputed to perform exceptionally well, the deep learning community seems not to have settled on the best way to apply CNNs for image registration. In effect, since 2012 with remarkable performance of AlexNet [42] for object recognition in the popular ImageNet challenge [13], deep neural networks have been successfully extended and applied to other tasks such as object detection [71] and segmentation [10]. Deep learning-based image registration, on the other hand, is still at an early stage although a few works have proposed methods to apply CNNs for image registration.

Iterative registration

Early applications of learning methods to medical image registration are a direct extension of the conventional framework. Thus those methods perform registration in an iterative manner. A similarity metric is usually learned followed by an optimization step where the best transformation is found iteratively. A common strategy is to formulate the similarity metric learning task as a binary classification (similar/dissimilar) problem [44, 62]. Lee et al. [44] used a support vector machine-based regression whereas Michel et al. [62] proposed an Adaboost method. Similarity-sensitive hashing is proposed by Bronstein et al. [8]. Although these methods show better results compared to conventional techniques on particular applications, they lack generalizability.

Zagoruyko and Komodakis [93] proposed to learn the similarity of a pair of images using different CNN architectures. They used a siamese network made of two branches with the same architecture and shared weights, a pseudo-siamese network where the weights of the two architecturally identical branches are not shared, and a 2-channel network where the input pair is used as a single image with two channels. The similarity of the pair of images is framed as a binary classification problem: matching and non-matching pairs. The 2-channel network gives better results, showing improved results for CNN-based similarity metrics over hand-crafted feature descriptors such as SIFT (scale-invariant feature transform) [52] and DAISY [82]. Several subsequent similar learning methods are based on their work.

Simonovsky et al. [77] proposed a CNN-based similarity metric for multimodal 3D image registration. They used the 2-channel network introduced by Zagoruyko and Komodakis [93] to estimate the dissimilarity of two cubic patches of the same size. The network consists of 3D convolutional layers with non-unit strides, preferred to a max-pooling operation for better performance. The learned similarity metric is used to perform deformable image registration on a set of neonatal images with better alignment than MI-based registration methods.

Cheng et al. [11] proposed a fully connected neural network to learn a similarity metric for the registration of computed tomography (CT) and magnetic resonance (MR) head images. They also formulate similarity metric learning as a binary classification problem (corresponding versus non-corresponding patches). Their network is initialized with a stacked denoising autoencoder (SDAE). However, images look very different across modalities, which makes any attempt to directly learn their correlation very difficult. To circumvent this, they apply a denoising autoencoder (DAE) to each imaging modality and then concatenate their high-level feature representations. This method shows better results compared to statistical methods such as mutual information (MI) and local cross-correlation (LCC).

Haskins et al. [30] formulated the determination of the similarity metric for 3D magnetic resonance (MR) and transrectal ultrasonography (TRUS) rigid image registration as a CNN-based regression problem. A pair of MR and TRUS images are fed to a 2-channel network, as proposed by Zagoruyko and Komodakis [93], which outputs an estimation of the target registration error (TRE). The network consists of 3D convolutional layers with unit strides. The learned similarity metric is then inserted into an iterative affine registration pipeline which finds the spatial transformations (translations and rotations) that best align the moving TRUS image to the MR image. Their similarity metric outperforms statistical methods such as MI and MIND (modality-independent neighborhood descriptor).

Gu et al. [27] proposed a CNN-based image registration method to improve the capture range of 2D/3D image registration. At each iteration of the registration process, a siamese CNN model takes the X-ray image and the DRR corresponding to the current pose as inputs and predicts the new pose estimate. The poses are updated following the Riemannian pose gradients. When the CNN model converges, the pose estimation is refined using conventional intensity-based image registration. The proposed method is used to register 3D CT volume with DRRs and X-ray images of the human pelvis. The results show that the method has a wider capture range compared to intensity-based registration techniques.

Similar to the work of Fotouhi et al.[21], Grimm et al. [26] proposed a method for automatic pose initialization for 2D to 3D image registration. A neural network is trained on DRRs generated from CT scans to detect the projections of 3D anatomical landmarks onto an X-ray image. To reduce the domain gap between real X-rays and DRRs, the authors used a domain randomization technique where DRRs are generated using different generators. Post-processing transformations are also applied to the input image to further increase the variability in the training data. They use the convolutional pose machine model [87] to detect anatomical landmarks in the input images. The initial 2D/3D registration pose is determined using a modified perspective-n-point (PnP) algorithm by minimizing a weighted reprojection error. The final registration is performed iteratively using Powell's optimization algorithm [70] with the cross-correlation similarity measure. The registration is done in three stages: translation optimization, rotation optimization, and joint translation and rotation optimization.

Jaganathan et al. [34] proposed a deep learning method for 2D/3D registration by incorporating the update step of iterative registration methods into a neural network. The method builds on the point-to-plane correspondence (PPC) method, introduced by Wang et al. [7], by learning the update operator. The PPC operator is embedded in a layer of a deep neural network and trained to perform the update operation. The model is trained by minimizing the mean target registration error (mTRE), which measures the distance between the ground-truth pose and the predicted pose. The authors reported an improvement in registration accuracy using the proposed method.

A self-supervised image registration method was proposed by Jaganathan et al. [33] to reduce the domain gap between DRRs and real X-ray images and improve performance when models trained on DRRs are tested on real X-ray images. They used the deep iterative registration network introduced by Jaganathan et al. [34]. An auto-encoder neural network is trained to estimate the correspondences between DRRs generated from different viewpoints. Furthermore, a transfer network is used for domain adaptation between X-ray images and DRRs. The proposed method is trained and evaluated on CT volumes of the thoracic and lumbar regions. The reported results show an improvement in performance compared to similar methods without domain adaptation.

One-shot registration

First attempts to apply learning-based methods to image registration focused on the similarity metric component of the classic registration process. However, these methods are time-consuming, which is a major drawback in medical imaging applications where a real-time response is highly desirable. Therefore a new research direction has emerged, whereby neural networks are used to register images in a single shot.

Chou et al. [12] used a learning-based 2D/3D image registration method for patient repositioning. A linear regression model is used to learn a patient position from a set of projections of a pre-treatment CT volume at sampled poses. Gaussian noise is added to the generated image projection to improve the robustness of the trained model. During treatment, the learned regression model is used to determine the transformation parameters that align the intra-operative X-ray image with the CT volume. The method's performance is evaluated on three patients' head-neck CT scans. The results show the effectiveness of the proposed method with a relatively low registration time.

Miao et al. [61] proposed a single-shot 2D/3D regression technique called hierarchical pose regressor (HPR) using a regression model based on a convolutional neural network. In HPR the rigid transformation is divided into groups: in-plane transformation parameters (two translations and one rotation) and out-of-plane transformation parameters (one translation and two rotations). A separate regressor is trained for each group of parameters hierarchically. This breaks down the initial regression problem into subtasks with reduced complexity. The CNN-based regression model is trained using the SSD as a cost function. Although a degradation in accuracy was observed, the method achieved a higher computational efficiency than classic registration techniques. This is an indication that CNNs can be effectively used to achieve real-time image registration, which is critical in medical imaging applications. A possible cause of the drop in accuracy in the proposed method might be the propagation of errors at one stage to the following stages of the hierarchical training adopted. Moreover, the authors show that HPR can be used for pose initialization in intensity-based iterative registration methods with promising results.

Han et al. [29] proposed a deep learning method for the registration of 2D ultrasound (US) images with 3D magnetic resonance (MR) volumes. They trained an Inception CNN model to regress the 3D pose of a US image with respect to an MR volume using the left-invariant Riemann distance between the ground-truth pose and the predicted pose as a cost function. The predicted pose is refined using a local structure orientation descriptor to improve the registration accuracy.

Other methods use a reinforcement learning approach where an artificial agent is trained to perform the registration task iteratively [41, 48, 54, 60, 67]. Although these methods demonstrate good results, a major drawback of reinforcement learning-based methods is their difficulty in handling high-resolution images. In fact, high-resolution images increase the action space of the artificial agent, which might be computationally prohibitive.

2.5 Viewpoint estimation from natural images

The release of the PASCAL3D+ [89] dataset triggered the development of deep learning methods for 3D object detection and viewpoint estimation. The PASCAL3D+ dataset is created from the PASCAL VOC 2012 dataset [19] by adding viewpoint annotations for all twelve object categories. The PASCAL3D+ dataset is augmented with more images from ImageNet [74] in order to make the dataset large-scale and suitable for deep learning methods. With twelve different object categories and more than 3,000 object instances for each category, the PASCAL3D+ dataset presents more variability than previous datasets. The PASCAL3D+ dataset thus became the primary benchmark for recent works on 3D pose estimation and viewpoint estimation. Recent works use either a regression approach or a classification approach to solve the viewpoint estimation problem.

2.5.1 Classification-based viewpoint estimation

Solving viewpoint estimation with a classification model is a challenging task as viewpoint space is continuous and lacks predefined boundaries from one viewpoint to another. An accurate delimitation of the viewpoint space is therefore a crucial step for classification-based viewpoint estimation methods. The common strategy is to discretize the viewpoint space into a finite set of bins representing the different classes [16, 79, 84].

After discretizing the viewpoint space defined by the three Euler angles (azimuth, elevation, in-plane rotation) into disjoint bins, Tulsiani and Malik [84] finetuned a CNN model on the PASCAL3D+ for viewpoint classification. The classifier, built on top of an object detector, predicts the viewpoint class of the object in the bounding box returned by the detector. New metrics ($MedErr$, Acc_θ, and $mAVP$) are introduced to evaluate the performance of this joint detection and viewpoint estimation task. The median error ($MedErr$) measures how far the predictions are from the target viewpoints; the accuracy at θ (Acc_θ) measures the fraction of predictions which are within θ degrees of the targets; the mean average viewpoint precision ($mAVP$) measures the correctness of the object detection and the viewpoint classification jointly. The performance of the proposed method is evaluated on the PASCAL3D+ dataset: $MedErr = 13.6$, $Acc_{\frac{\pi}{6}} = 81\%$, and $mAVP = 31.1\%$.

One major obstacle to the development of deep learning models for viewpoint estimation is the lack of sufficient training data with accurate viewpoint labels. Even the largest dataset available (PASCAL3D+) has only 22000 images, which can be limiting when developing an effective deep learning model. Su et al. [79] propose a method to augment the PASCAL3D+ dataset with millions of synthetic images rendered from 3D CAD models extracted from ShapeNet [9]. In order to prevent classifiers from overfitting, a random image selected from the SUN397 dataset [90] is used as a background for each rendered view from the 3D model. In addition, the images are randomly cropped and the objects are occluded to make the dataset more resistant to overfitting. To leverage the big dataset size, Su et al. partition the viewpoint space into fine-grained bins (360 viewpoints at 1-degree intervals). They train a CNN-based viewpoint estimation classifier on top of an off-the-shelf object detection model (R-CNN [23]) using real images from PASCAL3D+ and synthetic images generated from the 3D models. A geometric structure-aware cost function, derived from the cross-entropy loss function, is used to increase the correlation between nearby viewpoints. The performance of the proposed method is evaluated on the PASCAL3D+ dataset: $MedErr = 11.7$, $Acc_{\frac{\pi}{6}} = 82\%$, and $mAVP = 19.8\%$.

Massa et al. [59] studied the different factors that affect the performance of viewpoint estimation models (network architecture, dataset, and cost function). They report that the best performances are achieved using deeper CNN models and larger dataset sizes. They divide the viewpoint space into 24 bins and formulate the problem as a 24-class classification task. The authors reported a mean average viewpoint precision of 36.1% on PASCAL3D+.

Grabner et al. [25] propose a 3D pose estimation model based on convolutional neural networks. They found that networks with larger kernel sizes such as ResNet [31] perform better than those with smaller kernel sizes such as the VGG [78] models. Moreover, they upscale the images containing small objects to facilitate the detection of such objects by the CNN model. The results are evaluated on PASCAL3D+: $MedErr = 10.9$, and $Acc_{\frac{\pi}{6}} = 83.92\%$.

Kao et al. [37] use an appearance and structure fusion network to deal with the visual and structural ambiguities of objects in an image. They proposed a viewpoint prediction model with two parallel branches. The first branch (the appearance branch), based on the VGG model, learns the appearance of objects using viewpoint labels as input. The second branch (the structure branch), based on Feature Pyramid Networks (FPNs) [50], first estimates the keypoints of objects before predicting the viewpoint from the estimated keypoints. The appearance and structure branches are fused using a convolutional fusion layer which infers the final viewpoint probability vector by combining predictions from the two parallel branches. The authors use an adapted cross-entropy cost function to embed structure and appearance features. The performance of the proposed method, evaluated on the PASCAL3D+ dataset, is as follows: $MedErr = 7.9$, $Acc_{\frac{\pi}{6}} = 85.2\%$, and $mAVP = 31.4\%$.

In contrast with previous works, which separate object detection and viewpoint classification tasks by adding the viewpoint estimation classifier on top of an off-the-shelf object detection model, Divon and Tal [16] combined the viewpoint estimation and the object detection models as a unified network. This network is trained end-to-end to perform viewpoint estimation on the PASCAL3D+ dataset. Similar to the work of Su et al. [79], Divon and Tal train their model on an augmented version of PASCAL3D+. Each training image of the PASCAL3D+ dataset is flipped horizontally, and synthetic images rendered from 3D models are also used. They further generated more data by using nearby frames in videos of objects. The viewpoint estimation model is trained by optimizing a geometric structure-aware cost function in order to address the geometric nature of the problem. The performance is evaluated on PASCAL3D+ dataset: $MedErr = 8.9$, $Acc_{\frac{\pi}{6}} = 89\%$, and $mAVP = 45.9\%$.

Yang and Wang [92] proposed a key-point-based deep neural network to estimate objects' viewpoints. The proposed viewpoint estimation model (VE-Net) is made of two sub-networks: a key-points detection network (KD-Net), and a viewpoint estimation network (KV-Net). KD-net extracts keypoints from the input images, and the predicted keypoints are used by KV-Net to predict the viewpoint. The two sub-networks are trained jointly using a composite cost function that combines the key-point prediction error and the viewpoint prediction error. The KV-Net is based on a long short-term memory (LSTM) model and a fully-connected layer to classify the key-point into viewpoint classes. The authors reported the performance on the PASCAL3D+ dataset as follows: $MedErr = 14.3$, $Acc_{\frac{\pi}{6}} = 81\%$.

2.5.2 Regression-based viewpoint estimation

Given the continuous nature of the viewpoint space, regression seems like a natural fit for the problem. However, regression-based viewpoint estimation methods are very rare in the literature. This is possibly due to the fact that regression methods perform generally worse than classification-based approaches, which seem less appropriate for the task a priori. Here we review some of the best-performing works using CNN-based regression for viewpoint estimation.

Mahendran et al. [56] propose a regression model based on convolutional neural networks to solve the viewpoint estimation problem. They adapt the VGG network to perform continuous angle predictions. The viewpoints are represented using axis-angle and quaternion representations. They train the regression model on an augmented version of the PASCAL3D+ dataset (using in-plane rotations). Synthetic images from Su et al. [79] are also used to augment the data. The model is trained by minimizing the geodesic distance between the target and the predicted viewpoint. The results are reported on the PASCAL3D+ dataset: $MedErr = 15.38$, and $mAVP = 18.35\%$.

Liao et al. [49] propose a spherical regression approach where the regression model's output is constrained using an exponential activation function defined on an n-sphere space. The results are evaluated on the PASCAL3D+ dataset: $MedErr = 9.2$, and $Acc_{\frac{\pi}{6}} = 88.2\%$.

2.6 Summary

In the first part of this chapter, we presented a background theory of neural networks. A literature review of the different methods for viewpoint estimation was presented in the second part of the chapter. These methods were separated into three categories: the methods for medical images, the methods for 2D/3D image registration, and those for natural images. The viewpoint estimation methods for medical images focus on identifying coarse in-plane rotation angles or standard views such as lateral and frontal views. We also presented works in 2D/3D image registration where the rigid transformation aligning a 3D volume to its 2D projection is determined. In this work, we propose machine learning methods for viewpoint estimation in medical images. These methods can be used to improve single-shot image registration frameworks.

Chapter 3

Data generation

The foundation of a learning-based predictive model is the dataset on which the model is trained. Training a model on inadequate data would inevitably result in the model failing at the task for which it was designed. Especially for supervised learning methods where models are trained on labeled data, it is critical to ensure that the data is suitable for the task and that the labels are accurate. The models developed in this project are mostly based on supervised learning. We therefore require a dataset adapted for viewpoint estimation with accurate viewpoint labels.

In this chapter, we present how a dataset suitable for viewpoint estimation is generated from raw data of CT scan volumes. Section 3.1 describes the dataset of CT scan volumes from which the viewpoint estimation dataset is generated. In Section 3.2, we present the additive projection method used for offline data generation. Section 3.3 presents the Fourier volume rendering method used for online data generation. The experiments and results are presented in Section 3.4, followed by a presentation of other DRR generation methods in Section 3.5. We conclude the chapter in Section 3.6 with a summary of the different results obtained.

3.1 CT scans dataset

The raw CT scan dataset consists of 35 post-mortem CT volumes of human bodies. The data is collected from both male and female individuals of various age groups. This data is a subset of the SMIR full-body CT dataset [40]. Although the dataset was initially created for statistical shape analysis in medical imaging, we repurposed it to generate data for viewpoint estimation in medical imaging.

Each scan is embedded in a volume of size $512 \times 512 \times h$, where h varies according to the height of the body. In our experiments we use the upper part of the body. Thus the size of our CT volume is 512 × 512 × 512. Figure 3.1 shows a few views taken around our CT volume of interest. Points on this 3D grid represent voxels with intensities indicated by Hounsfield units (HU), also known as CT numbers. The voxel intensities indicate the attenuation of the X-ray radiation as it passes through the body. Similarly, pixel intensities on an X-ray image represent the attenuation of X-ray radiation through the body in a particular direction. Thus it is possible to generate digitally reconstructed radiographs (DRRs) by projecting voxels in a CT volume onto a 2D plane.

Since our goal is to build a model that is able to predict the viewpoint of a random radiograph taken of patients for image-guided diagnosis or surgery, we need to train the predictive model on a dataset that has the viewpoint information from the dataset of CT scans. There are several methods used to project CT volumes into DRRs. In the next section, we give an overview of the most effective methods.

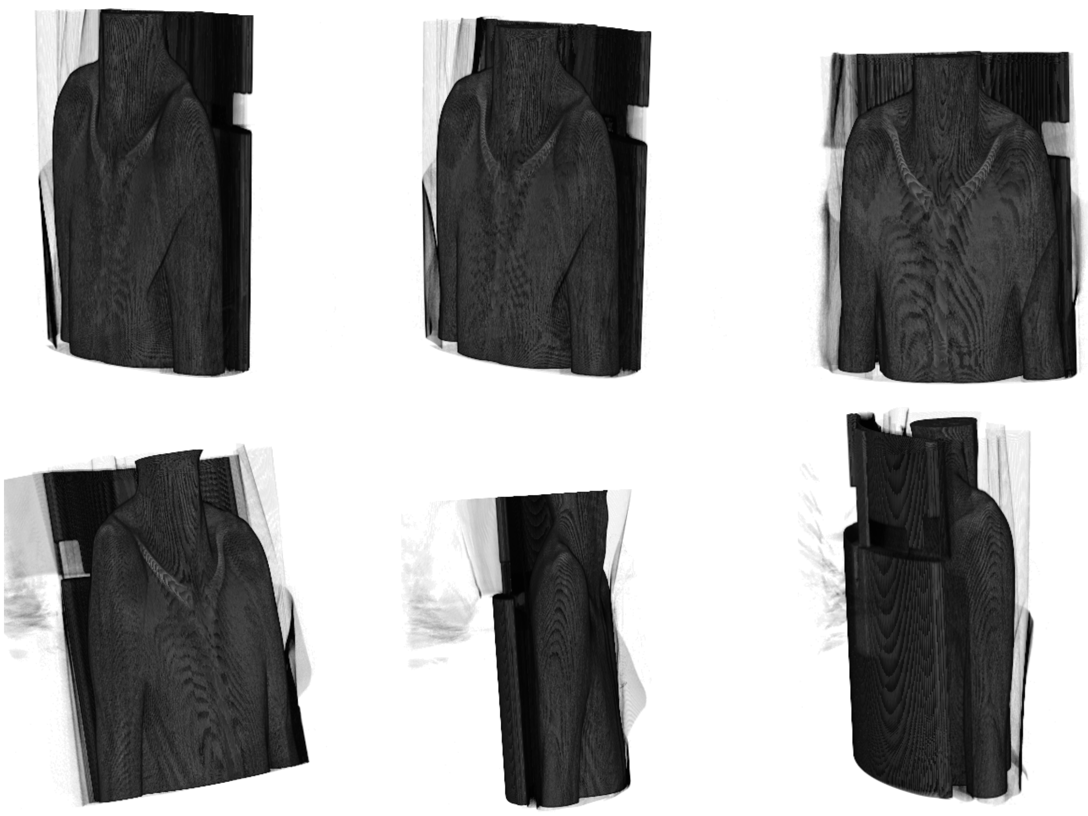


Figure 3.1: Views around a CT volume.

3.2 Additive projection

This method requires the rotation of the whole 3D volume around the longitudinal axis (z-axis). The viewpoint space is discretized into 360 bins at 1-degree intervals. For each bin, the 3D volume is rotated to the corresponding angle and we perform an orthogonal projection onto the yz-plane. The pixel values of the resulting image are obtained by summing up all voxel values in the 3D volume along a line orthogonal to the yz plane and passing through the pixel location in the projected image. Considering a volume of dimensions $N \times N \times N$, the generation of a single projection has a time complexity of $O(N^3)$. Thus the additive projection method becomes computationally expensive when the number of projections increases. In an offline training setting, where the dataset is generated prior to training, the additive projection could be used as a straightforward way to generate 2D DRRs from CT volumes. However, data generated offline do not allow full leverage of 3D information from the CT volume. Certain types of data augmentation such as random out-of-plane rotations are difficult to implement when the dataset is generated offline. This sort of data augmentation requires the data to be generated at each training iteration. Figure 3.2 illustrates a few projected views from the CT volume using additive projection.

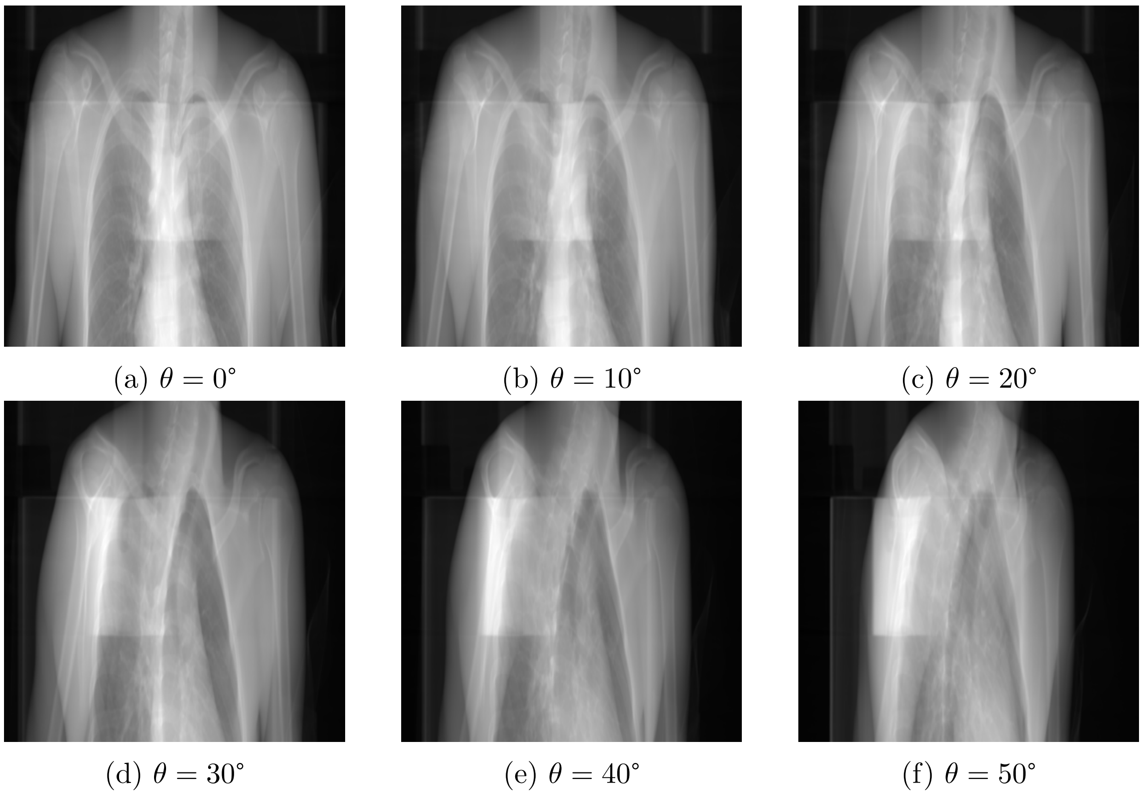

(a) $\theta = 0°$ (b) $\theta = 10°$ (c) $\theta = 20°$
(d) $\theta = 30°$ (e) $\theta = 40°$ (f) $\theta = 50°$

Figure 3.2: Views using additive projections.

In an online training setting where a batch of images needs to be generated at each training iteration, a slow data generation method such as additive projection becomes prohibitive. To reduce the data generation time and allow the integration of the data generation into the training pipeline we resort to a more efficient method, namely the Fourier volume rendering (FVR) method.

3.3 Fourier volume rendering

Fourier volume rendering [17, 57] (also known as frequency-domain volume rendering [83]) is a method for DRR generation from a 3D volume. Based on the Fourier projection-slice theorem, FVR generates DRRs by converting a 3D volume into the frequency domain (as opposed to conventional methods, which project the 3D volume directly). To better understand the process of DRR generation from CT volumes with FVR, we present here the Fourier projection-slice theorem.

3.3.1 Fourier projection-slice theorem

Consider the volume defined by a real-valued function $f : (x, y, z) \mapsto f(x, y, z)$. The 3D case of the Fourier projection theorem states that an orthogonal 2D projection of the volume defined by f at an arbitrary angle can be computed by taking the inverse Fourier transform of a central slice (orthogonal to the direction of projection) of the 3D Fourier transform of f.

The orthogonal projection of f onto the xy-plane is

$$p_z(x, y) = \int_{-\infty}^{\infty} f(x, y, z)\, dz. \tag{3.1}$$

The 2D Fourier transform of p_z is defined by:

$$\begin{aligned} P_z(u, v) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} p_z(x, y)\mathrm{e}^{-j(ux+vy)}\, dx\, dy \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \left(\int_{-\infty}^{\infty} f(x, y, z)\, dz\right) \mathrm{e}^{-j(ux+vy)}\, dx\, dy \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x, y, z)\mathrm{e}^{-j(ux+vy)}\, dx\, dy\, dz. \end{aligned} \tag{3.2}$$

And the 3D Fourier transform of f is given by

$$F_3(u, v, w) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x, y, z)\mathrm{e}^{-j(ux+vy+wz)}\, dx\, dy\, dz. \tag{3.3}$$

The slice of $F_3(u, v, w)$ orthogonal to the projection direction and passing through the origin is defined by

$$F_3(u, v, 0) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x, y, z)\mathrm{e}^{-j(ux+vy)}\, dx\, dy\, dz, \tag{3.4}$$

which corresponds to the Fourier transform of $p_z(x, y)$. Therefore, the projection

$p_z(x, y)$ can be rewritten as

$$p_z(x, y) = F_2^{-1}\{F_3(u, v, 0)\}. \tag{3.5}$$

The proof of the Fourier projection-slice theorem in higher dimensions can be established using analogous reasoning.

3.3.2 Interpolation in the frequency domain

Digitally-reconstructed radiograph generation with FVR requires the extraction of a 2D slice of the Fourier transform from the 3D volume. As the Fourier transform of a 3D CT volume is only known for a finite number of sample points on a 3D voxel grid, slices from this 3D Fourier transform volume must be resampled and interpolated using values of nearby sample points. A good interpolation method is therefore critical to ensure an accurate rendering. Interpolation at new sample points can be achieved in two steps. The first step consists of finding a continuous function (interpolated function) that interpolates between the known sample points. The second step consists in sampling the interpolated function at the new sample points [17, 65].

Without loss of generality, we can use projections from a 2D image to illustrate the effect of the interpolation method on the rendering. Consider a 2D image of size 6×6, for example, where pixels are regularly spaced on the 6×6 grid. In order to generate a vertical projection from this image using FVR, the projection slice theorem requires the extraction of a central slice orthogonal to the direction of projection from the 6×6 Fourier transform of the image. In the present case, the slice of interest will be the horizontal line through the center of the image. As can be seen in Figure 3.3, this slice does not go through any pixel locations in the Fourier transform of the image. An interpolation is therefore required to determine the Fourier transform at the new sample points on the projection. As the interpolation is done in the Fourier transform of the 2D image, we need to choose an interpolation method that is most adapted for interpolation in the frequency domain. We investigate a number of interpolation methods: sinc, nearest neighbor, linear, and spline interpolation.

Consider one row of a 2D Fourier transform (FT) of a 2D image. On this row, the FT is only known at discrete pixel locations. These FT values could be considered as a sampled signal $F_s(u)$ from a continuous signal $F(u)$ in the frequency domain at these pixel locations. Let $f(x)$ and $f_s(x)$ be the inverse Fourier transform (IFT) of $F(u)$ and $F_s(u)$ respectively. Suppose that the samples are uniformly spaced with

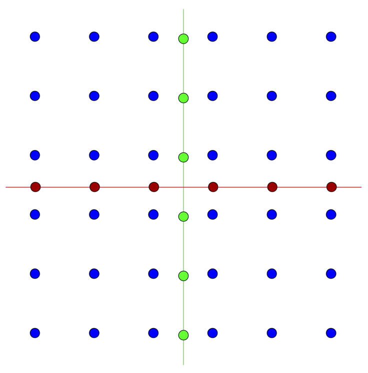


Figure 3.3: Image represented by a 6 × 6 grid of pixels (blue dots) with its vertical projection (red dots), and the horizontal projection (green dots). The red line represents the central line orthogonal to the vertical direction of projection, and the red dots represent the re-sampled pixel locations after interpolation. Similarly, the green line represents the central line orthogonal to the horizontal direction of projection, and the green dots represent the re-sampled pixel locations after interpolation.

sampling frequency $\frac{1}{T}$. Then the FT of the sample signal is defined by

$$F_s(u) = F(u) \cdot \sum_{n=-\infty}^{\infty} \delta\left(u - \frac{n}{T}\right). \tag{3.6}$$

The signal in the spatial domain can be recovered by taking the inverse Fourier transform of Equation 3.6 as follows:

$$\begin{aligned} f_s(x) &= \mathcal{F}^{-1}(F_s(u)) \\ &= \mathcal{F}^{-1}\left(F(u) \cdot \sum_{n=-\infty}^{\infty} \delta\left(u - \frac{n}{T}\right)\right) \\ &= \mathcal{F}^{-1}(F(u)) * \mathcal{F}^{-1}\left(\sum_{n=-\infty}^{\infty} \delta\left(u - \frac{n}{T}\right)\right) \\ &= f(x) * T \sum_{n=-\infty}^{\infty} \delta(x - nT) \\ &= T \sum_{n=-\infty}^{\infty} f(x) * \delta(x - nT) \\ &= T \sum_{n=-\infty}^{\infty} f(x - nT). \end{aligned} \tag{3.7}$$

Thus the reconstructed signal $f_s(x)$ is made up of infinite replicas of the original signal $f(x)$ with replication period T. A high sampling rate is necessary to avoid overlapping of the replicas and aliasing in the reconstructed signal. To reduce the effect of the replicas on the reconstructed signal $f_s(x)$, we need to multiply $f_s(x)$ by a function $i(x)$ that suppresses the replicas as much as possible. In the frequency domain, this is equivalent to convolving the sampled function $F_s(u)$ with $I(u)$, the Fourier transform of $i(x)$. The function $I(u)$ is the interpolation function in the frequency domain. The quality of the interpolation depends on the type of interpolation function used. We investigate the most common type of interpolations used for the frequency domain.

Sinc interpolation

For sinc interpolation the interpolating function is defined by $I(u) = \text{sinc}(Tu)$, so the interpolated continuous function is

$$G_s(u) = F_s(u) * I(u) = F_s(u) * \text{sinc}(Tu). \tag{3.8}$$

The inverse Fourier transform of this interpolated function is defined by

$$\begin{aligned} g_s(x) &= \mathcal{F}^{-1}(F_s(u)) \cdot \mathcal{F}^{-1}(\text{sinc}(Tu)) \\ &= T \sum_{n=-\infty}^{\infty} f(x - nT) \cdot \frac{1}{T}\text{rect}\left(\frac{x}{T}\right) \\ &= f(x). \end{aligned} \tag{3.9}$$

Thus, the replicas observed in Equation 3.7 are effectively removed. This makes sinc interpolation the ideal method for the interpolation of signals in the Fourier domain. However, sinc interpolation is often computationally prohibitive due to its infinite extent. In practice, simpler and more efficient interpolation methods are preferred although they are less accurate than sinc interpolation.

Nearest neighbor interpolation

In contrast to sinc interpolation, which requires contributions from all sample points to interpolate at a new sampling point, nearest neighbor interpolation only uses the contribution from the closest sample to the point where the interpolation is needed. This interpolation method is very efficient but its accuracy is very limited.

Linear interpolation

Linear interpolation in 1D requires contribution from the two nearest samples to compute the interpolated value at the new sample point. This interpolation method is generally more accurate than simple nearest neighbor interpolation, although this interpolation method is not perfect. For applications where linear interpolation is not good enough, polynomial interpolation is often used.

Spline interpolation

Cubic spline interpolation is a polynomial interpolation method that is more accurate than simple linear interpolation. This interpolation method suppresses the replicas of the reconstructed signal more effectively than the methods discussed previously.

3.3.3 Zero-padding in the spatial domain

The resampling in the frequency domain causes replication of the reconstructed signal in the spatial domain. When these replicas overlap, aliasing artifacts occur in the reconstructed signal. In images, these appear as “ghosting” artifacts where replicas of objects are observed at the edges of the image. To minimize the effect of potential ghosting artifacts (and also to facilitate the interpolation in the frequency domain), the 3D volume is zero-padded in the spatial domain before taking the Fourier transform. This leads to a finer sample spacing (so a higher sampling frequency) in the spectral domain, which in turn widens the separation between replicas in the spatial domain to avoid overlaps in the reconstructed image.

3.3.4 Computational complexity of the FVR

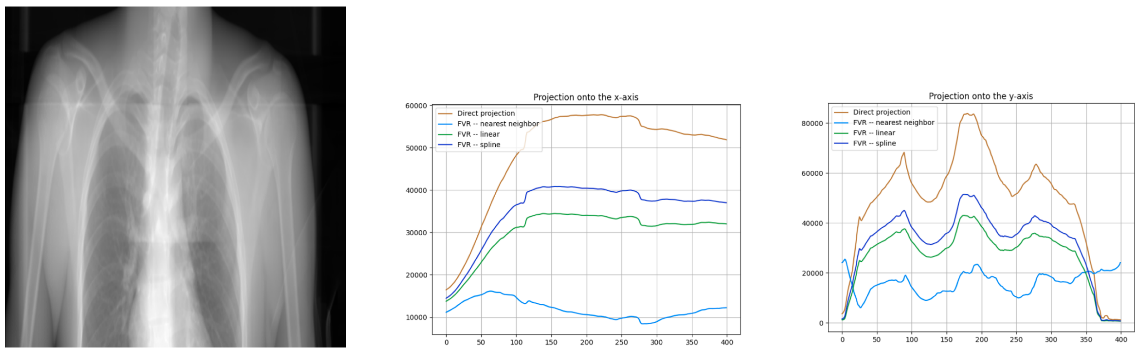
Using the FVR technique, the 3D FFT of the CT volume is first computed. The DRR from a particular viewpoint is obtained by extracting a slice through the center of the 3D FFT of the CT that is orthogonal to the projection direction. The DRR is then obtained by computing the inverse Fourier transform of the extracted slice. Thus the generation of a DRR for each view of a $N \times N \times N$ volume requires the sampling of points on a $N \times N$ slice passing through the center of the volume, followed by the IFT of the sampled points. This results in a time of complexity of $O(N^2 \log N)$ for the FVR method, as opposed to the $O(N^3)$ of additive projection. This more efficient DRR generation method allows us to train viewpoint estimation models in an online fashion whereby a batch of training images is generated from the CT volume at each training iteration. This allows us to implement several data augmentation strategies from the 3D volume directly, with small random out-of-plane rotations and translations.

3.4 Experiments and results

We use the Fourier volume rendering method as an alternative to the additive projection method for DRR generation from CT scans. While FVR is proven to be more efficient than additive projection, we must ensure that the DRRs generated using FVR are qualitatively as close as possible to those generated using AP. To this end, we investigate the factors influencing the quality of the DRR generated by FVR. We first focus on the effect of the interpolation method used on the accuracy of the rendering. We then investigate how zero-padding in the spatial domain could be used to improve the quality of the DRR generated. Although the DRRs are generated from 3D volumes we test the accuracy of the FVR in the 2D case, where the projection from each direction is a one-dimensional signal. This is done in order to compare and better visualize the projections from the FVR and the AP methods.

3.4.1 Effect of interpolation methods

We experimente with the nearest neighbor, linear, and cubic spline interpolation methods. An image of size 400 × 400 is projected to a signal vector of length 400 using the projection from the AP method as reference. The FVR projection using each of the three interpolation methods is compared to the reference projection. As expected from the theory, the cubic spline interpolation method gives the closest projection to the reference projection, followed by linear and nearest neighbor interpolations respectively. The effect of interpolation methods on rendering performance is illustrated in Figure 3.4. As can be seen in Figure 3.4, FVR based on cubic splines reconstructs the projection quite fairly, although there is still a gap between the reference projection and the projection obtained with FVR. We attempt to reduce this gap using the zero-padding technique.



(a) 2D image (b) Horizontal projection (c) Vertical projection

Figure 3.4: Interpolation methods for FVR.

3.4.2 Effect of zero-padding

In order to improve the FVR projection accuracy, we zero-pad the image in the spatial domain before taking the Fourier transform. We experimented with three padding sizes. The original image of size 400×400 (with no zero-padding) is used as a reference. The original image is then zero-padded to get images of size 800×800 and 1200×1200. In each case, the projection vector is obtained by extracting the center of the padded projected vector. The extracted projection is then compared to the reference projection. The zero-padding effect on rendering accuracy is illustrated in Figure 3.6. This figure shows that a bigger padding size leads to more accurate projections. However, as illustrated in Figure 3.5, the computational cost increases with the size of the padded image. A trade-off between projection accuracy and computational efficiency is therefore required. To keep the computational cost low while having a reasonable projection quality, we limit the size of the padded images to 800×800 to generate our dataset for viewpoint estimation using FVR.

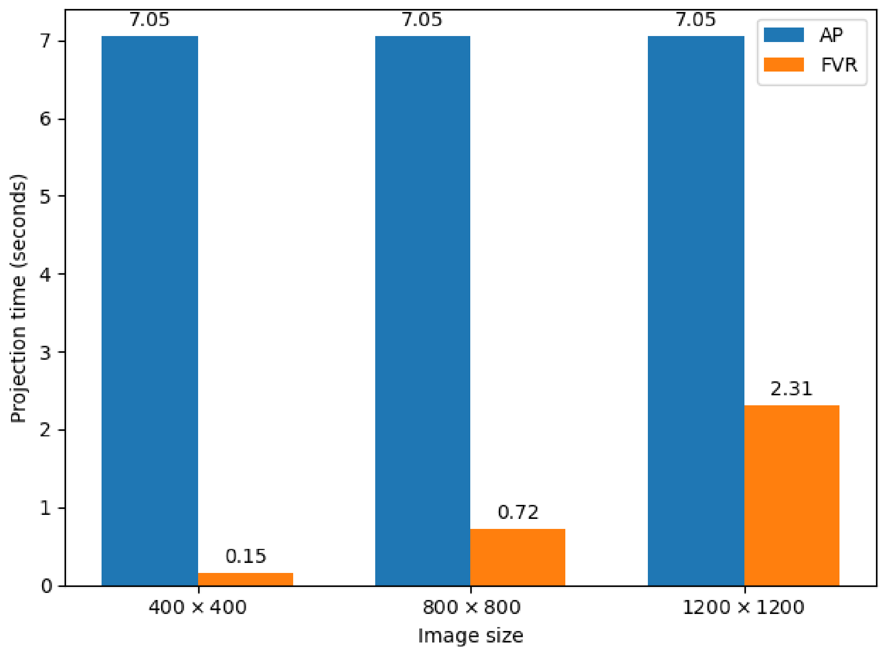
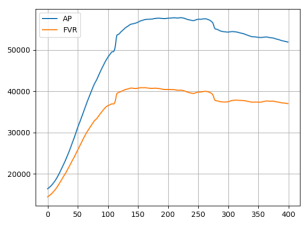


Figure 3.5: Average projection time per image for the AP and FVR methods.

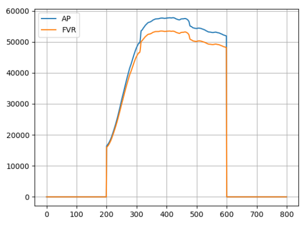
Figure 3.7 gives a qualitative comparison of the FVR and the AP methods.

3.5 Other DRR generation methods

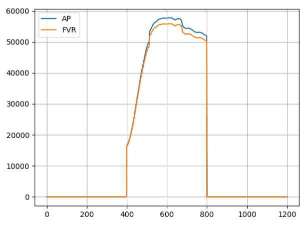
The additive projection (AP) and Fourier volume rendering (FVR) methods use parallel projection to produce DRRs. However, the X-ray image generation process could be better simulated using perspective or fan-beam projection. In this section



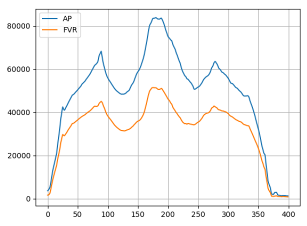
(a) x-axis (no padding)



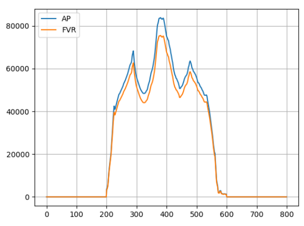
(b) x-axis (2X padding)



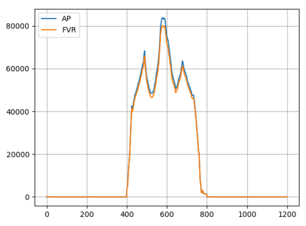
(c) x-axis (3X padding)



(d) y-axis (no padding)

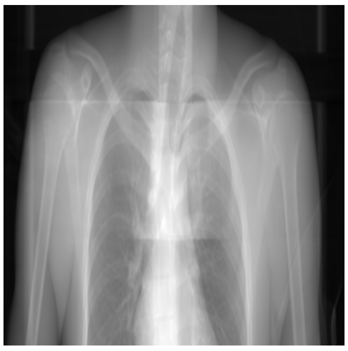


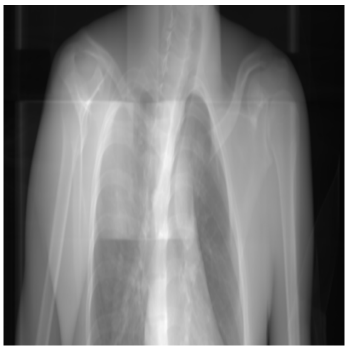
(e) y-axis (2X padding)

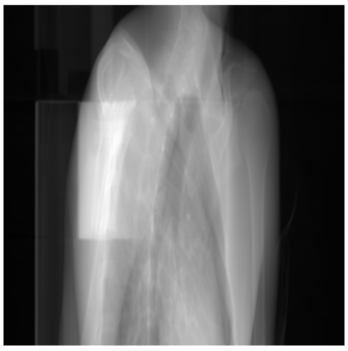


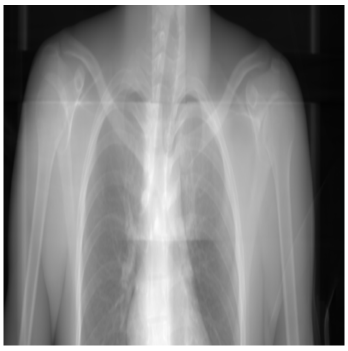
(f) y-axis(3X padding).

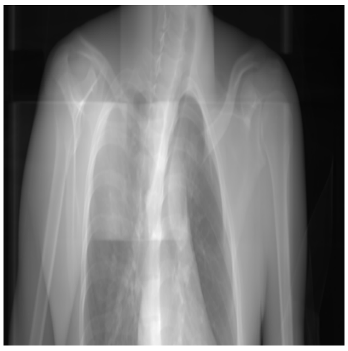
Figure 3.6: Effect of zero-padding on FVR.


(a) $\theta = 0°$ (AP)


(b) $\theta = 20°$ (AP)


(c) $\theta = 50°$ (AP)


(d) $\theta = 0°$ (FVR)


(e) $\theta = 20°$ (FVR)

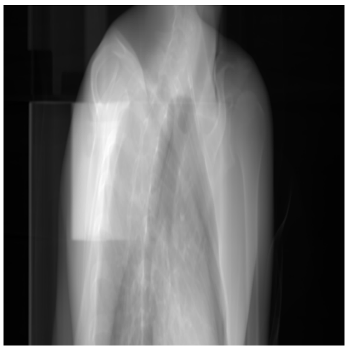

(f) $\theta = 50°$ (FVR)

Figure 3.7: Comparison of additive projections and FVR projections. The first row corresponds to the views generated using AP, while the second represents the views generated using FVR. The images in each column correspond to the same viewpoint.

we present other DRR generation methods that can be used as alternatives to AP and FVR.

3.5.1 Attenuation fields DRR

Attenuation fields DRR (AF-DRR) by Russakoff et al. [73] is a modification of the light field rendering method [47]. In light field rendering, each ray emanating from the source is parametrized by its intersection with two planes representing the focal plane and the image plane. The focal plane and the image plane are parametrized by their coordinate systems (u, v) and (s, t), respectively. Thus each ray is represented as a point in a 4-dimensional space: $p_i = (u_i, v_i, s_i, t_i)$, as illustrated in Figure 3.8. The set of all rays connecting a point in the focal plane to another point in the image

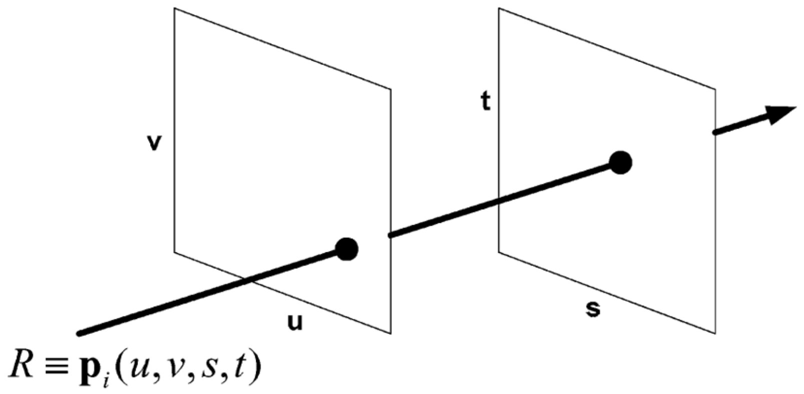


Figure 3.8: Ray parametrization in light field rendering [73].

plane constitutes a light slab. An image from any viewpoint within this light slab can be determined by considering the rays from the viewpoint to each pixel location in the image. Where no rays exist within the light slab between the viewpoint and a particular pixel location, an estimated ray is found using nearest neighbor interpolation.

In attenuation fields rendering, a third plane called the virtual image plane is introduced between the focal and the image plane of light field rendering. In addition, each ray $R_{\mathbf{p}_i}$ is associated with an attenuation value $U_{\mathbf{p}_i}$ defined by

$$U_{\mathbf{p}_i} = \int_{R_{\mathbf{p}_i}} \mu(s) ds \approx \sum_{\mathbf{x}_j \epsilon R_{\mathbf{p}_i}} \mu(\mathbf{x}_j) \Delta \mathbf{x}, \tag{3.10}$$

where each $\mathbf{x}_j$ is a voxel along the ray, $\mu(\mathbf{x}_j)$ is the linear attenuation coefficient corresponding to $\mathbf{x}_j$, and $\Delta \mathbf{x}$ is the distance between voxels along the ray. Figure 3.9 illustrates the AF-DRR process.

The computation time of AF-DRR is reduced compared to traditional ray-tracing methods since attenuation from new rays can be determined via lookup or nearest neighbor interpolation. However, the dense sampling of the light slab and the precomputation of the attenuation coefficients requires substantial memory storage. The memory requirement of the method is reduced by compressing the data using

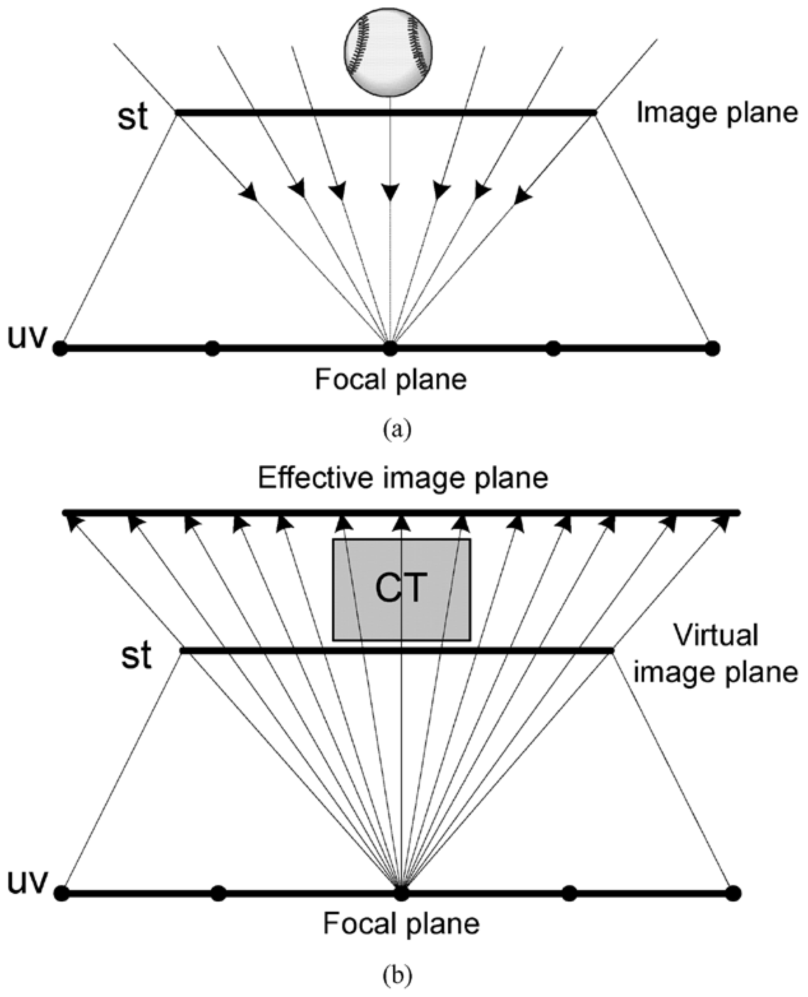


Figure 3.9: Overview of AF-DRR X-ray image simulation process. (a) Light field rendering. (b) Attenuation field for DRR generation [73].

vector quantization. AF-DRR offers faster DRR generation with comparable DRR quality to the conventional ray-tracing method.

3.5.2 DiffDRR

DiffDRR is a DRR generation method proposed by Gopalakrishnan and Golland [24]. Similar to common DRR generation methods, DiffDRR uses Siddon's ray-tracing technique [76]. According to Siddon's method, X-rays emitted from a source $\mathbf{s} \in \mathbb{R}^3$ pass through a CT volume and hit a detector plane. Thus, each ray R connecting the source $\mathbf{s}$ to a projection point $\mathbf{p} \in \mathbb{R}^3$ on the detector plane can be parametrized by $R(\alpha) = \mathbf{s} - \alpha(\mathbf{p} - \mathbf{s})$, with $\alpha \in [0, 1]$. As the ray R passes through the CT volume, its energy is attenuated. The total energy attenuation of the X-ray traveling from $\mathbf{s}$ to $\mathbf{p}$ is given by

$$E(R) \approx \|\mathbf{p} - \mathbf{s}\|_2 \sum_{m=1}^{M-1} (\alpha_{m+1} - \alpha_m) \mathbf{V} \left[\mathbf{s} + \frac{\alpha_{m+1} - \alpha_m}{2} (\mathbf{p} - \mathbf{s}) \right], \tag{3.11}$$

where α_m and α_{m+1} correspond to consecutive intersection points of the ray R with two adjacent planes on the cube (or rectangular parallelepiped) representing a voxel, $\mathbf{V}$ is the CT volume, and M is the number of intersections of the ray with the CT volume. The energy attenuation $E(R)$ corresponds to the intensity value at pixel location $\mathbf{p}$ in the projected DRR. Figure 3.10 illustrates the DRR generation process using Siddon's method.

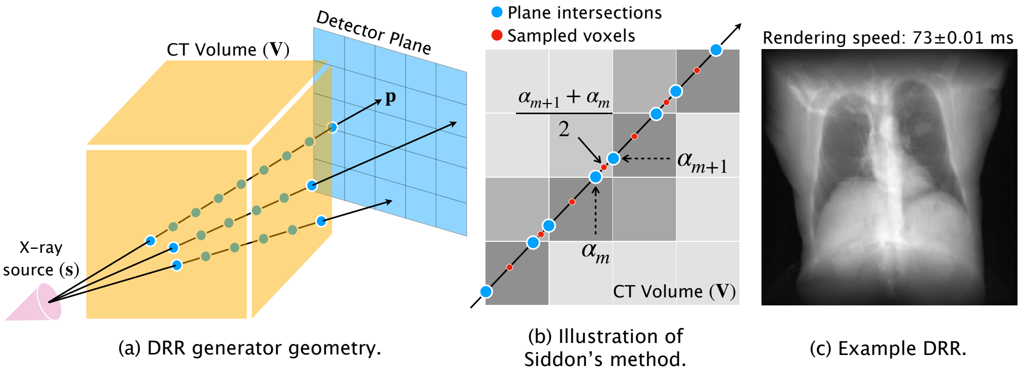


Figure 3.10: Illustration of Siddon's method for DRR generation [24].

DiffDRR vectorizes Siddon's ray-tracing operations for fast computation on graphical processing units (GPU). This method can generate DRRs and their derivatives with respect to the image geometry parameters for applications that require automatic differentiation such as slice-to-volume registration.

3.5.3 RealDRR

RealDRR, proposed by Dhont et al. [15], combines Siddon's ray-tracing method with deep learning to generate realistic DRRs. This X-ray simulation framework is made of two successive stages: projection and translation. The projection step uses ray-tracing to create a 2D image from the 3D volume by aggregating voxel intensity values along a ray from the X-ray source to the 2D image projection plane. This results in a raw DRR which is further processed in the image-to-image translation step. The image translation is done using a conditional generative adversarial network (cGAN), using a U-net convolutional neural network architecture for both its generator and its discriminator. The cGAN is trained using the generative adversarial network (GAN) loss with L1 regularization as a cost function. Figure 3.11 shows an overview of the RealDRR framework.

The qualitative results demonstrate that the RealDRR framework yields realistic simulated radiographs, especially in the intra-patient case where training and testing data come from the same patient. However, a drop in performance is observed when the framework is tested on data of the same anatomical region coming from

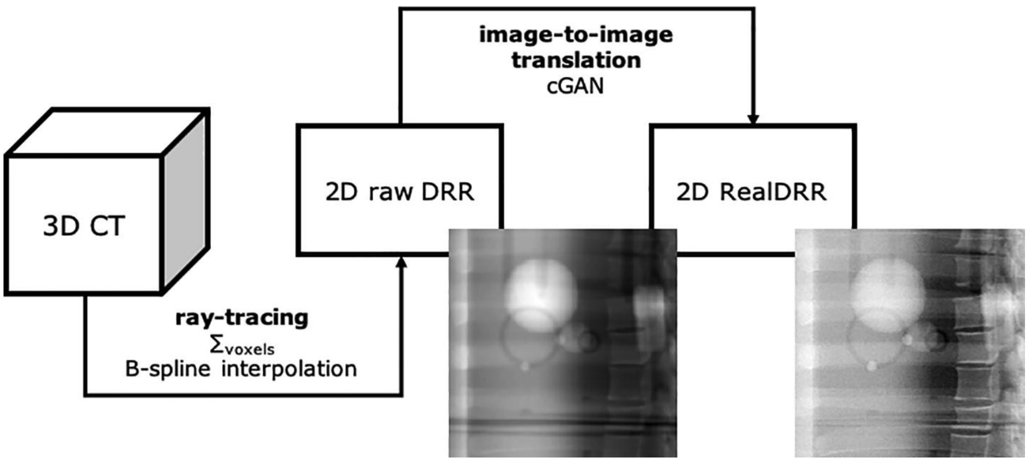


Figure 3.11: Schematic overview of RealDRR X-ray image simulation framework [15].

a different patient than the one used during training. This suggests that RealDRR has a limited inter-patient generalization capability. The inter-patient generalization could further degrade when data from different anatomical regions are used.

3.5.4 DeepDRR

DeepDRR, proposed by Unberath et al. [85, 86], is a method for DRR generation using deep learning models. The method simulates a C-arm X-ray data generation process using four steps: material decomposition, projection of the 3D volume to a 2D plane using ray-tracing, scatter estimation, and noise injection. An overview of the DeepDRR X-ray simulation process is presented in Figure 3.12.

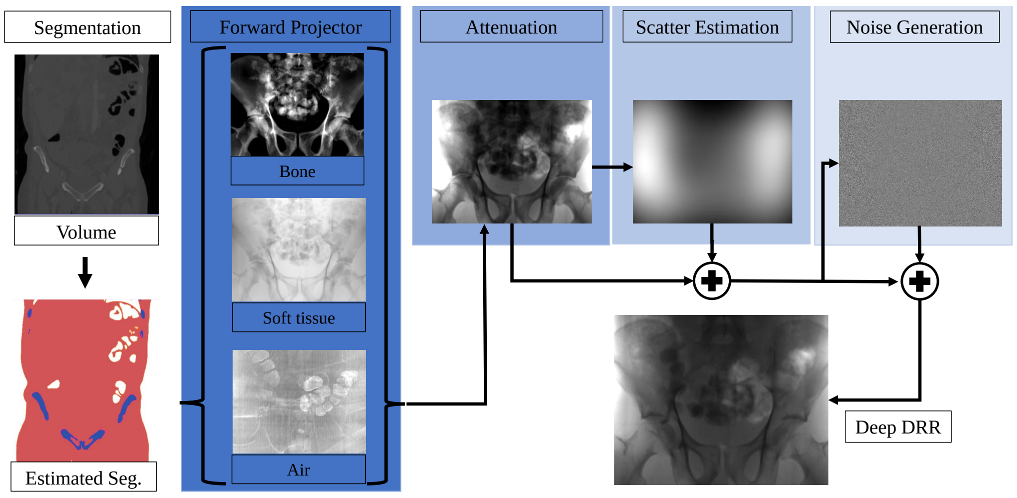


Figure 3.12: Schematic overview of DeepDRR X-ray image simulation framework [86].

The material decomposition step segments the CT volume according to the different materials present in the volume: bone, air, or soft tissue. This is achieved by training a deep convolutional neural network for semantic segmentation with a V-net [63] architecture. Once trained, the segmentation network can classify each of the voxels in the 3D volume into one of the three material classes. The 3D volume is projected onto a 2D detector using a projection matrix and the energy attenuation map is calculated using ray-tracing. The energy attenuation is computed by connecting a virtual ray from the source to a pixel location on the detector. As the ray passes through the 3D volume, the X-ray energy attenuation at the detector is calculated by taking into account the contribution of each material encountered along the ray. A convolutional neural network is trained to estimate the Rayleigh scatter. Finally, a Poisson noise model accounts for the noise due to pixel crosstalk on the detector plane. In addition, the electronic noise is modeled using additive Gaussian noise.

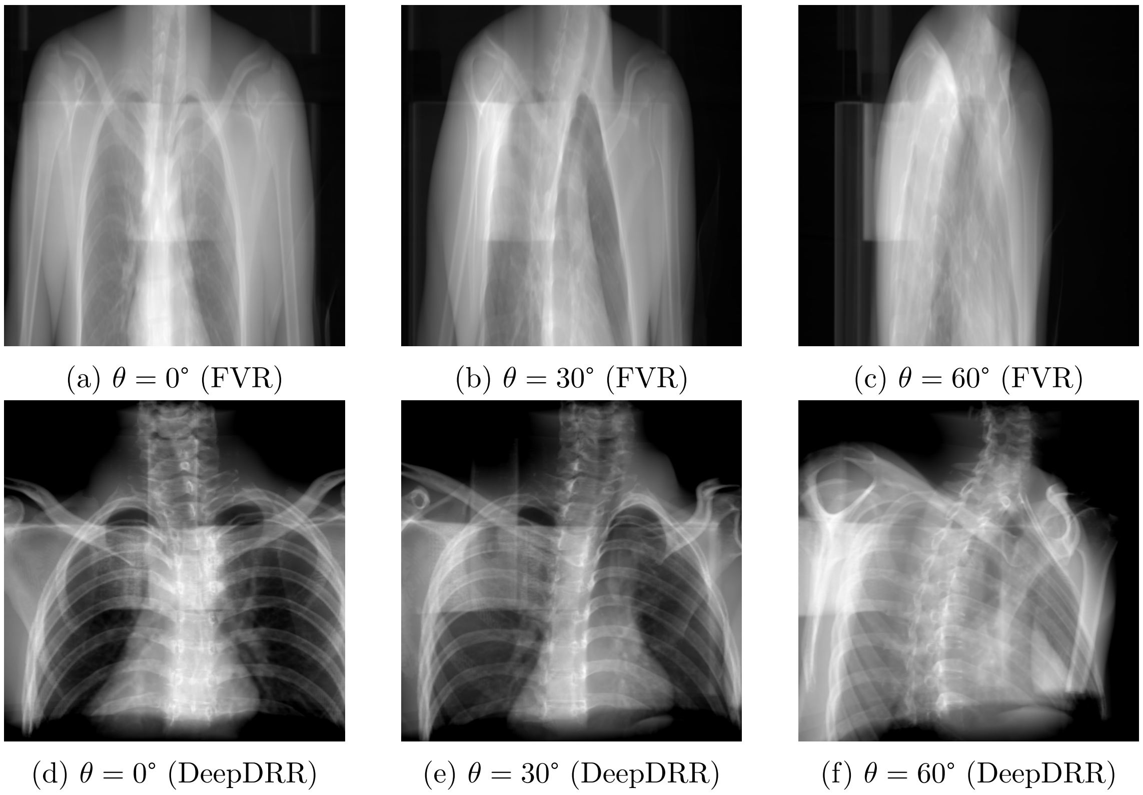
Using the Python implementation of DeepDRR[1], we generate a sample DRRs for comparison with those generated using FVR. To obtain realistic DRRs with size 512×512, we use the C-arm with a detector at a resolution of 2048×2048. The distance from the source to the detector is 800 mm. The C-arm rotates around the 3D volume along the longitudinal axis, and a DRR is generated at each position. The projector is set with the energy spectrum $60KV_AL35$ and a photon count of 10^5. The C-arm moves around the 3D volume in the range $[0°, 360°]$ with 1-degree increments, and a DRR is generated at each step. Figure 3.13 shows a sample of the X-ray images simulated with DeepDRR.

Compared to the DRRs generated using additive projection (AP) and Fourier volume rendering (FVR), the DRRs generated using DeepDRR look more realistic. Using a partition of an A100 Ampere graphics card with 20GB of memory, the computation of one DRR took 7 seconds on average. This computation time is similar to that of the AP method. This makes the DeepDRR method significantly slower than the FVR method.

3.6 Summary

In this chapter we presented two methods used to generate a dataset for viewpoint estimation from a raw dataset of full-body 3D CT scans, the additive method and the Fourier volume rendering method for DRR generation are presented. We exposed various parameters that influence the quality of the images generated using FVR. Using these methods we were able to generate a dataset with accurate viewpoint labels that can be used to train deep learning models for viewpoint estimation. Moreover, we presented other DRR generation methods that can be used to generate the dataset for viewpoint estimation.

[1]https://github.com/arcadelab/deepdrr.


(a) $\theta = 0°$ (FVR) (b) $\theta = 30°$ (FVR) (c) $\theta = 60°$ (FVR)

(d) $\theta = 0°$ (DeepDRR) (e) $\theta = 30°$ (DeepDRR) (f) $\theta = 60°$ (DeepDRR)

Figure 3.13: Sample of X-ray images simulated using FVR (first row) and DeepDRR (second row). The structure of the hard tissues can be clearly seen in the images generated using DeepDRR compared to those generated using FVR.

Chapter 4

One-dimensional viewpoint estimation

The view that one has of a 3D object depends on the position of the observer with respect to the object. Similarly, when projecting 2D views from a 3D volume, the resulting image depends on the direction of the projection. Thus, rotating a CT scan volume and taking projections would result in different X-ray images depending on the angle of rotation around the 3D volume. In three-dimensional space, rotations around a volume can be parametrized by three Euler angles (rotations around each axis of the Cartesian coordinate system in 3D). Each triplet of Euler angles then corresponds to a single projected view from the 3D volume.

Viewpoint estimation is essentially the inverse problem of projection. Given an image projected from a 3D volume, viewpoint estimation aims at recovering the rotation angles that were used to project the image from a 3D volume. This task is much more challenging than the simple projection from 3D to 2D. We first focus on the problem of one-dimensional viewpoint estimation where we are only interested in finding one rotation angle. This is a special case of the general 3D case. Given a 2D projection, we estimate the rotation angle around the longitudinal axis of the body (z-axis) from which a projection of the volume would result in the 2D image. We later extend the method developed for one-dimensional viewpoint estimation to full 3D viewpoint estimation in Chapter 6.

In this chapter we present the different methods proposed for one-dimensional viewpoint estimation. In Section 4.1, we present the problem of one-dimensional viewpoint estimation. The viewpoint space representation for classification and regression is also defined. Moreover, the different constraints associated with the viewpoint estimation task are also discussed. Considering the continuous nature of the viewpoint space, we present a regression approach for viewpoint estimation in Section 4.2. Since the regression model was not very effective for the problem, we resort to classification methods. In Section 4.3, we present the nearest neighbor method for viewpoint estimation. This baseline approach for classification models proves to be more effective than the regression approach. In Section 4.4 we present a CNN-based classification method, which confirms the effectiveness of classification methods over regression-based approaches. We conclude the chapter with a summary in Section 4.5.

4.1 Problem of viewpoint estimation

The task of viewpoint estimation aims at recovering the location from which a particular 2D projection of an object originates in a 3D scene. This typically requires the estimation of 3D rotation angles around the 3D object. These 3D rotations can be parametrized using several representations such as Euler angles, axis-angle, and quaternions. Among these various parametrizations, the Euler angle representation is the most used in the literature due to its simplicity.

In this experiment, we use the Euler angle parametrization of 3D rotations. Since the three rotations of the Euler angle are independent, we focus on developing methods to estimate one Euler angle. Thus we will be solving the problem of 1D viewpoint estimation. The developed methods can then be extended to full 3D viewpoint estimation by applying the same method to the other two angles. In order to present the viewpoint estimation problem precisely, we need to define the representation of the viewpoint space used and the constraints associated with the problem.

4.1.1 Viewpoint space representation

The viewpoint space can be parametrized as a continuous angle around a 3D object or a discrete partition of angles around the object. The choice of parametrization mainly depends on the requirements for the method used.

Continuous viewpoint space

The viewpoint space around a 3D object is inherently continuous. For one-dimensional viewpoint estimation, we can parametrize the viewpoint space by a rotation angle $\theta \in [0, 2\pi]$ covering the circumference of the object. The continuous representation is used for regression models where the outputs are continuous. However, this type of representation is not adapted to classification models which require a discrete and finite number of classes.

Discrete viewpoint space

A classification model requires a finite set of outputs. A continuous parametrization is therefore not suited to this type of model. We use a discrete representation of the viewpoint space instead. In order to discretize the viewpoint space, we divide the rotation angles into bins. Each bin can then be used as a class for a classification model. Finer bins result in more accurate classification and more precise viewpoint estimation. However, this comes at the price of more complex classification models which can be difficult to train. In this project, we use bins of one-degree intervals, leading to a total of 360 bins to cover the full angle range ($\theta \in [0, 2\pi]$) around the 3D object. This allows us to have a precise viewpoint estimate. We use transfer learning with effective data augmentation techniques to train the classification models.

4.1.2 Geometric constraints

The particular geometry of the viewpoint space imposes some constraints on the models developed for the task. Given that the viewpoint space is circular, there is more similarity between nearby classification bins than between bins that are further apart. Also, a bigger absolute difference between viewpoints does not necessarily imply dissimilar viewpoints. For example, $0°$ and $359°$ have the highest absolute distance, but they correspond to very close viewpoints on the circular viewpoint space. Therefore our predictive models (classification or regression models) must be built to take these factors into account.

4.1.3 Dataset for viewpoint estimation

We use the additive projection method to generate a dataset for viewpoint estimation. This dataset generated offline from a 3D CT scan volume is made of 2D DRRs with accurate viewpoints labels. Each 2D view is obtained by rotating a 3D volume around the z-axis (rotation axis), followed by an orthogonal projection of the volume in the xz-plane (projection plane) for different values of the azimuth angle. Pixel intensities of the resulting DRRs are obtained by summing voxel intensities along parallel rays from the rotated volume to the projection plane. The views are generated at 1-degree intervals, resulting in 360 different views. The DRR generated from each viewpoint has a size of 512×512. In order to acquire more data, a central patch of size 400×400 from each DRR is translated horizontally and vertically. Thus for each viewpoint, 80 more images are generated in addition to the original image by translating a sliding window of size 400×400 in the range [-20, 20 pixels] and extracting an image patch at each location. This results in a training data of 29,160 images of size 400×400 generated from one CT scan. To assess the performance of the model during training, we generated validation data from another CT scan. The test data is generated from a third CT volume to evaluate the generalization power of the trained model.

4.2 Regression-based viewpoint estimation

Regression models predict the relationship between an independent variable and one or more dependent variables. In machine learning, regression models are used to predict a continuous output quantity given a set of input data. As the viewpoint space around a 3D object is intrinsically continuous, a regression model seems to be the natural choice to estimate the viewpoint of a 3D object. Regression models are generally categorized as either linear or non-linear. While linear regression models are simple to implement, they are limited when the relationship between dependent and independent variables is not linear. For these types of problems, a non-linear regression is required. In this project, we implement non-linear regression models for viewpoint estimation using a convolutional neural network (CNN).

4.2.1 CNN model for viewpoint regression

Deep learning methods, notably CNN, have been particularly successful at solving classification problems in computer vision. The remarkable success of CNN models in image classification has motivated their extension to other tasks such as object detection, image segmentation, or image registration. Here we turn a CNN model, initially developed for a classification, into a regression model by replacing the last layer of the CNN (the classification layer) with a regression block that outputs a continuous quantity. As CNN feature maps are passed through non-linear activation functions, the resulting regression models are non-linear.

Mathematical formulation

Consider a labeled dataset $\mathcal{D} = \{(x_i, y_i) : i \in [1, N]\}$, where N is the size of the dataset. Each x_i represents an image, and each y_i is the corresponding label. Let $\mathcal{X}$ be the set of all images in the dataset, and $\mathcal{Y}$ be the subset of $\mathbb{R}$ corresponding to the possible values of the viewpoint. Suppose each $x_i \in \mathcal{X}$ is related to the corresponding $y_i \in \mathcal{Y}$ by some unknown labeling function $f : x_i \mapsto y_i = f(x_i)$. The goal of the CNN model is to assign a label to each image in a test dataset for which a ground truth label is not available. In the case of viewpoint estimation, we want to know the corresponding viewpoint y_j to any image x_j that is not in the training set $\mathcal{D}$. Since the function f is unknown, it cannot be used to find y_j by substituting x_j into f. We therefore require an approximation of f in order to estimate the viewpoint y_j for any image x_j that is not in the training set.

A regression model approximates the function f with another mapping $\hat{f} : x_j \mapsto y_j = \hat{f}(x_j) \approx f(x_j)$. For any image x_j in the test set, the viewpoint can be estimated as $y_j = \hat{f}(x_j)$. A CNN-based regression model approximates $\hat{f}$ by learning from the training dataset $\mathcal{D}$. The model learns from the training dataset by adjusting its parameters such that for each training image x_i, the predicted $\hat{y}_i = \hat{f}(x_i)$ is as close as possible to $y_i = f(x_i)$. The closeness of each predicted $\hat{f}(x_i)$ to the ground truth y_i is measured by a cost function, which is typically a mean square error (MSE) for regression problems. The adjustment of the model parameters is done by minimizing the cost function using some optimization method, usually based on gradient descent. The function learned by the CNN model can be defined by $\hat{f}_\Theta : x_i \mapsto \hat{f}_\Theta(x_i)$, where Θ represents the state of the parameters at any given training iteration.

Model architecture

Convolutional neural network models generally involve millions of trainable parameters. Training such models from scratch requires a large amount of data. When the amount of training data is insufficient, the model can easily overfit the training data. Given the limited size of our viewpoint dataset, training a CNN model from scratch would therefore be impractical as this would certainly lead to overfitting. Instead, we use transfer learning where a model trained on one dataset is fine-tuned on another dataset to solve a different problem. This approach usually requires less training data compared to training a model from scratch. Thus transfer learning is more adapted to applications where the size of the dataset is not sufficient to train a model from scratch.

For the CNN-based regression, we use the Inception-V3 [81] model which was initially trained on ImageNet [13] for natural image classification. The last layer of the pre-trained model is removed and replaced by a fully-connected layer whose weights are randomly initialized. This fully-connected layer is followed by a single neuron layer that gives the output angle for each input image. Figure 4.1 shows a diagram of the regression model.

Cost function

The cost function, also called loss, is a function that evaluates the closeness of the predicted label $\hat{y}_i = \hat{f}(x_i)$ to the ground truth label $y_i = f(x_i)$. For regression tasks,

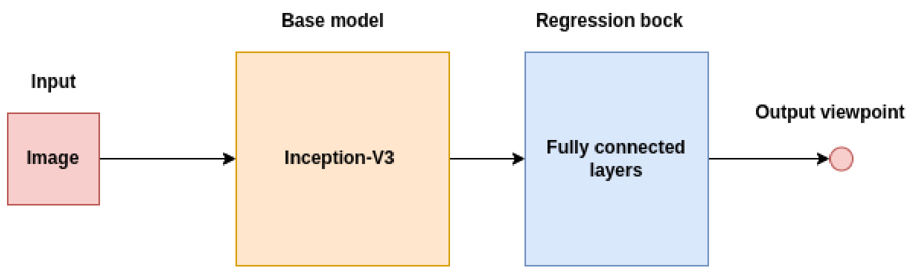


Figure 4.1: Block diagram of the CNN-based regression model for viewpoint estimation.

a typical cost function is the mean square error (MSE) defined by

$$\mathcal{L}_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2. \tag{4.1}$$

In essence, the MSE cost function "guides" the model to make predictions as close as possible to the ground truth, when the model parameters are chosen such that the loss is minimal. However, the MSE does not take into account the geometrical attributes of the viewpoint space. In effect, predictions deemed very distant by the MSE loss could in fact correspond to very close viewpoints. For example, a prediction of $\hat{y} = 359°$ when the ground truth is $y = 0°$ would suffer a very high penalty by the MSE cost function while this prediction is actually very close to the ground truth viewpoint. Thus nearby views might score a higher MSE cost than far away views since the viewpoint space is circular. A cost function capable of dealing with the geometric nature of the problem is therefore required for a successful viewpoint regression model.

Let θ and $\hat{\theta}$ be the target and the predicted viewpoint respectively. Let us associate θ and $\hat{\theta}$ with two points M_1 and M_2 (respectively) on a unit circle. We can then derive a cost function from the square of the Euclidean distance between M_1 and M_2.

Let $M_1 = (\cos\theta, \sin\theta)$ and $M_2 = (\cos\hat{\theta}, \sin\hat{\theta})$. The square of the Euclidean distance between M_1 and M_2 is defined by

$$(M_1M_2)^2 = (\cos\theta - \cos\hat{\theta})^2 + (\sin\theta - \sin\hat{\theta})^2, \tag{4.2}$$

which reduces to

$$(M_1M_2)^2 = 2(1 - \cos(\theta - \hat{\theta})). \tag{4.3}$$

Our regression cost function is derived from the Euclidean distance between M_1 and

M_2:

$$L_{REG}(\theta, \hat{\theta}) = 1 - \cos(\theta - \hat{\theta}). \tag{4.4}$$

It is worth noting that the cost function defined in Equation (4.4) is one among many loss functions that could be used to train a regression-based viewpoint estimation model. Here we opt for this cost function as it captures well the circular distribution of the viewpoint space. This cost function is minimized using the stochastic gradient descent (SGD) optimizer.

4.2.2 Model training

The objective of the training process is to find the optimal state of the model's parameters that minimizes the cost function. When the objective function is convex, there exists an absolute minimum that can be obtained without difficulty. However, due to the use of non-linear activation functions and the depth of CNN models the cost function is non-linear, which leads to a non-convex optimization problem. In such a case, the convergence to the global minimum is not guaranteed, and one generally aims at finding the best local minimum. The optimization requires more subtle methods to avoid getting stuck in a bad local minimum. There are several methods that are used to optimize the cost function of a CNN model. These methods are typically based on the gradient descent algorithm.

We use the stochastic gradient descent (SGD) algorithm to optimize the cost function defined in Equation (4.4). The SGD algorithm evaluates the gradient of the cost function with respect to each trainable parameter in the CNN model. At each training iteration, the state of the model parameters is updated to minimize the loss. The model parameters are updated in the negative direction of the gradient as described by

$$W_{i+1} = W_i - \lambda \nabla L_{REG}, \tag{4.5}$$

where the constant λ is the learning rate, W_i is the weights at iteration i, and L_{REG} is the loss at iteration i. To avoid any brusque alteration of the pre-trained weights, the initial learning is reduced progressively during training using exponential decay.

At each training iteration, the model predicts the viewpoints of a batch of images. The correctness of these predictions is measured by the cost function. The SGD algorithm is then used to update the model parameters according to their contribution to the total loss. After each training epoch, the model is evaluated on the validation set. We stop training when the performance on the validation set stops improving. The best-performing model on the validation set is restored and evaluated on the test set.

4.2.3 Evaluation metrics

We use the median error (*MedErr*) and the accuracy at θ (Acc_θ), as proposed by Tulsiani and Malik [84], to measure the performance of the model. We do not use the *mAVP* metric since this experiment performs a pure viewpoint classification (or regression) task without any prior object detection.

Median error

The median error is the median of the vector of distances between the targets and the predicted viewpoints. The distance between two viewpoints θ and $\hat{\theta}$ is defined by the geodesic distance between the rotation matrices R_1 and R_2 associated with θ and $\hat{\theta}$ respectively [84]:

$$\Delta(R_1, R_2) = \frac{\left\|\log(R_1^T R_2)\right\|_F}{\sqrt{2}}. \tag{4.6}$$

Accuracy at θ

The accuracy at θ is the frequency at which the predicted viewpoint falls within a threshold of θ from the target viewpoint (i.e. $\Delta(R_1, R_2) < \theta$). In line with what is proposed by Tulsiani and Malik [84], we use $\theta = \frac{\pi}{6}$.

4.2.4 Results

The results for our regression method for viewpoint estimation ($Acc_{\frac{\pi}{6}}$, and *MedErr* in degrees) are as follows: $Acc_{\frac{\pi}{6}} = 30.83\%$ and $MedErr = 45.38$. It turns out that the regression method does not work very well for viewpoint estimation despite the fact that it seems to be a natural formulation to solve the problem. We will investigate classification methods, where each viewpoint is treated as a separate class.

4.3 Nearest neighbor classifier

In contrast with our expectations, the regression method for viewpoint estimation was not very effective. We therefore resort to classification methods to solve this problem. To start our investigation of classification models for viewpoint estimation we first use a non-parametric model, namely the k-nearest neighbor (kNN) classifier, as a baseline to compare the performance of more elaborate classification models. The performance of the nearest neighbor model is also compared to the regression method. In this section we present the nearest neighbor classification method and its results on the viewpoint estimation task.

The nearest neighbor classifier (or more generally the k-nearest neighbor (kNN) classifier) is a non-parametric machine learning algorithm that predicts the label of an unseen data point by making comparisons with data in a training set. A kNN classifier outputs the k closest data points to the test data point. The closeness of the data points is measured using a suitable similarity metric. In the current case of image classification, our data points are images. As these images are of high dimensional (400×400), a direct comparison of the test image to every image in the training is not feasible. Instead, the images are reduced to a lower-dimensional feature space where the comparison can be done more efficiently. Thus the first step of our kNN-based viewpoint estimation is to extract some useful features from the training images.

4.3.1 Feature extraction

Image features are attributes, often of lower dimension than the original image, that characterizes an image. Here we use a CNN-based method to extract features from the images. More precisely, we use a convolutional autoencoder (CAE) that converts the raw 400×400 images to lower-dimensional features.

Convolutional autoencoder

The convolutional autoencoder (CAE) is composed of two mirrored fully convolutional network (FCN) [51] blocks such that the output of the first FCN model corresponds to the input of the second FCN model, as illustrated in Figure 4.2. The task of the first FCN model (encoder) is to reduce the image to a lower-dimensional feature map of size 25×25 (CAE codes). Thus the input image of size 400×400 is fed to the first FCN model, and passed through various convolutional layers which finally produce the CAE codes. The second FCN (decoder) attempts to reproduce the original input image from the CAE codes. Therefore the CAE model learns to reproduce its input image at the output, without explicitly learning an identity map. This is achieved by minimizing the MSE between the input image and the output image.

Feature maps

After the convergence of the convolutional autoencoder model, the decoder part of the model is discarded. The encoder is then used to generate the 25×25 feature maps representing the image features. Each image in the dataset is fed to the encoder, and the output of the encoder is stored as the feature of that image along with the corresponding viewpoint label. Figure 4.3 illustrates the result of the encoding process.

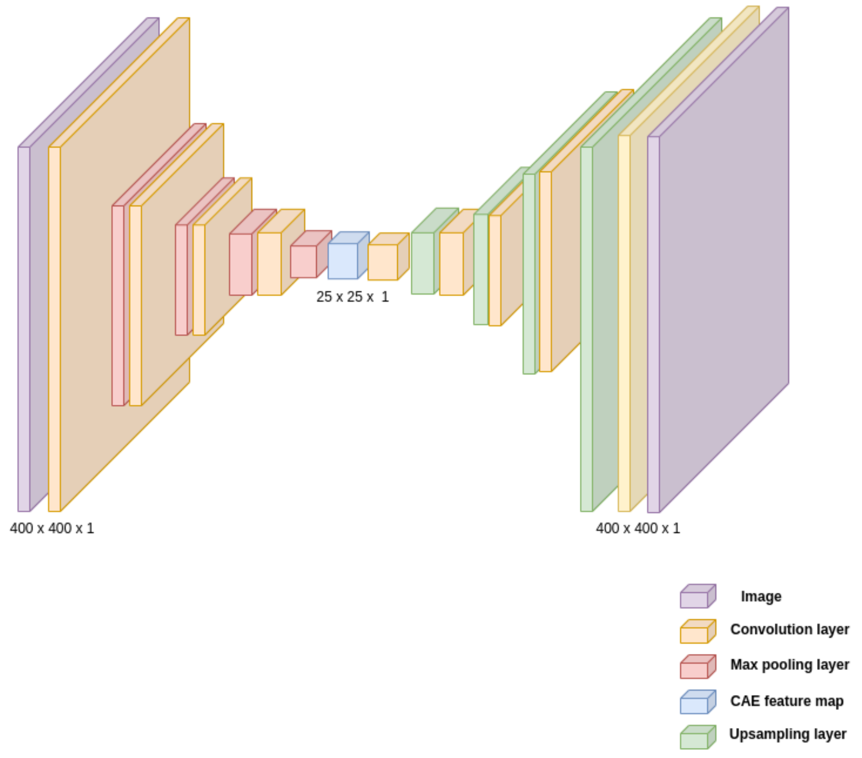


Figure 4.2: Architecture of the convolutional autoencoder that reduces the input images into lower dimensional feature maps.

4.3.2 Viewpoint prediction

The extracted feature maps of size 25×25 are flattened into 1D feature vectors. For each image in the training set, a feature vector is computed and stored along with the corresponding viewpoint label. Given a test image, we extract a feature vector, which is compared to the feature vectors from the images in the training set. We use the Euclidean norm to evaluate the distance between feature vectors.

Generally, a kNN classifier outputs the k most similar classes from the training set. Here we choose $k = 1$. For each test image, the kNN classifier outputs one viewpoint class corresponding to the closest image in the training set.

4.3.3 Results

The performance of the kNN model is evaluated using the accuracy at θ and the median error as described in Section 4.2.3. The results of the evaluation are as follows: $Acc_{\frac{\pi}{6}} = 57.78\%$ and $MedErr = 23.50$.

Although there is an improvement over the regression method, the performance of this classification method remains low and it is certainly not good enough for practical uses. We therefore need to develop more advanced classification methods.

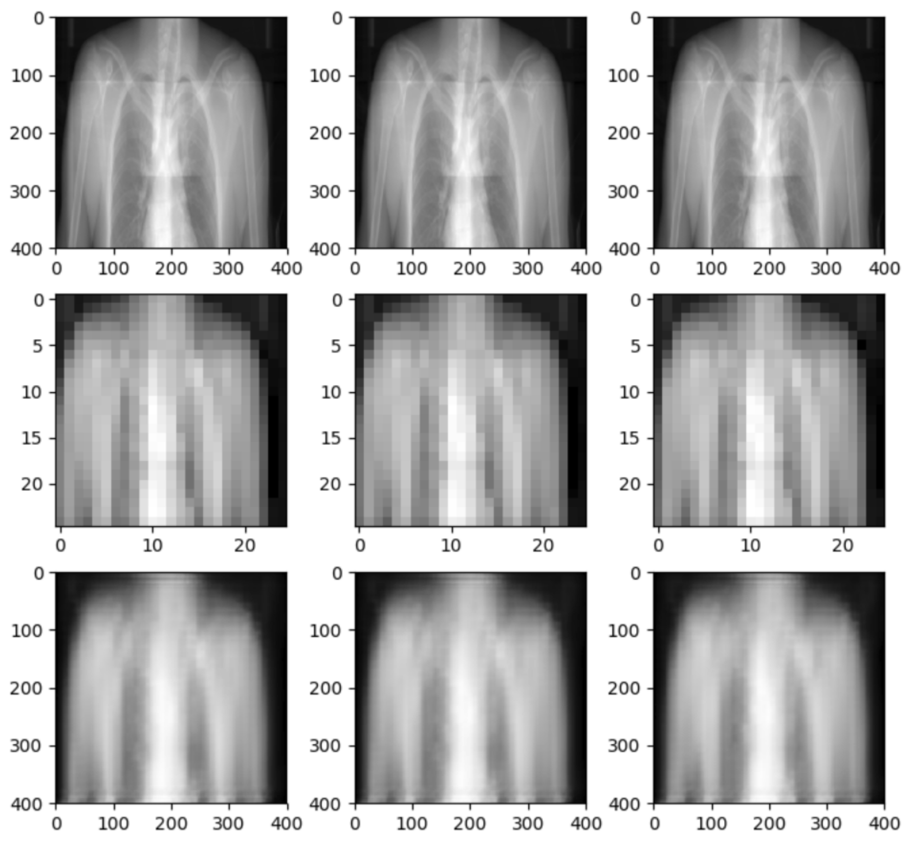


Figure 4.3: Autoencoder feature extraction. The first row represents three input images. Their encoded representation (feature map) is depicted in the second row. The last row is the reconstruction of the input image at the output by the autoencoder.

4.4 CNN-based classification

Deep learning models, particularly CNN, have achieved tremendous success and produced state-of-the-art results on many image classification problems [31, 42, 81]. Here we attempt to solve the viewpoint estimation problem using a classification model based on CNNs. In this section we present the classification model, the different experiments performed, and the results obtained.

4.4.1 CNN model

In order to solve the viewpoint estimation problem using a CNN, we need to discretize the viewpoint space as discussed in Section 4.1.1. The CNN model needs to solve a 360-class classification problem. Our CNN model is based on the Inception-V3 model [81] which was pre-trained on ImageNet for natural image classification. Since this model was initially designed to solve a 1000-class classification problem, its architecture must be modified for 360-class classification. The top fully-connected layer made of 1000 neurons is replaced by a pooling layer that summarizes the salient features extracted by the base CNN model. The pooling layer is then followed by a classification block (made of two fully-connected layers) which classifies the features extracted from the base model. The last fully-connected layer of the classification block is made of 360 units that represent each of the viewpoint classes. The diagram of the CNN-based classification model is depicted in Figure 4.4.

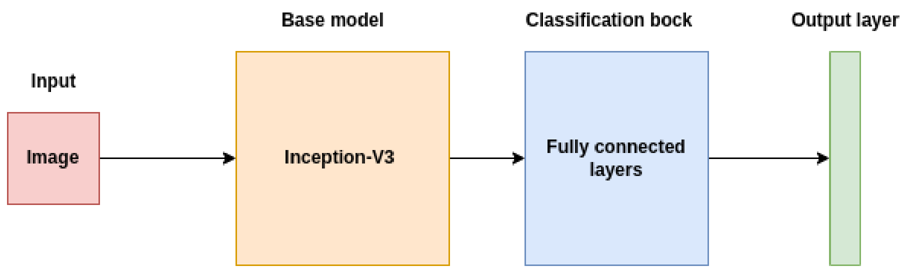


Figure 4.4: Block diagram of the CNN-based classification model.

4.4.2 Viewpoint label encoding

The class labels are one-hot encoded, i.e. sparse vectors containing 1 at the position corresponding to the ground truth class and 0 everywhere else (e.g. $y = [0, 0, 1, 0, \ldots, 0]$). This type of encoding allows efficient computation of the cost function using vectorized operations which are particularly optimized in machine learning libraries.

4.4.3 Cost function

For this classification task, we use a standard cost function, namely the cross-entropy loss defined by

$$L_{CE} = -\sum_{k=1}^{360} y_k \log(p_k), \tag{4.7}$$

where y_k and p_k are the k^{th} element of the target vector and the predicted probability vectors respectively.

This cost function measures the distance between two probability distributions. In this case, the input vector $y = (y_k)_{1 \leq k \leq 360}$ and the output vector $p = (p_k)_{1 \leq k \leq 360}$ are interpreted as probability distributions. The goal of the optimization is to minimize the cross-entropy loss by making these probability distributions as close as possible.

4.4.4 Model training

The CNN-based classification model is trained and evaluated on the same dataset as the regression model in order to compare the performance of the models. The SGD optimizer is used to minimize the cross-entropy cost function. After setting the hyperparameters, we train the model for a few epochs. We use an initial learning rate $\lambda = 10^{-3}$. The initial learning rate is decayed exponentially at every 1000 iterations with a decay rate of 0.96. The performance of the model is evaluated on the validation set after each epoch, and the training is stopped when the performance on the validation set stops improving.

4.4.5 Model evaluation

After convergence of the optimization, we stop the training and the best model is evaluated on the test set. We use the accuracy at θ and the median error as performance metrics. The proposed CNN-based classification model has an accuracy at $Acc_{\frac{\pi}{6}} = 56.11\%$ of and a median error $MedErr = 19$.

4.5 Summary

In this chapter we presented different methods to solve the one-dimensional viewpoint estimation problem. We first used a regression model based on CNNs to estimate the rotation angle around a 3D volume. The accuracy and the median error on the test dataset show that the regression method is not effective for solving the viewpoint estimation problem. We then investigated classification models, namely k-nearest neighbor, and CNN-based classifiers. The classification models, especially the CNN-based model, perform better than the regression method, although their performance is still not satisfactory.

In the next chapter we investigate more elaborate classification methods for one-dimensional viewpoint estimation.

Chapter 5

Geometry-aware classification

The CNN-based classification model developed to solve viewpoint estimation in Chapter 4 requires the discretization of the viewpoint space into a finite number of viewpoint classes. Also, the labels are represented using one-hot encoding where each label is a sparse vector containing zeros everywhere except at the location corresponding to the ground truth class. Since this sort of encoding assigns the highest label score to the ground truth class and the lowest score to all the other classes, we term the CNN model using this type of label representation *hard classification*.

The hard classification does not take into account the geometric nature of the viewpoint estimation problem since all classes which do not correspond to the target class are zero-weighted in the cost function. Thus it does not make a difference if the model makes a wrong prediction that is close to the target or far away. This is not ideal since we would rather have a wrong prediction that is close to the target than a wrong prediction that is far from the target. To address this problem, we use a "soft" labeling technique whereby the classes close to the target class are assigned non-zero weights.

In this chapter we explore methods that make a CNN-based classification model more adapted to the viewpoint estimation problem. In effect, the problem of viewpoint estimation has specific requirements due to the geometry of the viewpoint space. In Section 5.1 we present a soft label encoding method as an alternative to one-hot encoding. Section 5.2 presents a geometry-aware classification model using a label smoothing regularization technique, where the label softening discussed in Section 5.1 is integrated into the cost function. In Section 5.3 we study the effect of the input image size on the CNN model performance. In Section 5.4 we present the results of viewpoint estimation with offline data generation. The experimental results using data generated with DeepDRR are presented in Section 5.5. Although the offline data generation method is effective to solve the problem, it does not fully leverage 3D information in the CT volume. Using online data generation, we are able to augment the training data with small random rotations at each training iteration. The viewpoint estimation method with online data generation is presented in Section 5.6. The augmentation of the training data with random out-of-plane rotations improves the result of the viewpoint estimation model. To develop models that generalize well, in Section 5.7 we extend the viewpoint estimation models to learn from multiple CT scans. Section 5.8 presents a summary of the experiments and results.

5.1 Classification with soft labels

The one-dimensional viewpoint space around a 3D object is circular and continuous. Thus close viewpoints yield similar projections from the 3D object. This leads to a similarity in the viewpoint classes after the discretization of the viewpoint space. In hard classification, where the labels are one-hot-encoded, the CNN model aims at maximizing the logit score of the correct class while the scores of all the other classes are minimized. Hard classification assumes that the different classes are completely independent and that there is no similarity between them. This is obviously not the case for the viewpoint estimation problem, where some classes are more similar than others. In this section we propose a soft label representation of the viewpoint labels as an alternative to the one-hot encoding labels. The soft labels are then used to train a CNN model with a cross-entropy cost function.

5.1.1 Soft label encoding

For soft viewpoint labels, a non-zero weight is assigned to the ground truth class. In our experiments, the weight of the ground truth class is set to 0.2. Nearby classes are then assigned non-zero weights smaller than the weight of the ground truth class. We set the weights of the eight nearest classes to the ground truth to 0.1. For example, the soft label corresponding to the viewpoint class $\theta = 5°$ is defined by $y = [0, 0.1, 0.1, 0.1, 0.1, 0.2, 0.1, 0.1, 0.1, 0.1, 0, \dots, 0]$. This type of label encoding guides the CNN model to make predictions that are either equal to the ground truth or equal to another viewpoint close to the ground truth.

5.1.2 Cost function

For an effective classification, it is essential to train the model using a cost function that is able to guide the model to separate the different classes. Building on the previous experiments (of hard classification), we use a modified cross-entropy cost function as defined in Equation (5.1):

$$L_{MCE} = -\sum_{k=1}^{360} y_k \log(p_k), \tag{5.1}$$

where y_k and p_k are the k^{th} element of the target vector and the predicted probability vectors respectively. The vector $y = (y_k)_{1 \leq k \leq 360}$ is the soft label vector instead of the one-hot encoded vector in a classical cross-entropy loss. The vector $p = (p_k)_{1 \leq k \leq 360}$ is the output probability distribution.

The optimization of the modified cross-entropy loss implicitly optimizes the cost function of the usual cross-entropy loss.

5.1.3 Model training

Similar to the hard classification model, the soft classification model is based on the modified Inception-V3 CNN. The model is trained by minimizing the modified cross-entropy loss function using the stochastic gradient descent (SGD) optimizer. In order to compare the results fairly, we trained the model using the same training and validation data splits as in the hard classification case. The model is trained for a few epochs until convergence. Between each epoch, the performance of the model on the validation set is evaluated and the corresponding state of the model parameters is saved. The training is stopped when the performance of the model on the validation set does not improve in two successive epochs.

5.1.4 Evaluation and results

After training, the best-performing model on the validation set is restored and evaluated on the test set. The soft classification method gives better results than the hard classification method, as summarized in Table 5.1. The label softening technique improves the accuracy from 56.11% to 76.39% and lowers the median error from 19.0 to 9.0 degrees. The predictions from the soft classification model are therefore more precise than those of the hard classification model.

Table 5.1: Viewpoint estimation results for soft classification.

Methods	$Acc_{\frac{\pi}{6}}$ (%)	$MedErr$ (degrees)
Hard classification	56.11	19.00
Soft classification	76.39	9.00

5.2 Geometry-aware classification

One shortcoming of the soft label approach discussed in the previous section is that the dataset has to be relabeled with the new label weights. Also, the choice of label weights is specific to the application and they are unlikely to generalize well to other applications. We, therefore, investigate methods to improve the label-softening strategy proposed in the previous section.

One alternative to "softening" the labels in the training set is to use a geometric structure-aware loss function as proposed by Su et al. [79].

5.2.1 Geometry-aware cost function

The new cost function is a modified version of the normal cross-entropy loss. The adapted cross-entropy loss is defined by

$$L_{GCE} = -\sum_{k=1}^{360} \exp\left(-\frac{\Delta(k_{gt}, k)}{\sigma}\right) \log(p_k), \tag{5.2}$$

where k_{gt} is the ground truth viewpoint, σ is a hyper-parameter, and p_k is the probability of viewpoint k.

In this cost function, each viewpoint is given a weight that decays exponentially as the viewpoint moves away from the target viewpoint. This implicitly forces the model to predict viewpoints close to the target.

5.2.2 Optimal label softening

As can be seen in Equation (5.2), the label softening depends on a hyper-parameter σ which controls the decay rate of the viewpoint weights. In other words, the hyper-parameter σ determines how broad the exponential decay label softening should be for an optimal viewpoint estimation result. Figure 5.1 shows the distribution of the viewpoint labels for different values of σ. Larger values of σ make the label weight distribution very wide. This implies that more viewpoint classes centered around the ground truth viewpoint class will contribute to the loss for each prediction. On the other hand, smaller values of σ lead to a smaller distribution extent. Thus fewer viewpoint classes contribute to the cost function. When the value of σ is sufficiently small, the modified cross-entropy tends to the classical cross-entropy loss. This means that the classical cross-entropy loss is a special case of the modified cross-entropy loss. Therefore, in theory, a model trained using the modified cross-entropy loss would generalize better than one trained using the classical cross-entropy loss for the problem of viewpoint estimation.

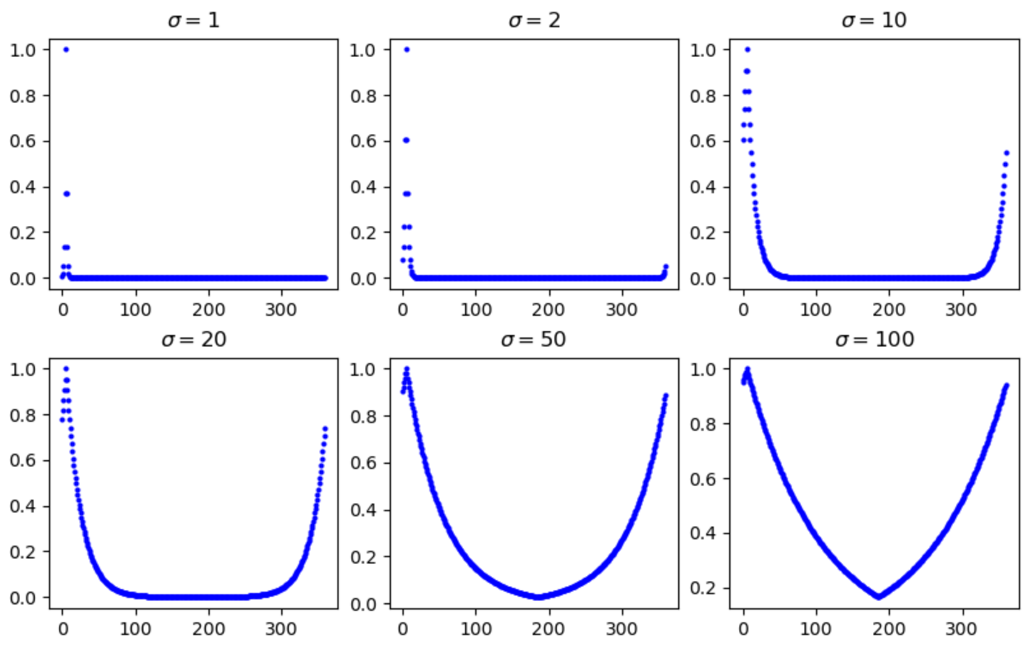


Figure 5.1: Class weights distribution for different values of σ, when the ground truth label is $\theta = 5°$.

We need to find the value of σ that gives the best result for our application. Figure 5.2 shows the result of viewpoint estimation for different values of the hyperparameter σ. For each value of the σ, a model is trained using the modified cross-entropy loss. The performance of the trained models is then evaluated on the test set. The value of σ that gives the best results is used in subsequent experiments.

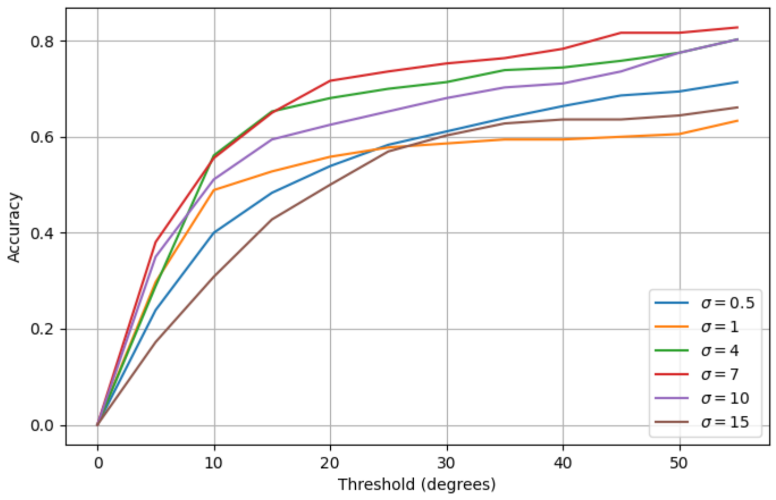


Figure 5.2: Variation of accuracy with σ.

5.2.3 Model training and evaluation

The model is trained using the SGD optimizer until convergence. The trained model is evaluated on the test data. The results of the geometric structure-aware classification method ($Acc_{\frac{\pi}{6}}$ and $MedErr$ in degrees) are compared to the results of other classification methods in Table 5.2.

Table 5.2: Viewpoint estimation results for soft classification with a geometry-aware cost function.

Methods	$Acc_{\frac{\pi}{6}}$ (%)	$MedErr$ (degrees)
Hard classification	56.11	19.00
Soft classification	76.39	9.00
Geometry-aware cost function	75.28	8.00

Thus the geometry-aware cost function method gives comparable results to the soft label classification model. However, the geometry-aware cost function has the advantage that the softening of the labels is not fixed. The label softening is therefore not application-specific as the labels can be adapted according to the degree of similarity between classes.

5.3 Effect of the size of the training images

In order to determine the optimal input size for the soft classification, we ran the experiment for different input sizes. The results for accuracy and median error are reported in Figure 5.3.

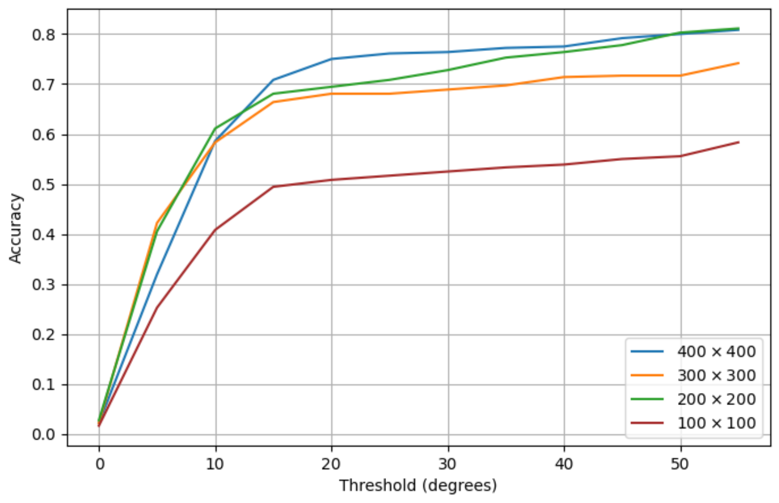


Figure 5.3: Effect of training image input size on accuracy.

We note a significant drop in performance when an input size of 100×100 is used, suggesting a loss of information due to the downsampling operation on the original image. The same drop in performance, although less pronounced than the one observed for the 100×100 case, is observed for the input size of 300×300. However, the input size of 200×200 gives similar results to the 400×400 input size case. This suggests that the amount of information lost when downsampling the original image from 400×400 is negligible. For computational efficiency, we will therefore be using an input size of 200×200 in all subsequent experiments.

5.4 Viewpoint estimation results

Here we summarize the results for viewpoint estimation for all the methods (regression, nearest neighbor, soft classification, hard classification, and geometry-aware classification). Figure 5.4 shows the variation of the accuracy as the threshold error increases, whereas Figure 5.5 shows the values of the median errors. We can see that the soft label and the geometry-aware classification methods significantly outperform the other methods.

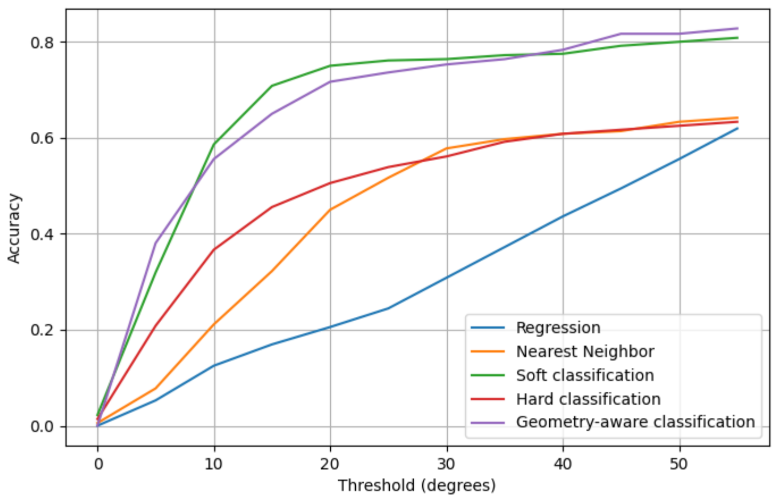


Figure 5.4: Accuracy of classification-based viewpoint estimation methods.

5.5 Experimental results using DeepDRR

To compare DeepDRR with FVR, we trained the viewpoint estimation models using data generated with DeepDRR. We report the results of the CNN-based regression and classification models in Table 5.3.

The results show that the models trained using data generated with DeepDRR perform better than those trained using FVR. This improved performance is probably

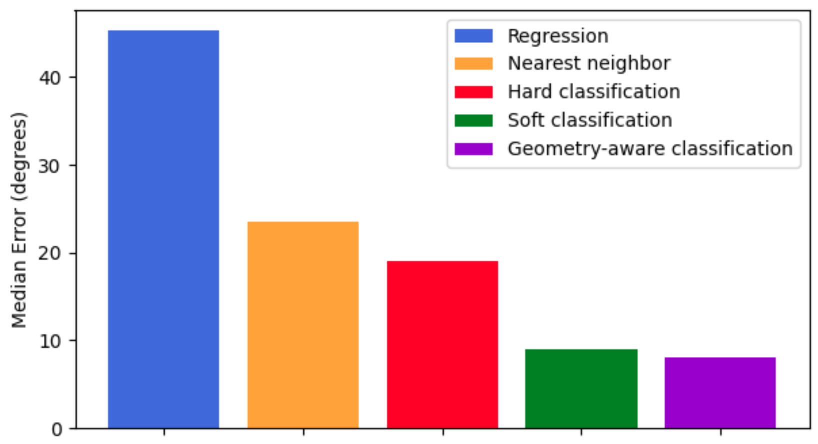


Figure 5.5: Median error of classification-based viewpoint estimation methods.

Table 5.3: Viewpoint estimation accuracy of viewpoint estimation models when trained using data generated with DeepDRR and FVR.

Methods	FVR	DeepDRR
Regression	30.83	37.22
Hard classification	56.11	61.67
Geometry-aware classification	75.28	87.22

due to the fact that the projection done in DeepDRR yields DRRs with more detailed hard tissue structures compared to the parallel projection done by FVR.

However, the experimental investigation done in this thesis would not be possible using DeepDRR—the computational and memory requirements would be too high for many of the methods explored. In the next experiments, we use the FVR method due to its computational efficiency, which allows us to generate data online at training time.

5.6 Viewpoint estimation using online data generation

In this section, we present the viewpoint classification experiment using an online data generation strategy. In Section 5.6.1 the data generation method is presented, followed by a description of the different data augmentation methods used to train the classification model in Section 5.6.2. The training process is presented in Section 5.6.3, followed by the evaluation and results in Section 5.6.4.

5.6.1 Online data generation

We use an online data generation strategy where the data required at each training iteration is generated on the fly. This allows the incorporation of 3D information in order to build a model that is more robust to variation between training and test images. However, online data generation has a drawback in that it adds a data generation time component to the overall training time. This could be limiting especially when the data generation method is not very fast. We, therefore, use the Fourier volume rendering (FVR) method for DRR generation, which is more efficient than the additive projection method. With the online data generation method, we are able to augment the training data with random translation, in-plane rotation, and out-of-plane rotation. The CT volume used for data generation has dimension $512 \times 512 \times 512$. After projection, each DRR has a size of 512×512.

5.6.2 Online data augmentation

As deep learning model training requires large amounts of data, various data augmentation techniques are usually used to effectively get more training data. We augment our training data using small translations and rotations of the input image.

Translation

The training images are made of 400×400 patches taken around the center of the 512×512 projected images from the CT volumes. Figure 5.6 shows the central patch taken from a projected image. For data augmentation with translations, a window defining the central image is translated horizontally and vertically by a random amount. The patch corresponding to the new position of the sliding window is extracted from the 512×512 projection and used as a new training image. We limit the translation to the range -20 to 20 pixels (up, down, right, and left) in the horizontal and vertical directions. The translation range is kept small to avoid getting training images that do not contain the body region of interest.

Random in-plane rotation

The training data is also augmented with small in-plane rotations in the interval -10 to 10 degrees. This augmentation allows us to get more training images and also to make the model robust to the possible difference between test data and training data due to in-plane rotation.

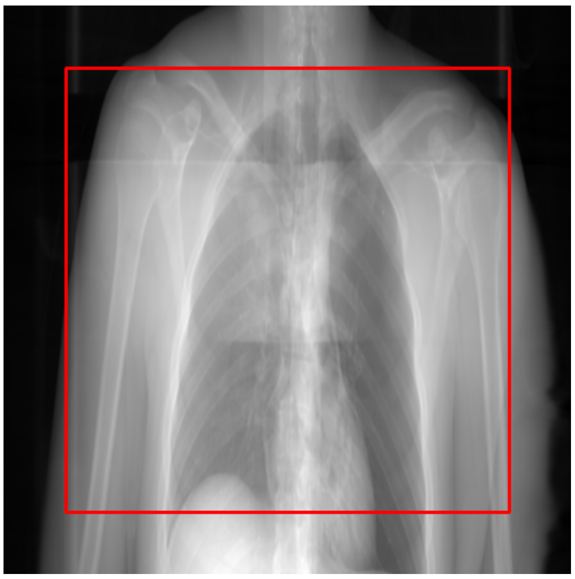


Figure 5.6: Central patch of size 400 × 400 extracted from an original image of size 512 × 512 using a bounding box.

Random out-of-plane rotation

In one-dimensional viewpoint estimation, we are interested in predicting the rotation angle around one of the reference frame axes, which we have chosen to be the longitudinal axis in this work. Rotations around the other axes (in-plane and out-of-plane rotations) can be used to augment the training data. Thus we augmented the training data with small out-of-plane rotations in the interval -5 to 5 degrees.

5.6.3 Model training

We use the same CNN model as in the previous experiment. For this experiment the data required at each training iteration is generated just before it is fed to the CNN model. Also, different data augmentation techniques are applied to the images to get sufficient training data. The model is trained by minimizing the geometric-aware cost function using the stochastic gradient descent algorithm. During the training process, the performance of the model on the validation data is evaluated every 2,000 iterations. The model is trained until the performance on the validation set stops improving. The performance of the trained model is then evaluated on the test data.

5.6.4 Evaluation and results

We evaluate the accuracy of the trained CNN model using different data augmentation methods. We compare the effect of in-plane rotation and small out-of-plane rotation on the performance of the model. The results from the additive projection

(AP) data generation method and the Fourier volume rendering (FVR) data generation are also compared to check the consistency of results. Table 5.4 summarizes the results of viewpoint estimation using data augmentation.

It turns out that augmenting the dataset with random out-of-plane rotations (± 5 degrees) significantly improves viewpoint estimation performance.

Table 5.4: Accuracy of the viewpoint estimation model trained with data augmentation.

Input data	$Acc_{\frac{\pi}{6}}(\%)$
AP	75.28
FVR	75.0
FVR + in-plane rotation	75.83
FVR + out-of-plane rotation	81.94

A comparison of the results with previous experiments is shown in Table 5.4. We note that the performance of the previous experiment (without random rotation data augmentation) can be reproduced using the online data generation strategy. Also, this experiment shows that data augmentation with random out-of-plane rotation substantially improves the viewpoint estimation performance (by 6.66%).

The results obtained are consistent with those obtained using the offline data generation method, proving that online data generation is not only an efficient method but also an effective strategy to train a CNN model for the viewpoint estimation task. Also, the out-of-plane rotation gives better performance than the in-plane rotation and translation-only data augmentation approaches.

5.7 Viewpoint estimation from multiple CT scans

In the previous experiments we generated all the training images from a single CT scan and used a different CT scan of a different patient to generate test data. The results demonstrate that this approach works reasonably well. However, we notice that the performance tends to drop when we test the model on other CT scans in the dataset, which suggests that the generalization power of the model is limited. To improve the generalization capability of the model, we augment the training set with images from other CT scan volumes. Thus the training set used is generated from 21 different CT scans, and the test set comes from eight different CT scans. Since the CTs are taken from different persons, there might be a mismatch between corresponding views from different CTs. This would render the viewpoint labels noisy and lead to poor performance. We need a cost function that is able to accommodate these noisy labels. With training images being generated from 21 different CT scans, an online DRR generation approach would be time-consuming and limiting. We therefore resort to an offline data generation method. With this approach the type of data augmentation that can be done during training is restricted to small translations and random in-plane rotations on the projected training images.

In Section 5.7.1 we investigate the effect of the translation range on the performance of the viewpoint estimation model. The effect of the input image size is presented in Section 5.7.2. In Section 5.7.3 we present the effect of test-time data augmentation.

5.7.1 Effect of translation range on viewpoint estimation performance

In this section we aim to determine whether there is an optimal range of the translation window in the training image that accounts for objects shifting between the training image and the test image. We can try to make the viewpoint estimation model translation invariant by augmenting the training data with appropriate random translates of the training images

We ran a series of experiments with different translation ranges and evaluated the effect of translation on viewpoint estimation performance. We started with a translation range of 1×1, corresponding to the central sub-image of 400×400 taken from the original DRR of size 512×512. Thus the translation range of 1×1 corresponds to an absence of translation augmentation in the training image. This is used to establish a baseline for all the experiments with various translation ranges. The translation window sizes experimented with are 1×1, 10×10, 20×20, 40×40, and 50×50.

For each translation range experimented, the model is evaluated on a test image with a translation range 1×1 (i.e. on the central sub-image without translation). As the translation range increases, the model is increasingly able to capture the shift between training and test images. However, the performance does not improve indefinitely, since the performance drops above the 40×40 translation range. Figure 5.7 summarizes the results for the different translation ranges experimented with.

5.7.2 Effect of the size of the sub-window used for training

In this section we investigate the sensitivity of the viewpoint classification model to the size of the input image. We experiment with two different sub-windows of the original image. The first experiment was done using an image of size 400×400, and the second with an image of size 200×200.

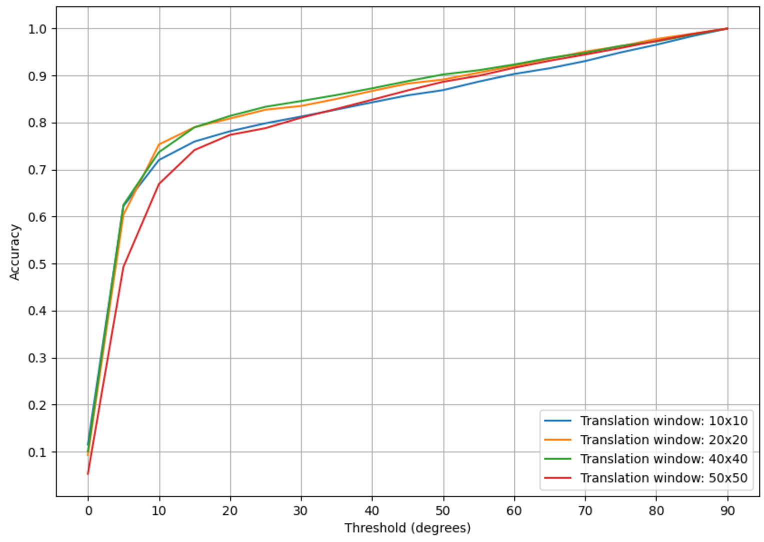


Figure 5.7: Effect of translation range on viewpoint estimation performance.

The results of the two experiments are shown in Figure 5.8. It can be seen that the results obtained are quite similar. The size of the sub-image of the DRR used for training does not have a significant impact on the model's performance. This confirms that the model is able to extract texture information from the image, and is not just classifying the image based on the silhouette of the object present in the image.

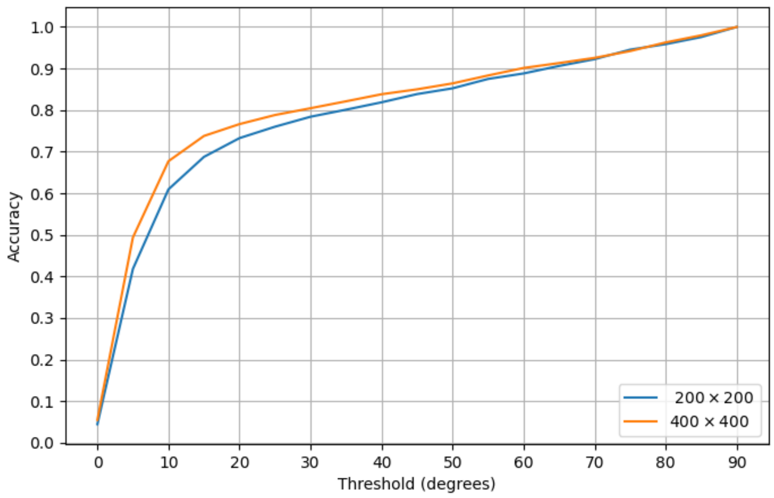


Figure 5.8: Effect of the size of the sub-window used for training.

5.7.3 Test-time data augmentation

Despite the augmentation done on the training data to make the model translation invariant, there is still a drop in performance for certain test CT scans due to the shift between training and test images. This suggests that the translation on the training images is not sufficient to capture the misalignment between the object in a training image and a test image. One solution to this problem is test-time data augmentation [42] whereby predictions are made not only for a test image but also for augmented versions of the test image. The projected DRR from the CT volumes are images of size 512×512. We extract a test image from this DRR by taking a crop of size 400×400 from the center of the original DRR, in conformity with the method used to generate the training images. We then generate augmented versions of this test image by translating a sliding window horizontally and vertically around the center of the original DRR. At each location of the sliding window, a patch is extracted from the DRR to form a new test image. The main test image and all its augmented versions are passed through the classification model, which predicts a candidate viewpoint. Since the predicted viewpoint must be unique for each test image, we need some way to determine the actual prediction from all the candidate viewpoint predictions. One way to derive the viewpoint prediction is by taking the average value of all the candidate predictions, as proposed by Krizhevsky et al. [42]. Alternatively, we can associate a confidence score to each candidate prediction and use the candidate prediction with the highest confidence score as the predicted viewpoint. In this section we first investigate the effectiveness of the average aggregation method. We then explore a few confidence scoring methods, namely an entropy measure, and an autoencoder reconstruction error. The performance of the test-time augmentation methods is compared to the result of the single test image without data augmentation in Table 5.5.

Prediction average

Given a DRR of size 512×512 generated from a test CT scan, we crop 81 patches of size 400×400 around the center of the DRR by moving a sliding window across the image in a range of -20 to 20 pixels horizontally and vertically. The viewpoint estimation model makes a prediction for each test image, and the average of all 81 predictions is used as the predicted viewpoint for the original DRR. The results summarized in Table 5.5 show that the averaged prediction is worse than the prediction obtained using only the central patch of the original test image.

Averaging the prediction from all possible object locations is not an effective strategy. This can be explained by the fact that the averaging combines good predictions and bad predictions together without any scoring mechanism to evaluate the accuracy of each prediction. We therefore require a more elaborate technique that is able to quantify the accuracy of each prediction from a candidate image. The next experiments explore a few confidence scoring methods.

Entropy

In information theory, the entropy measures the uncertainty in a probability distribution. The entropy is maximal for a uniform distribution (total uncertainty since no event has priority over the other) and minimal for a certain outcome. Since the output of our neural network is a softmax probability distribution, we can use the entropy to quantify how confident the model is about the predicted viewpoint. A low entropy value would indicate high confidence in the model prediction whereas a high entropy would indicate a high uncertainty in the model's prediction.

We investigate whether the entropy of the softmax probability distribution of the viewpoint estimation model would be a good measure of the confidence of the model about the predicted viewpoint. Thus the prediction with the smallest entropy is compared to the prediction with only the central sub-image of the test DRR.

For each test DRR of size 512×512, we generated 81 augmented test images of size 400×400 around the center of the test DRR. The prediction of each test image of size 400×400 is stored on a heat map of confidence scores as shown in Figure 5.9. The prediction corresponding to the minimal value of the entropy is used as the predicted viewpoint of the original test image. This approach proves to give better performance than simply evaluating the performance using the central image of each DRR, as illustrated in Table 5.5.

Autoencoder

An autoencoder is primarily used for dimensionality reduction to compress high-dimensional data into a more compact representation. Technically the autoencoder is a self-supervised learning model that aims to reconstruct its input at the output without simply copying the input to the output. In the dimensionality reduction context, it is the inner representation of the autoencoder that is of interest, as the latent representation of the input is typically of lower dimension than the input. When an autoencoder is successfully trained to reconstruct images from the training set of a particular dataset, the quality of a reconstructed test image could be used to evaluate how similar the test image is to the images that the autoencoder was trained on. Thus an autoencoder can be used as an outlier detector. Here we aim to use the autoencoder to identify whether the test image is significantly different from the images used to train the viewpoint estimation model, and to use prediction

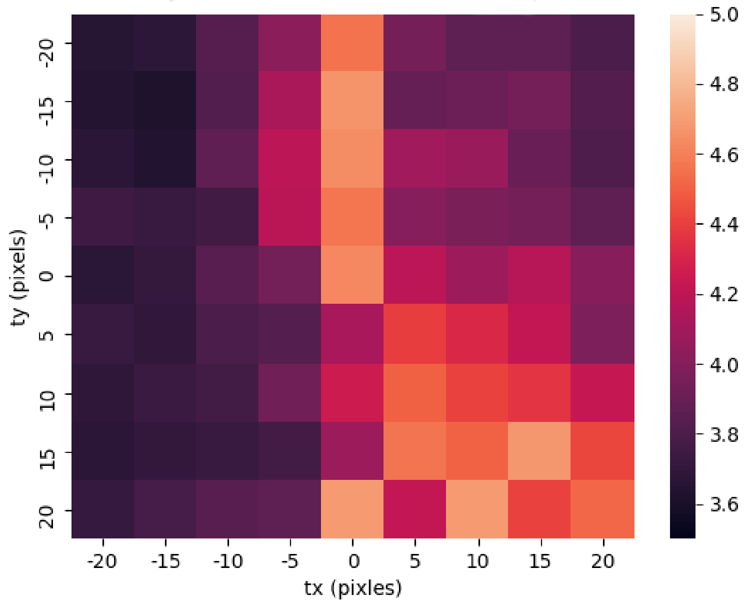


Figure 5.9: Entropy heat map.

from the most relevant candidate images only.

We use the reconstruction error of an autoencoder to detect outlier images in the test data. We then evaluate the viewpoint estimation model on those test images that the autoencoder can reconstruct accurately.

We implemented a convolutional autoencoder that reconstructs training images accurately. Each of the 81 test images cropped from the original DRR is passed to the autoencoder. The image with the lowest reconstruction error represents the image that most resembles the training data. We used the prediction of this test image as the actual predicted viewpoint of the model. It can be seen that although the predictions obtained with the autoencoder are better than the predictions of the baseline, the performance is not as good as the performance obtained with the entropy method. The results are summarized in Table 5.5.

Table 5.5: Test-time data augmentation results. The Reference method is obtained using a single test image without data augmentation.

Scoring Method	$Acc_{\frac{\pi}{6}}$ (%)
Reference	80.45
Entropy	83.68
Average	67.33
AE	77.64

5.8 Summary

In this chapter we presented several CNN-based classification methods for the viewpoint estimation problem. We first used soft labels as an alternative to one-hot-encoded labels. This makes the classification model more aware of the similarity of nearby viewpoint classes. We then integrated the label softening process into the cost function by using a label softening, which eliminates the need to re-label the dataset with soft labels. Moreover, we investigated methods to train the viewpoint estimation model using online data generation. Generating the data online allows us to augment the training data with out-of-plane rotation which improves the performance of the viewpoint classification model. We also investigated how test-time data augmentation influences viewpoint estimation performance.

Chapter 6

Three-dimensional viewpoint estimation

The viewpoint estimation task aims to identify the position of a camera with respect to an object in a 3D world using the projection of this object in a 2D image. Viewpoint estimation is a 3D problem that cannot be solved by estimating just one rotation angle. A full 3D rotation is therefore necessary to fully define an object's viewpoint. There are various parametrizations possible for a 3D rotation angle. The most common representation uses Euler angles, which are an extension of one-dimensional rotation representations, by considering rotations around each of the three axes of the Cartesian coordinates system. This representation becomes sometimes ambiguous due to the order of the combination of the three individual rotations. Another possible representation is the axis-angle representation, where a rotation in three-dimensional space is defined by a one-dimensional rotation around an arbitrary axis. A more effective method to represent rotations in 3D is the quaternion representation where any rotation in a 3D space is represented by a unit 4D vector. The quaternion representation is equivalent to the axis angle parametrization as each unit quaternion can be written in axis-angle notation. However, quaternion parametrization is often preferable as it represents rotations more concisely and makes rotation composition straightforward. In this chapter we present the rotation parametrization techniques and motivate the choice of quaternions for 3D viewpoint estimation.

In Section 6.1 we present the different parametrizations possible for 3D viewpoint estimation with their advantages and limitations. Section 6.2 presents a baseline approach for 3D viewpoint estimation using an Euler angle parametrization of rotations. In Section 6.3 we present a regression method for viewpoint estimation using a quaternion representation. In Section 6.4 we present a method for classifying quaternions for viewpoint estimation. Section 6.5 concludes the chapter with a summary of the results.

6.1 Viewpoint representation

In three-dimensional space the viewpoint of an object is fully defined by six parameters: $vp = [t_x, t_y, t_z, r_x, r_y, r_z]$, where the first three parameters define the position of the object and the last three define the orientation of the object. Here we restrict the viewpoint estimation problem to the determination of the orientation parameters (r_x, r_y, r_z). There are several parametrizations possible to define the 3D orientation of an object. In this section we present Euler angle, axis-angle, and quaternion representations of 3D rotations and expose their respective advantages and disadvantages.

6.1.1 Euler angles

In a three-dimensional Euclidean coordinate system, the Euler angles are defined by a rotation around each of the coordinate axes. Each of these rotations can be associated with a rotation matrix. The rotations of angle α, β, and γ around the x-axis, y-axis, and z-axis are defined by three rotation matrices as follows:

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}, \tag{6.1}$$

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}, \tag{6.2}$$

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{6.3}$$

$R_x(\alpha)$ represents a rotation of angle α about the x-axis, $R_y(\beta)$ represents a rotation of angle β about the y-axis, and $R_z(\gamma)$ is a rotation of angle γ about the z-axis. A 3D rotation using an Euler angle representation is defined as the composition of rotations around each of the three axes. Thus the resulting rotation can be expressed as $R = R_x(\alpha)R_y(\beta)R_z(\gamma)$, which represents rotations around $z - y - x$ axes successively. Since matrix multiplication is not commutative, changing the order of composition results in a different 3D rotation matrix. There are six possible combinations: $x - y - z$, $x - z - y$, $y - x - z$, $y - z - x$, $z - x - y$, and $z - y - x$.

Each of these sequences corresponds to a particular 3D rotation.

Thus the Euler angle rotation definition depends on the order of composition of the rotation around each of the three axes of the Euclidean coordinate system. Once the order of composition is chosen, any 3D rotation can be obtained by using the three individual 1D rotations. The main advantage of Euler angles for 3D rotation representation is that they are simple to understand since they are a mere extension of 1D rotation (three consecutive 1D rotations around different axes). This type of representation of 3D rotations has a major shortcoming, namely the gimbal lock problem where objects subject to 3D rotations lose one degree of freedom and become restricted to 2D rotations. This happens when the angle of the rotation around the second axis, the y-axis in this case, is equal to $\frac{\pi}{2}$.

Let $\beta = \frac{\pi}{2}$ in $R = R_x(\alpha)R_y(\beta)R_z(\gamma)$. Then the resulting rotation is given by

$$R = R_x(\alpha)R_y(\frac{\pi}{2})R_z(\gamma) \tag{6.4}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.5}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ \sin\gamma & \cos\gamma & 0 \\ -\cos\gamma & \sin\gamma & 0 \end{bmatrix} \tag{6.6}$$

$$= \begin{bmatrix} 0 & 0 & 1 \\ \cos\alpha\sin\gamma + \sin\alpha\cos\gamma & \cos\alpha\cos\gamma - \sin\alpha\sin\gamma & 0 \\ \sin\alpha\sin\gamma - \cos\alpha\cos\gamma & \sin\alpha\cos\gamma + \sin\alpha\sin\gamma & 0 \end{bmatrix} \tag{6.7}$$

$$= \begin{bmatrix} 0 & 0 & 1 \\ \sin(\alpha+\gamma) & \cos(\alpha+\gamma) & 0 \\ -\cos(\alpha+\gamma) & \sin(\alpha+\gamma) & 0 \end{bmatrix}. \tag{6.8}$$

The resulting rotation matrix depends on $\alpha + \gamma$, which can be replaced by a third rotation angle. Thus the rotations around the $z - x$ axes collapse into a rotation around a single axis. This leads to the loss of one degree of freedom known as gimbal lock.

Gimbal lock could be a serious problem in some applications when an object subject to 3D rotation using Euler angles representations is stuck in some orientation (e.g. an aircraft unable to pitch to change its altitude). An alternative to the Euler angle representation is the axis-angle representation where 3D rotations are parametrized as a rotation around an arbitrary axis in a three-dimensional space.

An illustration of the projection resulting from the three rotation axes is presented in Figures 6.1, 6.2, and 6.3.

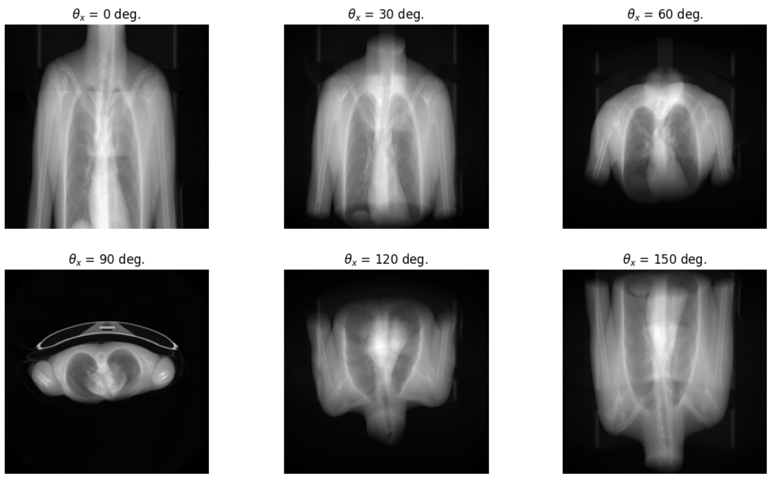


Figure 6.1: Rotation around the x-axis.

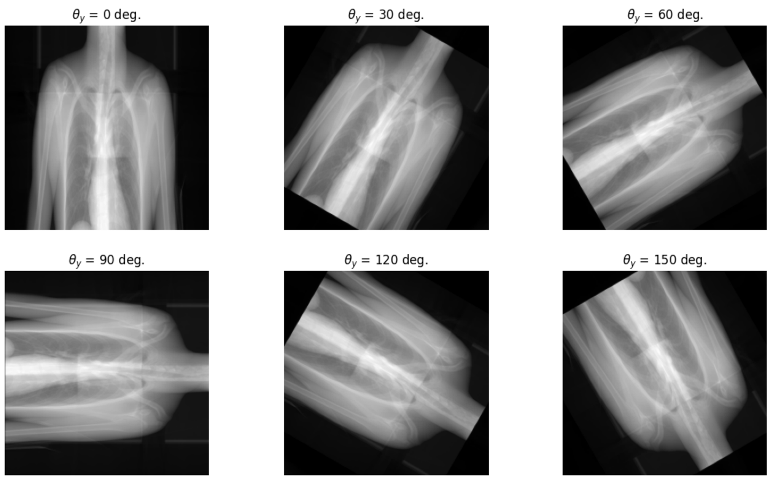


Figure 6.2: Rotation around the y-axis.

6.1.2 Axis-angle representation

The axis-angle representation is defined by a rotation around an arbitrary axis in 3D. This representation is parametrized by $(\theta, \mathbf{u})$, where $\mathbf{u}$ is an arbitrary unit vector in 3D and $\theta \in [0, \pi]$ is a rotation angle. In particular, rotations of angles α, β, and γ around the base vectors $\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$ of the Cartesian coordinates systems can be represented as $(\alpha, \mathbf{i})$, $(\beta, \mathbf{j})$, $(\alpha, \mathbf{k})$ respectively. Thus $(\frac{\pi}{4}, [1, 0, 0])$ represents a rotation of angle $\theta = \frac{\pi}{4}$ about the x-axis. As the rotation vector is not restricted

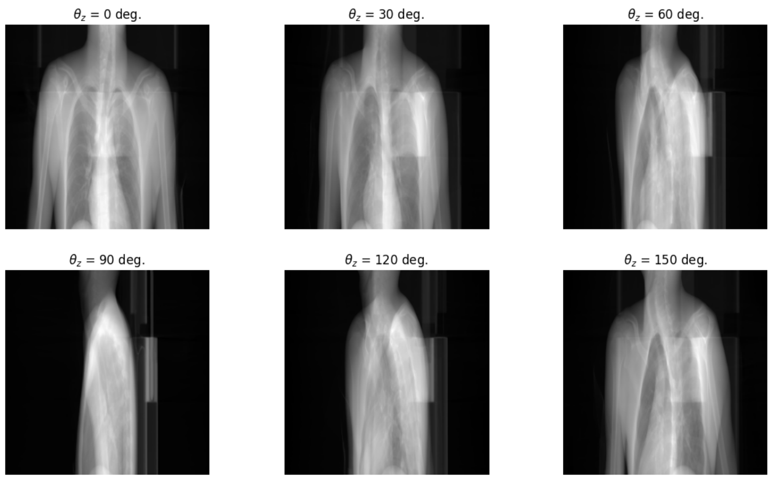


Figure 6.3: Rotation around the z-axis.

to the reference axes of the Cartesian coordinates system, any orientation can be achieved by choosing a suitable rotation axis and rotation by some angle about the chosen rotation axis.

Axis angle can be converted from Euler angle representation. Let

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{6.9}$$

be the rotation matrix obtained after composing the three Euler angle rotations. The equivalent axis-angle representation is defined by the angle $\theta = \arccos\left(\frac{\mathrm{trace}(R)-1}{2}\right)$ and the rotation axis $\mathbf{u} = \frac{1}{2\sin\theta}\begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix}$. Although the axis-angle representation avoids the gimbal lock problem encountered by the Euler angle representation, it comes with its own shortcomings. Like Euler angles, the axis-angle representation is not very practical for rotation interpolation. A representation of 3D rotation that works well for rotation interpolation is the quaternion representation.

6.1.3 Quaternions

For years mathematicians sought an extension of complex numbers to higher dimensional spaces. Just as a complex number $z = a + ib$ (where a and b are real numbers and i is an imaginary number) can be represented on a 2D plane as a point of coordinates (a, b), William Hamilton initially proposed an extension of complex numbers in a 3D space by adding a second imaginary number. Thus the equivalent of a complex number in 3D could be represented as $z = a + ib + jc$, where (a, b, and c are real numbers and i and j are imaginary numbers such that $i^2 = j^2 = -1$). But this representation turned out not to be useful as it is not defined for some elementary algebraic operations such as multiplication.

Definition

Hamilton extended the new complex number with another imaginary part. Thus the resulting "complex" number is written as $z = a+ib+jc+kd$, where a, b, c, d are real numbers and i, j, k are imaginary numbers such that $i^2 = j^2 = k^2 = ijk = -1$. This extension of complex numbers is called a quaternion [28]. We can easily check that addition and multiplication are well-defined with this new extension.

Addition and multiplication of quaternions

Let $z_1 = a_1 + ib_1 + jc_1 + kd_1$ and $z_2 = a_2 + ib_2 + jc_2 + kd_2$ be two quaternions. We define the quaternion addition as

$$\begin{aligned} z &= z_1 + z_2 \\ &= (a_1 + a_2) + i(b_1 + b_2) + j(c_1 + c_2) + k(d_1 + d_2). \end{aligned} \tag{6.10}$$

Thus the addition of two quaternions is again a quaternion. Moreover, we can define the multiplication of two quaternions as

$$\begin{aligned} z &= z_1 z_2 \\ &= (a_1 + ib_1 + jc_1 + kd_1)(a_2 + ib_2 + jc_2 + kd_2) \\ &= (a_1a_2 - b_1b_2 - c_1c_2 - d_1d_2) + i(a_1b_2 + b_1a_2 + c_1d_2 - d_1c_2) \\ &+ j(a_1c_2 - b_1d_2 + c_1a_2 + d_1b_2) + k(a_1d_2 + b_1c_2 - c_1b_2 + d_1a_2). \end{aligned} \tag{6.11}$$

Therefore the product of two quaternions is also a quaternion.

Unit quaternion representation

Similar to a complex number that can be represented as a couple of two real numbers, a quaternion $z = a+ib+jc+kd$ can be represented as 4-dimensional vector (a, b, c, d), with Euclidean norm $\|z\| = \sqrt{a^2 + b^2 + c^2 + d^2}$. When $\|z\| = 1$, z is called a unit quaternion. Just like a unit norm complex number is used to represent rotations on a plane, unit quaternions can be used to represent rotations in a 3D space. A 3D rotation corresponding to the axis-angle $(\theta, \mathbf{u})$ is equivalent to the unit quaternion $q = (\cos\frac{\theta}{2}, u_x \sin\frac{\theta}{2}, u_y \sin\frac{\theta}{2}, u_z \sin\frac{\theta}{2})$.

One of the main advantages of (unit) quaternion representation over other types of 3D rotation representations is that rotation transformations can be composed efficiently just by multiplying quaternions. Since the product of two unit quaternions is again a unit quaternion, multiplying unit quaternions is equivalent to composing rotations.

6.1.4 Distance between rotations

When predicting viewpoints in 3D, there is generally a difference between the predicted viewpoint and the ground truth viewpoint. One often needs to estimate the error made by predicting a viewpoint different from the ground truth viewpoint. There is a need to find an accurate metric to measure the distance between 3D rotations.

Consider two 3D rotations parametrized by their Euler angle representations $(\alpha_1, \beta_1, \gamma_1)$ and $(\alpha_2, \beta_2, \gamma_2)$. The angular distance between these two rotations could be evaluated using the Euclidean distance between the corresponding points in a 3D space as

$$d = \sqrt{(\alpha_1 - \alpha_2)^2 + (\beta_1 - \beta_2)^2 + (\gamma_1 - \gamma_2)^2}. \tag{6.12}$$

However, the expression in Equation (6.12) is not really an angular distance due to the 2π-periodicity of orientations in 3D (an orientation corresponding to a rotation angle θ is equivalent to an orientation corresponding to a rotation angle $\theta + 2\pi$). The distance function in Equation (6.12) could be large for different sets of angles corresponding to the same orientation. An alternative approach to measuring angular distance is to use a matrix representation.

Let R_{α_i}, R_{β_i} and R_{γ_i} be the rotation matrices corresponding to the rotation angles α_i, β_i and γ_i respectively. The Euler angle vector $(\alpha_1, \beta_1, \gamma_1)$ corresponds to the rotation $R_1 = R_{\alpha_1} R_{\beta_1} R_{\gamma_1}$, and $(\alpha_2, \beta_2, \gamma_2)$ corresponds to $R_2 = R_{\alpha_2} R_{\beta_2} R_{\gamma_2}$. The distance between R_1 and R_2 is defined by

$$d(R_1, R_2) = \frac{\|R_1 R_2^T\|_F}{\sqrt{2}}, \tag{6.13}$$

where $\|R\|_F$ is the Frobenius norm of the matrix R. Let $R = (a_{ij})_{1 \leq i,j \leq N}$. The Frobenius norm of R is given by

$$\|R\|_F = \sqrt{\sum_{i=1}^{N} \sum_{j=1}^{N} a_{ij}^2}. \tag{6.14}$$

Although the distance based on the matrix representation effectively captures the symmetry in 3D rotations, it is computationally more expensive than the distance metric using a quaternion representation.

Let q_1 and q_2 be the unit quaternions corresponding to rotation matrices R_1 and R_2 defined previously. The distance between the rotations corresponding to q_1 and q_2 is defined by

$$d(q_1, q_2) = 2 \arccos |q_1 \cdot q_2|. \tag{6.15}$$

Therefore, the quaternion representation offers a compact and efficient way of calculating the distance between rotations.

6.2 Euler angles estimation

The orientation of an object in a three-dimensional space using the Euler angle representation is fully defined by three distinct rotation angles: rotation about the x-axis, rotation about the y-axis, and rotation about the z-axis. In this section we extend the method developed for rotation angle estimation around the z-axis to the simultaneous prediction of all three Euler angles.

6.2.1 Model architecture

The model used for the estimation of the 3D Euler angles is an extension of the proposed model for 1D viewpoint classification. Here we use the Inception-V3 [81] model for feature extraction, followed by a three-branch classification block. The architecture of the proposed model is presented in Figure 6.4.

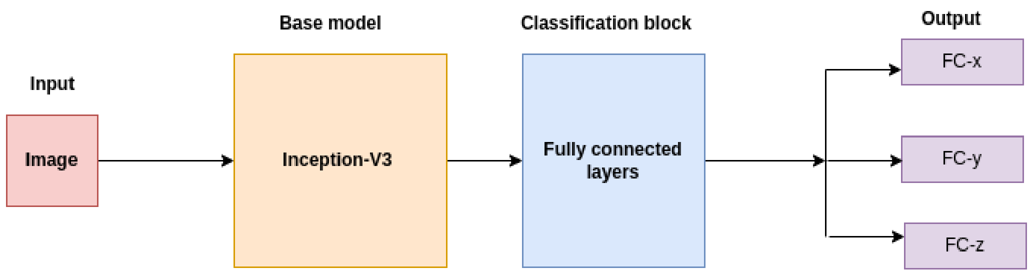


Figure 6.4: CNN-based classification model for 3D Euler angle estimation.

Each branch of the classification block predicts the angle around a particular rotation axis. For accurate viewpoint prediction, each rotation angle is partitioned into fine-grained bins of one degree each. Thus the rotation angle, θ_x, representing the elevation angle is partitioned into 180 bins (in the interval $[0, 180)$). On the other hand, the rotation angles θ_y and θ_z, representing the in-plane rotation and azimuth angles respectively, are partitioned into 360 bins ($\theta_y, \theta_z \in [0, 360)$). Therefore, the output of the classification branch of θ_x is an FC layer with 180 neurons whereas the outputs of the branches corresponding to θ_y and θ_z are made up of FC layers of 360 neurons each.

6.2.2 Model optimization

The classification model for 3D Euler angle estimating is trained by minimizing a composite cost function that optimizes the loss of each of the three classification branches. The resulting cost function is defined by

$$L = L_{\theta_x} + L_{\theta_y} + L_{\theta_z}, \tag{6.16}$$

where L_{θ_i} is the loss corresponding the prediction of rotation angle θ around the axis $i \in \{x, y, z\}$. Each loss is given by the geometric structure-aware cost function

$$L_{\theta_i} = -\sum_{k=1}^{N} \exp\left(-\frac{\Delta(k_{gt}, k)}{\sigma}\right) \log(p_k), \tag{6.17}$$

where N stands for the number of bins, k_{gt} is the target viewpoint, σ is a hyper-parameter, and p_k is the probability of viewpoint k.

This minimization of this composite cost function simultaneously minimizes the loss associated with each rotation angle. The stochastic gradient descent (SGD) optimizer with scheduled learning rate decay is used for minimization.

6.2.3 Evaluation

For each input image, the three Euler angles are predicted and the accuracy at $\theta = \frac{\pi}{6}$ for each angle is calculated using the same procedure as for the 1D viewpoint estimation case. Figure 6.5 shows the accuracy at $\theta = \frac{\pi}{6}$ for each rotation angle. The results indicate that the rotations around the x and y axes are easier to predict than the rotation around the z-axis. This is probably due to the fact that the rotation angle θ_z represents an out-of-plane rotation covering the whole contour of the body ($\theta_z \in [0, 360)$), whereas the rotation around the x-axis is an out-of-plane rotation covering half of the rotation range around the x-axis ($\theta_z \in [0, 360)$)— which makes the projection less ambiguous and easier to predict. On the other hand, the rotation around the y-axis represents an in-plane rotation of the rigid body, which is understandably easy to predict since all rotating points in the image constantly remain in the same plane.

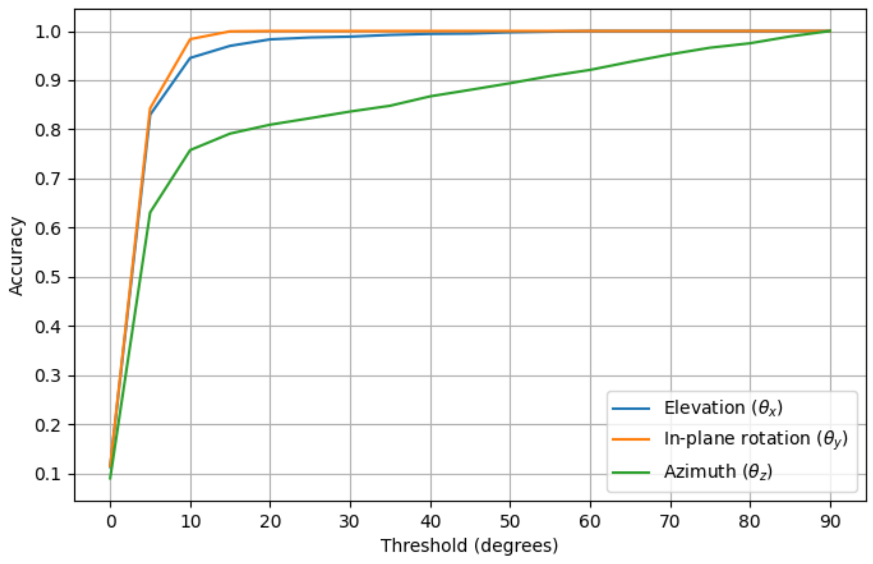


Figure 6.5: Accuracy of the joint estimation of all three Euler angles.

6.3 Regression of viewpoints using quaternions

The 3D viewpoint space is continuous by definition. It is therefore natural to solve the viewpoint estimation problem using a regression model. Since the results of regression models using an Euler angle representation did not perform well, in this section we investigate the use of a quaternion as the angle representation. Since quaternions are nicer to interpolate than Euler angles, we use a regression model for this type of angle representation. Given that the viewpoint space is continuous, we need to sample unit quaternions around the object before regression.

6.3.1 Representation of viewpoints with unit quaternions

In this section, we present the different methods we use to represent the 3D viewpoint with unit quaternions. Firstly, we present how the Euler angles can be converted to unit quaternions. We use a method proposed by Deserno [14] to generate the Euler angles such that they cover a unit sphere uniformly to avoid oversampling on the viewpoint space and the gimbal lock problem. The sampled Euler angles are then converted to unit quaternions. Lastly, we present how to generate unit quaternions directly from the viewpoint space using a sampling method proposed by Marsaglia [58].

Converting Euler angles to unit quaternions

Each triplet of Euler angles can be associated with a unit quaternion. We can therefore sample unit quaternions from the viewpoint space by converting a triplet of Euler angles to quaternions.

Let α, β and γ be the rotation angles around the x-axis, y-axis, and z-axis respectively. The rotations of angles α, β and γ can be represented with three unit quaternions: $q_\alpha = \cos\frac{\alpha}{2} + i\sin\frac{\alpha}{2}$, $q_\beta = \cos\frac{\beta}{2} + j\sin\frac{\beta}{2}$, and $q_\gamma = \cos\frac{\gamma}{2} + k\sin\frac{\gamma}{2}$. Arbitrary rotation in the 3D space can be obtained by multiplying the three elementary quaternions. Here we define 3D rotation by successive rotation about the z-axis, x-axis, and y-axis (representing roll, pitch, and yaw convention). The equivalent quaternion representation of this rotation is defined by

$$\begin{aligned}
q &= q_\beta . q_\alpha . q_\gamma \\
&= \left(\cos\frac{\beta}{2} + j\sin\frac{\beta}{2}\right)\left(\cos\frac{\alpha}{2} + i\sin\frac{\alpha}{2}\right)\left(\cos\frac{\gamma}{2} + k\sin\frac{\gamma}{2}\right) \\
&= \left(\cos\frac{\beta}{2}\cos\frac{\alpha}{2}\cos\frac{\gamma}{2} + \sin\frac{\beta}{2}\sin\frac{\alpha}{2}\sin\frac{\gamma}{2}\right) \\
&+ i\left(\cos\frac{\beta}{2}\sin\frac{\alpha}{2}\cos\frac{\gamma}{2} + \sin\frac{\beta}{2}\cos\frac{\alpha}{2}\sin\frac{\gamma}{2}\right) \\
&+ j\left(\sin\frac{\beta}{2}\cos\frac{\alpha}{2}\cos\frac{\gamma}{2} - \cos\frac{\beta}{2}\sin\frac{\alpha}{2}\sin\frac{\gamma}{2}\right) \\
&+ k\left(\cos\frac{\beta}{2}\cos\frac{\alpha}{2}\sin\frac{\gamma}{2} - \sin\frac{\beta}{2}\sin\frac{\alpha}{2}\cos\frac{\gamma}{2}\right).
\end{aligned} \tag{6.18}$$

The conversion from Euler angles to quaternions offers an effective way of sampling quaternions for viewpoint regression. However, the converted quaternions would suffer from the gimbal lock problem if the original Euler angle representation suffers from the gimbal lock problem. To avoid this problem, we can use the method proposed by Deserno [14] to generate equidistant points on a unit sphere.

Generating uniformly distributed points on the surface of a unit sphere

Deserno [14] proposed a method for generating equidistant points on the surface of a unit sphere using the spherical coordinates where a point on the sphere is defined by its azimuth and the elevation angles. In the setting of our experiments, the azimuth angle corresponds to the rotation around the z-axis (γ) and the elevation angle corresponds to the rotation around the x-axis (α). Thus any point $M(x, y, z)$ on the unit sphere can be represented by

$$\begin{cases} x = \sin\alpha\cos\gamma \\ y = \sin\alpha\sin\gamma \\ z = \cos\alpha. \end{cases} \tag{6.19}$$

Equidistant points on the surface of the sphere are obtained by methodically sampling the azimuth and Elevation angles in Equation (6.19). To generate N points from the surface of the unit sphere, the azimuth and elevation angles are generated using Algorithm 1.

Algorithm 1 Sampling N equidistant points from the surface of a unit sphere [14].

```
$N_{counter} \leftarrow 0$
$a \leftarrow 4\pi/N$
$d \leftarrow \sqrt{a}$
$M_\alpha \leftarrow \text{round}(\pi/d)$
$d_\alpha \leftarrow \pi/M_\alpha$
$d_\gamma \leftarrow a/d_\alpha$
for $m \in [0, M_\alpha - 1]$ do
    $\alpha = \pi(m + 0.5)/M_\alpha$
    $M_\gamma = \text{round}(2\pi \sin\alpha/d_\gamma)$
    for $n \in [0, M_\gamma - 1]$ do
        $\gamma = 2\pi n/M_\gamma$
        $N_{counter} = N_{counter} + 1.$
    end for
end for
```

The distribution of the points sampled using Algorithm 1 is illustrated in Figure 6.6. We use $N = 1000$ in Algorithm 1 to generate the points displayed on Figure 6.6. Although we used $N = 1000$, only 998 points were generated, since the algorithm outputs the closest number to N such that the distance between any adjacent points is approximately the same.

Converting azimuth-elevation angles to unit quaternions

We convert the azimuth and elevation angles generated by Algorithm 1 to unit quaternion using the method described in Section 6.3.1. We set the third Euler angle (rotation angle around the y-axis) to zero. Thus the converted unit quaternions are obtained by substituting $\beta = 0$ in Equation (6.18). The resulting unit quaternions are defined by

$$q = \left(\cos\frac{\alpha}{2}\cos\frac{\gamma}{2}\right) + i\left(\sin\frac{\alpha}{2}\cos\frac{\gamma}{2}\right) - j\left(\sin\frac{\alpha}{2}\sin\frac{\gamma}{2}\right) + k\left(\cos\frac{\alpha}{2}\sin\frac{\gamma}{2}\right). \tag{6.20}$$

Converting azimuth-elevation and in-plane rotations unit quaternions

The quaternion sampling method presented in Section 6.3.1 only uses the azimuth and elevation angles to generate the unit quaternions. Thus the in-plane rotation angle is ignored. In order to include the in-plane rotation angle, we use a modified version of Algorithm 1. In Algorithm 2, we not only sample the azimuth and eleva-

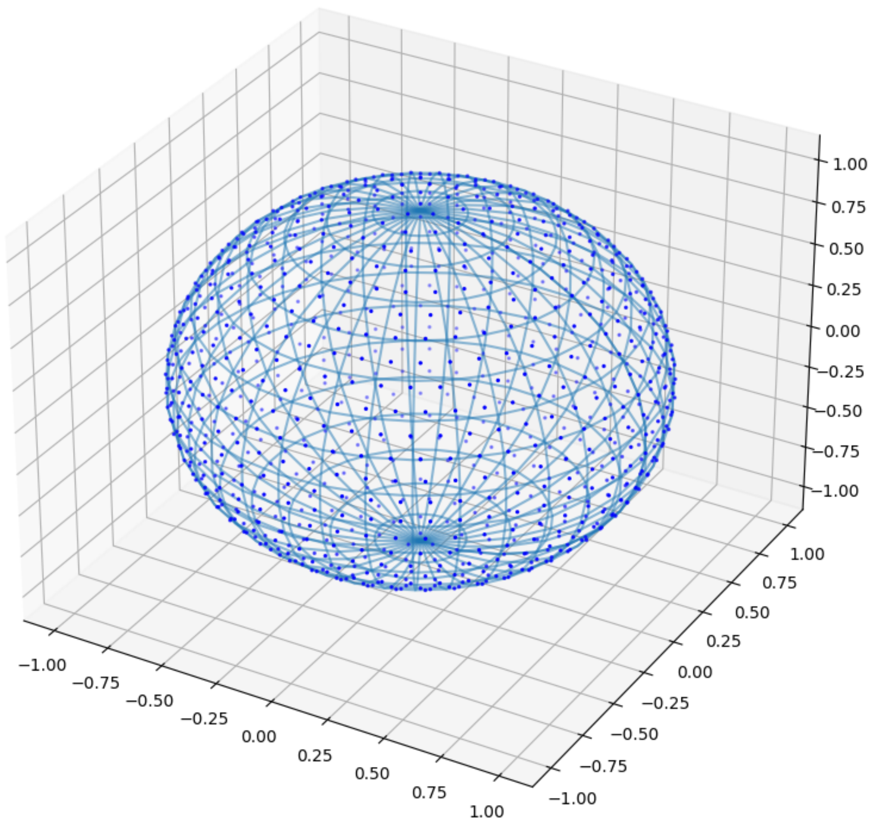


Figure 6.6: Distribution of points on a unit sphere using the Algorithm 1.

tion angles such that the corresponding points on a unit sphere are equidistant but also include coarse in-plane rotations in the range $[-20, 20]$ at 10-degree increments.

Using Algorithm 2, we sample five in-plane rotation angles for each pair of azimuth-elevation angles. However, the in-plane rotation sampling is coarse and its range is limited to $[-20, 20]$ degrees.

Sampling unit quaternions directly

One alternative to generating unit quaternions from Euler angles is to sample quaternions directly from the viewpoint space. Here we generate unit quaternions by sampling independently and uniformly points on a unit 4D sphere as proposed by Marsaglia [58], as described in Algorithm 3.

We can check that the quaternion returned by Algorithm 3 is indeed a unit quater-

Algorithm 2 Sampling N equidistant points from the surface of a unit sphere with in-plane rotation.

```
$N_{counter} \leftarrow 0$
$a \leftarrow 4\pi/N$
$d \leftarrow \sqrt{a}$
$M_\alpha \leftarrow \text{round}(\pi/d)$
$d_\alpha \leftarrow \pi/M_\alpha$
$d_\gamma \leftarrow a/d_\alpha$
for $m \in [0, M_\alpha - 1]$ do
    $\alpha = \pi(m + 0.5)/M_\alpha$
    $M_\gamma = \text{round}(2\pi \sin\alpha/d_\gamma)$
    for $n \in [0, M_\gamma - 1]$ do
        $\gamma = 2\pi n/M_\gamma$
        $N_{counter} = N_{counter} + 1$
        for $l \in \{-20, -10, 0, 10, 20\}$ do
            $\beta = \pi l/180.$
        end for
    end for
end for
```

Algorithm 3 Uniform sampling of unit quaternions [58].

```
$S_1 \leftarrow \infty$
$S_2 \leftarrow \infty$
while $S_1 > 1$ do
    choose uniformly $a \in (-1, 1)$
    choose uniformly $b \in (-1, 1)$
    $S_1 \leftarrow a^2 + b^2$
end while
while $S_2 > 1$ do
    choose uniformly $c \in (-1, 1)$
    choose uniformly $d \in (-1, 1)$
    $S_2 \leftarrow c^2 + d^2$
end while
$q_w \leftarrow a$
$q_x \leftarrow b$
$q_y \leftarrow c\sqrt{\frac{1-S_1}{S_2}}$
$q_z \leftarrow d\sqrt{\frac{1-S_1}{S_2}}$
return $(q_w, q_x, q_y, q_z)$
```

nion. Let $q = (q_w, q_x, q_y, q_z)$. Then

$$\begin{aligned} \|q\|^2 &= q_w^2 + q_x^2 + q_y^2 + q_z^2 \\ &= a^2 + b^2 + c^2\left(\frac{1-S_1}{S_2}\right) + d^2\left(\frac{1-S_1}{S_2}\right) \\ &= a^2 + b^2 + (c^2 + d^2)\left(\frac{1-S_1}{S_2}\right) \\ &= S_1 + S_2\left(\frac{1-S_1}{S_2}\right) \\ &= 1. \end{aligned} \tag{6.21}$$

Hence q is indeed a unit quaternion by construction.

Figure 6.7 illustrates the distribution of 1000 sampled viewpoints on a unit sphere using Algorithm 3. As can be seen, the viewpoints are well distributed on the surface of the sphere, which prevents any part of the viewpoint space to be oversampled.

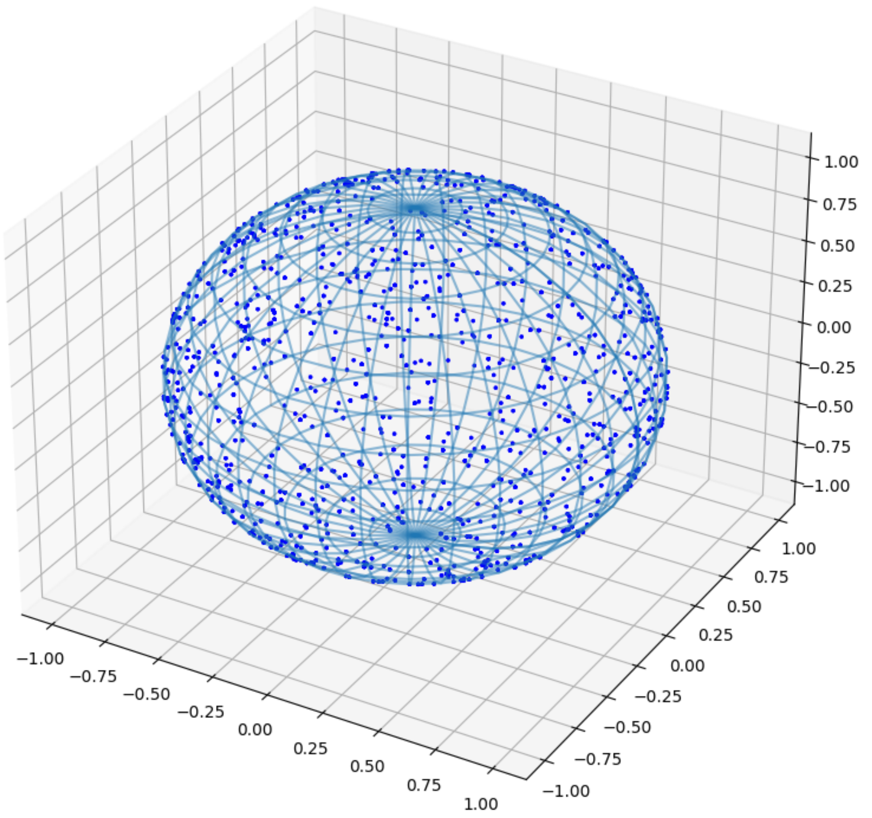


Figure 6.7: Distribution of points on a unit sphere using the Algorithm 3.

6.3.2 Model architecture

The viewpoint regression model is based on the CNN model used for 1D viewpoint classification. The architecture of the CNN-based regression model is presented in

Figure 6.8. The main difference with the classification model is that the output FC layer for 360-class classification is replaced by another FC layer with four units, followed by a linear activation function instead of the softmax activation function for the classification model. The output is passed through an L2 normalization layer to convert the raw output quadruplet to a unit quaternion.

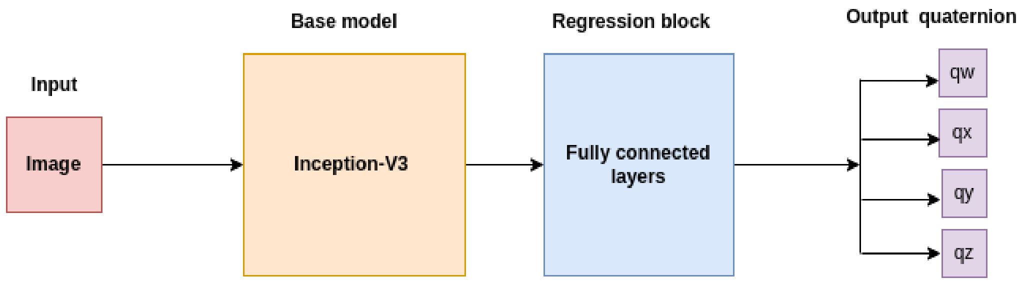


Figure 6.8: CNN-based regression model for 3D viewpoint estimation using quaternion representation.

6.3.3 Model optimization

A regression model is typically trained by minimizing some objective function that measures the distance between a predicted value and a target value. This distance function is often the mean squared error (MSE), the mean average error (MAE), or another similar error function. Since we are predicting quaternions, the cost function must be adapted to measure the distance between the predicted quaternion and the target quaternion. Thus the cost function used to train the regression model is

$$L(q, \hat{q}) = 1 - (q \cdot \hat{q})^2, \tag{6.22}$$

where q and $\hat{q}$ are the ground truth and the predicted quaternions respectively. This cost function is minimized using an SGD optimizer with a learning rate that decays exponentially during training. The model is trained until convergence or until the performance on the validation set stops improving. The resulting model is then evaluated on the test set.

6.3.4 Experiments and results

In this section we present the experiments related to viewpoint estimation using a regression model with quaternion representation. For each of these experiments, we use one of the unit quaternion sampling methods presented above.

Azimuth-elevation to unit quaternions

In this experiment we sampled azimuth-elevation angle pairs such that the corresponding points are uniformly distributed on a unit sphere. Here, the in-plane rotation angle is set to zero. We generated a total of 998 azimuth-elevation pairs, where each pair corresponds to a particular viewpoint. The training data is generated by projecting a CT scan volume at each of the 998 viewpoints using the Fourier volume rendering (FVR) method. The training data are generated from 21 CT scans, which results in a total of 20,958 training images. During model training, a translational cropping strategy is used for data augmentation. Image patches of size 400×400 are extracted from the original images of size 512×512 by randomly moving a sliding window in the range $[-20, 20]$ pixels. We used a similar approach to generate the test data. We generated 998 images from each of the eight CT scans reserved for test data, which results in a total of 7,984 images. We then extract the central patch of size 400×400 from each of the original images of size 512×512.

The CNN-based regression model presented in Section 6.3.2 is trained by minimizing a loss based on the distance between unit quaternions. We used 20% of the training data as a validation set to monitor the performance of the model during training. We train the model until its performance of the validation set stops improving. The generalization performance is then evaluated on the test set. This CNN-based regression model trained using the unit quaternions converted from the azimuth-elevation angle pairs performs well with an accuracy $Acc_{\frac{\pi}{6}} = 94.98\%$ and a median error $MedErr = 4.77$. A graphical illustration of the performance is presented in Figure 6.9 with the plot of viewpoint estimation accuracy.

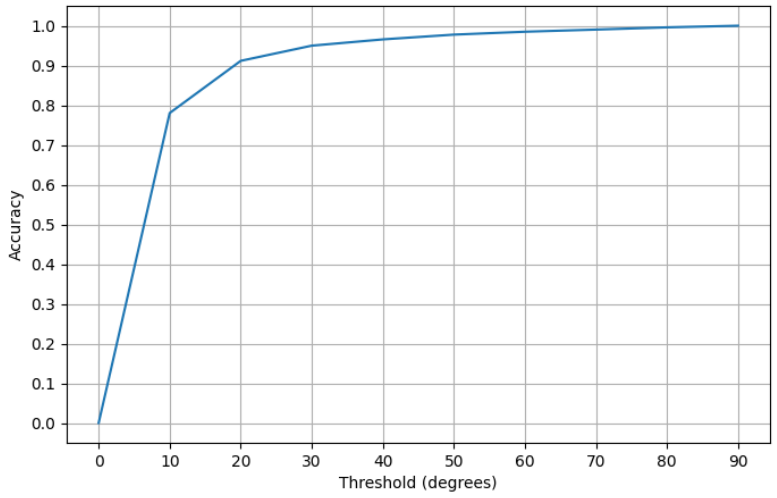


Figure 6.9: Accuracy of the CNN-based regression model using azimuth-elevation to quaternion conversion

Although the data generated using azimuth-elevation angles to unit quaternion conversion perform well on the test set, the generalization of the model could be limited on other datasets. In effect, by fixing the in-plane rotation angle at zero, the variability of the viewpoints is limited. Consequently, the model might perform poorly on other datasets where the in-plane rotation is not kept fixed.

Euler angles to unit quaternions

To improve the generalizability of the viewpoint estimation model, we augmented the training data to include in-plane rotation. Here we represent the viewpoint using unit quaternions converted from all three Euler angles, as described in Section 6.3.1. For each of the training CT scans, we sampled 998 azimuth-elevation angle pairs. For each pair of azimuth-elevation angles, we sampled five in-plane rotation angles ($\beta \in \{-20°, -10°, 0°, 10°, 20°\}$). This results in a total of 4,990 triplets of Euler angles, each corresponding to a particular viewpoint. We generated the training images by projecting each of the 21 training CT scans from one of the 4,990 viewpoints. Thus, the training data is made of 104,790 images. During training, the translational cropping technique is used to augment the data by extracting patches of size 400×400 from the original images of size 512×512. The test data is obtained by projection of the 8 CT scans at each of the 4,990 viewpoints, resulting in a total of 39,920 test images.

We trained the CNN-based regression model presented in Section 6.3.2 by minimizing the loss that measures the distance between the ground truth quaternion and the quaternion predicted by the model. We reserved 20% of the training data as a validation set to track the performance of the model during training. The performance of the model, evaluated on the test set, proves the effectiveness of the method with accuracy $Acc_{\frac{\pi}{6}} = 91.82\%$ and a median error $MedErr = 6.49$. The accuracy of the model is illustrated in Figure 6.10.

The generalization of the method presented in this section is still limited since the training data only includes images with coarse in-plane rotations. In the next section we extend the method by generating training data with unit quaternions sampled directly from the viewpoint space.

Direct sampling of unit quaternions

We sampled unit quaternions directly from the viewpoint space using the method presented in Section 6.3.1. To investigate the influence of the density of viewpoint sampling, we experimented with four quaternion sample sizes: 1000, 2000, 5000, and 20,000. Each quaternion corresponds to a viewpoint. We generate the training

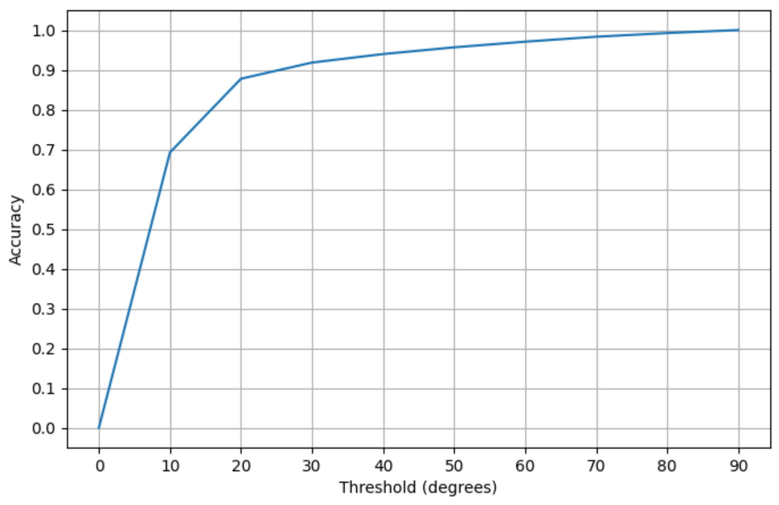


Figure 6.10: Accuracy of the CNN-based regression model using Euler angles to quaternion conversion

and test data by projecting a CT volume from a particular viewpoint.

After training, the best-performing model on the validation set is restored for prediction on the test set. The performance of the restored model is evaluated on the test set. The accuracy and median errors measured on the test set are reported in Table 6.1.

Table 6.1: Accuracy and median errors for the CNN-based regression model using direct sampling of unit quaternions. We present the results for the different number of quaternion sample sizes.

Sample size	$Acc_{\pi/6}$ (%)	$MedErr$ (degrees)
1000	79.31	11.44
2000	80.27	11.95
5000	87.87	6.90
20000	90.20	5.05

A graphical illustration of the results (measured in terms of the accuracy at θ) is presented in Figure 6.11. It can be noted from the results reported in Figure 6.11 that, contrary to the Euler angle-based regression case, the quaternion-based regression model performs well.

This improvement could be attributed to the fact that angle interpolation using quaternions is smoother and easier than interpolation with Euler angles since regression performance depends on the quality of the interpolation. Moreover, we investigate the effectiveness of a classification model for viewpoint estimation but using a quaternion representation instead of one using Euler angles.

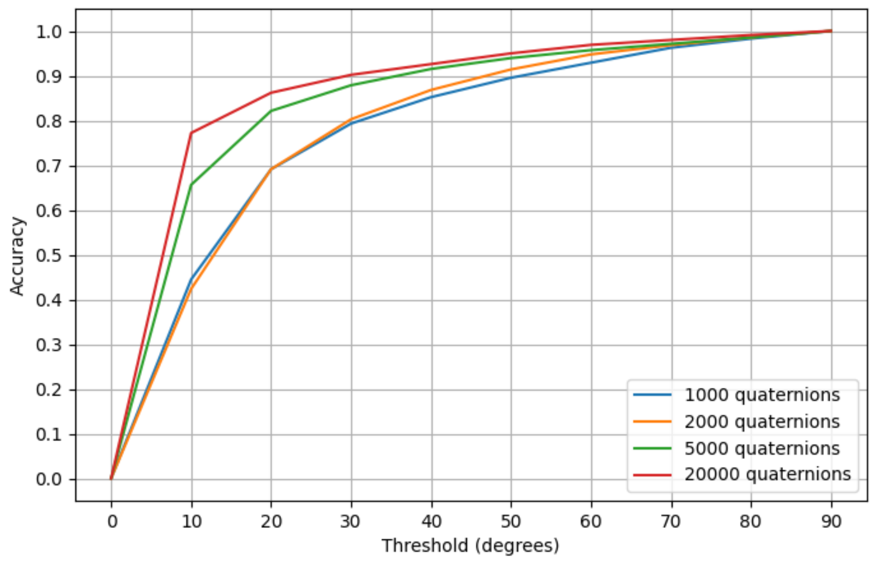


Figure 6.11: Accuracy of the CNN-based regression models for a varying number of sampled quaternions.

6.4 Classification of viewpoint using quaternions

Considering the continuous nature of the viewpoint space, it is natural to use a regression model for viewpoint prediction in 3D space. However, classification models have proven to be more effective at predicting viewpoint in the 1D case for Euler angle prediction. In this section we investigate the effectiveness of CNN-based classification models for viewpoint estimation when viewpoints are represented by unit quaternions.

6.4.1 Class definition

The number of different unit quaternions in the viewpoint space is infinite. As a classification model requires a finite number of classes, we need to sample a finite number of unit quaternions that will represent the viewpoint classes. Given the spherical distribution of viewpoints, the classes need to be carefully sampled such that the corresponding viewpoints cover the body. To this end, we use uniform sampling on a unit 4D sphere as proposed by Marsaglia [58]. Algorithm 3 summarizes the quaternion sampling method used to generate the viewpoint classes. The coverage of the viewpoint space depends on the number of viewpoints sampled. A high number of samples will lead to a denser viewpoint distribution. As the viewpoint distribution gets denser, the error made by predicting a neighboring class instead of the ground truth class decreases because the distance between classes is reduced. However, the number of parameters in the classification model increases because the dimension of the model's output (which is defined by the number of classes) increases. This increase in the dimension of the output (and therefore the number of model parameters) would make the classification model more difficult to train as it will require much more data due to the curse of dimensionality. Therefore, when sampling unit quaternions for viewpoint classification, one needs to make a trade-off between minimizing the prediction error by using a dense viewpoint distribution, and the model output dimension that can lead to underfitting due to the curse of dimensionality. We experiment with a number of output classes, starting from a coarse distribution and increasing the density of the viewpoint distribution gradually. Figure 6.12 illustrates the distribution of the viewpoint on a 3D sphere for a different number of classes. As can be seen in this figure, the viewpoint gets denser with an increased number of classes. Sample images generated from quaternion class locations are depicted in Figure 6.13.

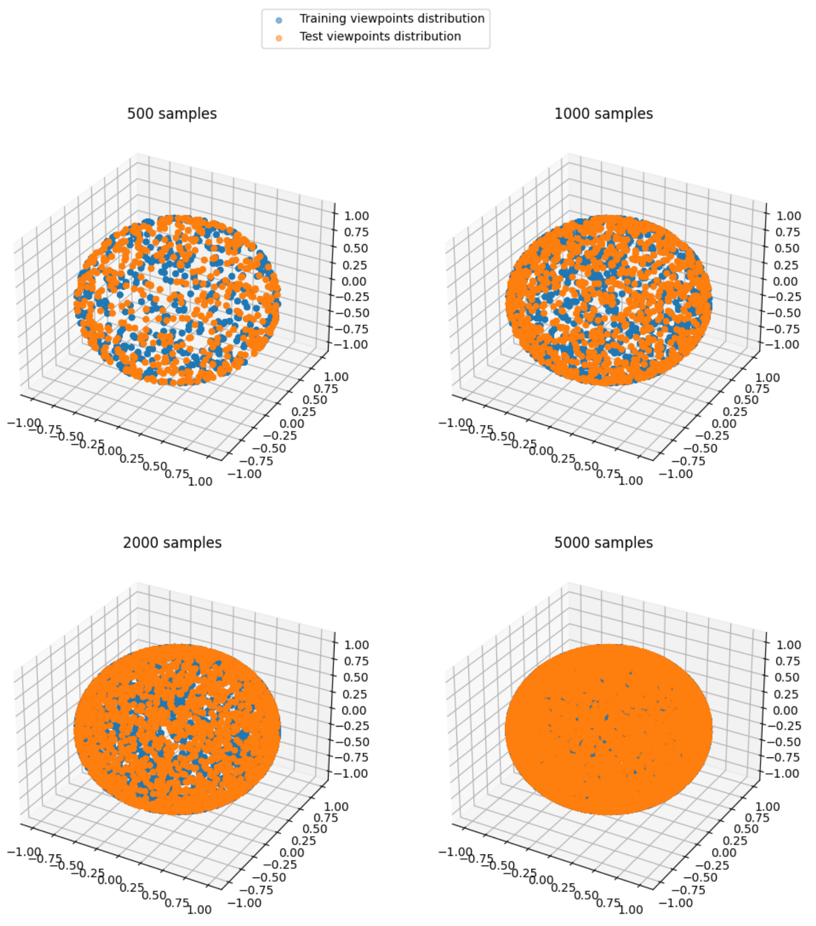


Figure 6.12: Viewpoint distribution on a unit sphere.

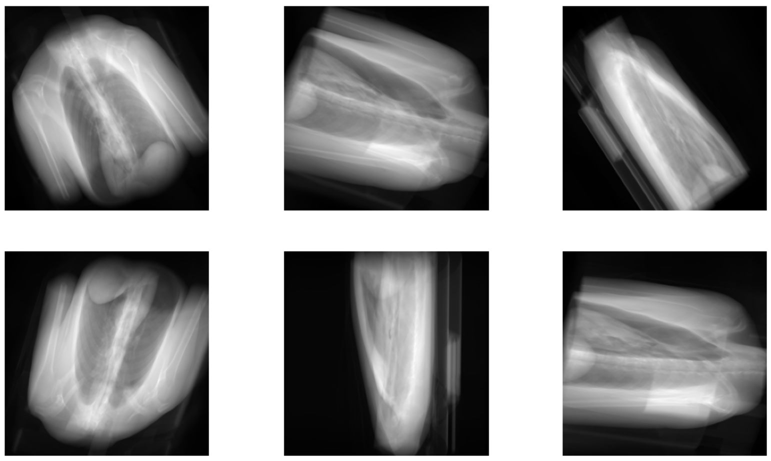


Figure 6.13: Examples of images generated from the sampled quaternion classes.

6.4.2 Model architecture and optimization

We use several CNN-based classification models to train the viewpoint estimator using a quaternion representation. The general architecture of these classification models is depicted in the diagram in Figure 6.14. All these models have the same base model (Inception-V3 [81]). The base model, pre-trained on ImageNet, extracts features from the raw images. These features are passed through a classification block made of fully-connected layers. The number of neurons in the last FC layer is determined by the number of classes. The classification model is trained by minimizing the geometric structure-aware cross-entropy loss defined by

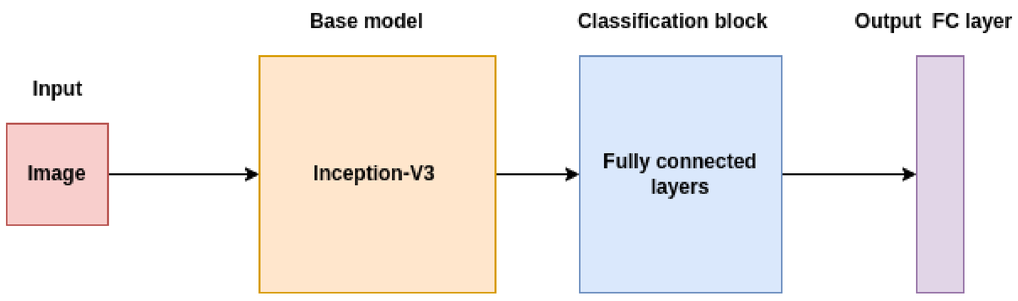


Figure 6.14: CNN-based classification model for 3D viewpoint estimation using quaternion representation.

$$L = -\sum_{k=1}^{N} \exp\left(-\frac{\Delta(k_{gt}, k)}{\sigma}\right) \log(p_k), \tag{6.23}$$

where k_{gt} is the ground truth viewpoint, σ is a hyperparameter, p_k is the probability of viewpoint k, and $\Delta(k_{gt}, k) = 1 - |q_{gt} \cdot q_k|$.

The models are trained using images generated at sampled viewpoints locations. Thus for each of the twenty CT scans that constitute the training data, N images are generated for the N viewpoint classes. Each original image (from a sampled viewpoint location) is augmented with small translations in the interval $[-20, 20]$ pixels. Similarly, N images are generated from the eight CT scans that represent the test data, at newly sampled viewpoint locations that do not necessarily correspond to the training class locations.

For hyperparameter tuning and model selection after training, 20% of the training set is used as a validation set. Each model is trained until the performance on the validation set stops improving.

6.4.3 Results and discussion

The best-performing model on the validation set is restored for prediction and its performance is evaluated on the test set. The accuracy at θ is used as an evaluation metric. The results obtained for the different numbers of classes are summarized in Figure 6.15 and Table 6.2. The results show that classification models are effective at predicting viewpoints using a quaternion representation.

Table 6.2: Accuracy and median errors for the CNN-based classification model using direct sampling of unit quaternions. We present the results for the different number of quaternion sample sizes.

Sample size	$Acc_{\pi/6}$ (%)	$MedErr$ (degrees)
1000	74.16	10.51
2000	82.33	7.31
5000	90.90	4.84
20,000	89.48	3.65

6.4.4 Comparison of regression and classification methods

The accuracy for the classification and the regression models, for different numbers of sampled quaternions, are compared in Figure 6.16. As the number of sampled quaternions increases, the accuracy tends to improve for both the classification and the regression models.

The median errors for the classification and the regression models, for different numbers of sampled quaternions, are compared in Figure 6.17. For both the classification and the regression models, the median error reduces, as the number of sampled quaternions increases. This performance improvement can be attributed to the fact that the viewpoint space gets denser as the number of samples increases.

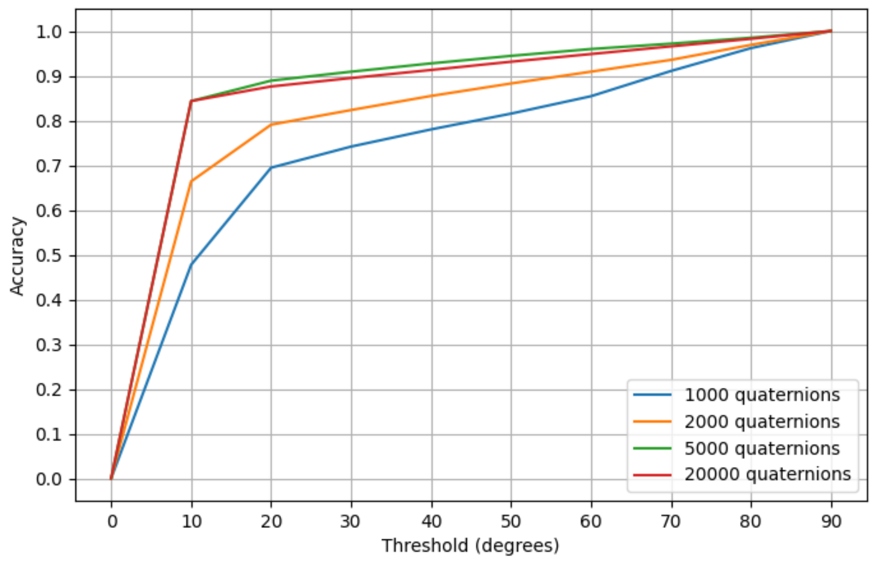


Figure 6.15: Accuracy of the CNN-based classification models for different number quaternion classes.

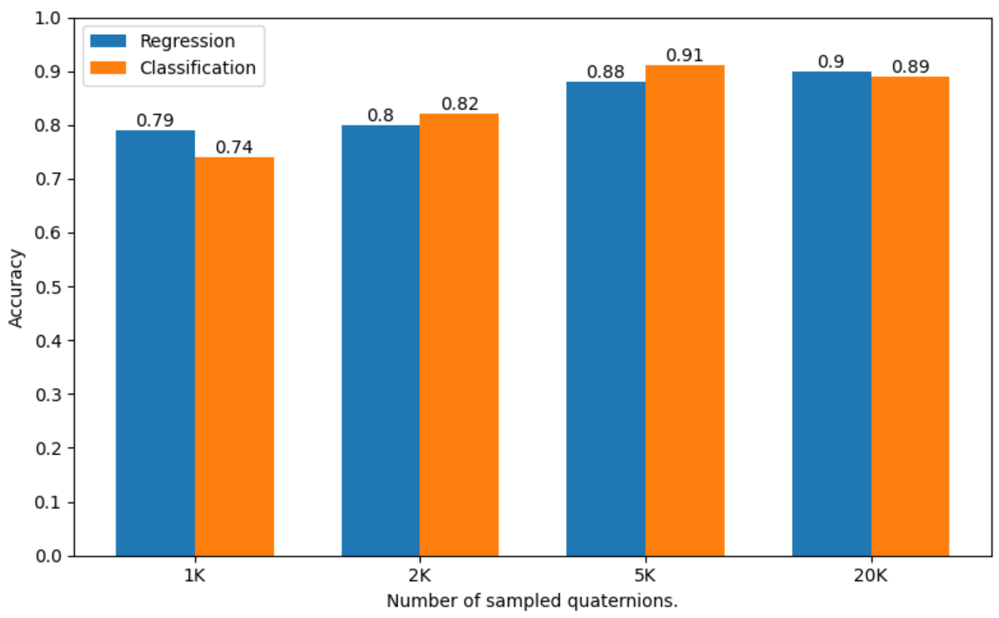


Figure 6.16: Comparison of the accuracy of classification and regression models.

6.5 Summary

In this chapter we extended the method for 1D viewpoint estimation to do full 3D viewpoint estimation. We show that the geometric structure-aware cross-entropy loss can be successfully used to train a viewpoint estimation model for viewpoint estimation even when the viewpoints are represented by unit quaternions. We show the effectiveness of using quaternions for viewpoint regression by training a

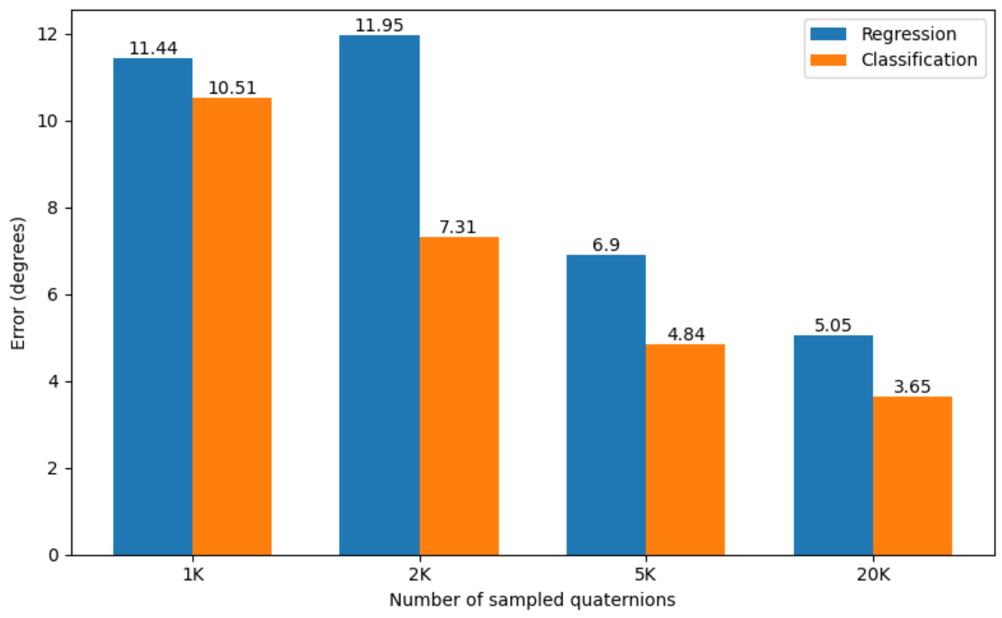


Figure 6.17: Comparison of the median errors of classification and regression models.

CNN-based regression model to directly predict viewpoints in 3D. Moreover, by sampling unit quaternions uniformly around the viewpoint space, we developed a classification-based viewpoint estimation model. The classification model is trained using the geometric structure-aware loss, where the cross-entropy loss is weighted by the quaternion distance between classes. The results show that the classification model performs better than the regression model when trained on the same amount of training data as presented in Figure 6.17.

Chapter 7

Viewpoint estimation for natural images

In the previous chapters we investigated how the viewpoint estimation problem can be solved for medical images. We showed that it is possible to generate enough images from a limited set of CT scans to train CNN models for viewpoint estimation. Given a random 2D image projected from a 3D volume, the trained models are able to predict the viewpoint from which the image was projected with reasonable precision. The proposed models are based on a combination of methods: the data generation strategy, the use of quaternions to represent rotations instead of Euler angles, and the use of a cost function adapted to the viewpoint estimation problem instead of the usual cross-entropy loss that is used for general classification tasks. The results show the effectiveness of the proposed method to solve the viewpoint estimation problem for medical images.

Viewpoint estimation is of undeniable importance in medical imaging as it can be used in a computer-aided diagnosis system. For example, it can be used to fuse information from a 3D pre-operative volume and 2D intra-operative images for guidance during surgery. However, the application of viewpoint estimation is not limited to medical imaging, since it can also be applied to natural images. There are several applications for viewpoint estimation for natural images. Most notably, viewpoint estimation can be used to gain an understanding of 3D scenes from 2D images. In robotics, viewpoint estimation of natural images can be used to understand a robot's surroundings and aid its navigation.

In this chapter we investigate whether the viewpoint estimation method proposed for medical images can also be successfully applied to natural images. Although the problems of viewpoint estimation for medical and natural images are similar by definition, there are some particularities that make them different in practice. As the 2D images used for medical image viewpoint estimation are projected from the same type of object, namely a CT scan of the human body, their diversity is less than the diversity of natural images. In effect, unlike X-ray images that generally are taken from one specific object (e.g. a human body part), natural images are more diverse as they could be taken from different types of objects. Also, these images are generally acquired by different sensors, which increases their variability in addition to various exterior factors such as lighting conditions and the interaction between the objects of interest and other objects in the scene. Therefore, it is not guaranteed that a method developed to solve a problem in medical imaging would extend to natural images.

In Section 7.1 we present the dataset used to develop the viewpoint estimation methods for natural images. In Section 7.2 we present the regression-based viewpoint estimation method. The classification-based viewpoint estimation method is investigated in Section 7.3. In Section 7.4 we compare the different methods. A summary of the chapter is presented in Section 7.5.

7.1 Datasets for viewpoint estimation

Datasets for viewpoint estimation are very rare compared to those for 2D image classification. The scarcity of datasets for viewpoint estimation is due to the specificity of the required labels for this task. In effect, a supervised learning-based viewpoint estimation model needs to be trained with accurate viewpoint labels of the object of interest, which is not an easy task. Here we use the PASCAL3D+ [89] dataset augmented with data from ImageNet [42] and synthetic images.

7.1.1 PASCAL3D+ dataset

PASCAL3D+ is an extension of the PASCAL VOC 2012 [18] dataset that was initially released for image classification and object detection of twelve different object categories: aeroplane, bicycle, boat, bottle, bus, car, chair, dining table, motorbike, sofa, train, and TV monitor. This dataset is split into 1,464 training images and 1,449 test images. The PASCAL VOC dataset was labeled with object categories and bounding annotations that are required for object classification and detection. The labels were later augmented with viewpoint annotations for each object category in the dataset by aligning the objects in the images with 3D reference CAD models and estimating the viewpoint of objects with respect to the CAD model. Moreover, the PASCAL VOC images for object detection are augmented with images from ImageNet. Thus, given an image in the dataset, all the objects of

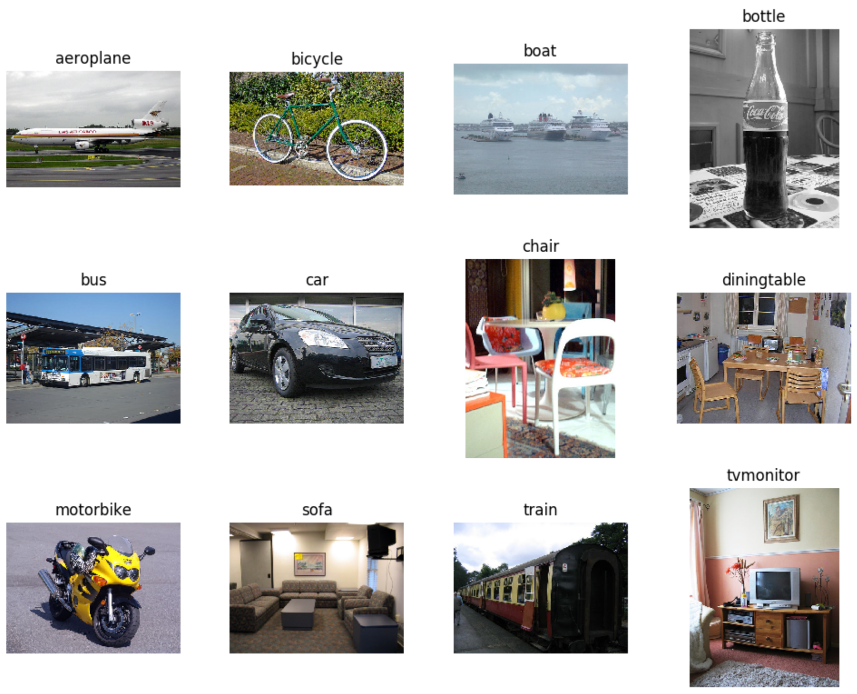


Figure 7.1: Sample images from PASCAL3D+ dataset.

interest are labeled with full 3D pose information that is represented by orientation and position parameters. The orientations are parametrized with three Euler angles: azimuth, elevation, and in-plane rotation. The position of each object with respect to the camera is represented with their coordinates (x, y, z). Thus the full 3D pose of an object is defined by six parameters $(x, y, z, \theta, \phi, \rho)$. Here we are interested in estimating the orientation parameters of the 3D pose.

The ImageNet dataset is a collection of very diverse object categories including the object categories in the PASCAL3D+ dataset. In order to increase the size of the PASCAL3D+ dataset, images from ImageNet with the same object categories as in PASCAL3D+ were selected. Since the ImageNet dataset was intended for an object recognition task, it was not labeled with object viewpoints. Thus the selected images from ImageNet were subsequently annotated with viewpoint information using the same procedure as for PASCAL3D+ images. A sample of images from the PASCAL3D+ dataset is presented in Figure 7.1.

7.1.2 Data augmentation by 3D pose jittering

The dataset is further augmented using small variations in the object's pose in the images. In particular, each object in the images was subjected to small in-plane rotation in the range $[-4, 4]$ degrees at 1-degree increments, and small out-of-plane rotation in the range $[-2, 2]$ degrees at 0.5-degree increments. Moreover, each image in the dataset is flipped horizontally which results in a doubling of the size of the dataset. This 3D pose jittering method for data augmentation, proposed by Mahendran et al. [56], increased the size of the dataset from 22 thousand to almost 3 million. Table 7.1 summarizes the size of the dataset for each object category.

Table 7.1: PASCAL3D+ dataset.

	Train-val	Train-val (augmented)	Test
Aeroplane	1,976	336,548	285
Bicycle	1,342	148,312	119
Boat	2,597	409,099	245
Bottle	1,527	260,104	259
Bus	1,084	191,826	154
Car	5,632	917,332	315
Chair	1,053	195,236	248
Dining table	2,313	121,951	21
Motorbike	1,257	125,162	138
Sofa	1,458	107,502	39
Train	1,308	181,506	113
TV monitor	1,252	223,682	222
Total	22,799	2,850,070	4,316

7.1.3 Synthetic dataset

The amount of data required to train machine learning models increases with the size of the model. Thus training a deep CNN model for viewpoint estimation generally requires more data than is available in the PASCAL3D+ dataset. Su et al. [79] proposed a method for generating synthetic images of all object categories in the PASCAL3D+ dataset. These images were annotated with accurate viewpoint information. In total, more than 2 million synthetic images were generated to complement the real images from the PASCAL3D+ dataset. Table 7.2 details the numbers of synthetic images generated per object category.

7.2 Regression-based viewpoint estimation

Given the continuity of the viewpoint space, a regression approach is the most natural fit to solve the viewpoint estimation problem. In this section we propose a CNN model to regress viewpoints represented by unit quaternions on the PASCAL3D+ dataset. Since the viewpoints on PASCAL3D+ are parametrized using Euler angles, we must first convert the original viewpoint labels to unit quaternions before the regression model can be trained.

Table 7.2: Synthetic images used to augment the training images.

Aeroplane	198,201
Bicycle	199,544
Boat	198,940
Bottle	199,641
Bus	198,961
Car	194,919
Chair	196,550
Dining table	195,685
Motorbike	199,765
Sofa	199,888
Train	199,712
TV monitor	199, 659
Total	2,381,565

7.2.1 Converting Euler angles to unit quaternions

The regression model used requires both the input label and the output prediction to have the same viewpoint parametrization. In the PASCAL3D+ dataset, images are labeled with Euler angles (azimuth, elevation, and in-plane rotation). Since we use unit quaternions to represent viewpoints, we need to convert the original Euler angles to unit quaternions. To convert the Euler angles labels to unit quaternions, we use the expression

$$\begin{aligned} q = &\left(\cos\frac{\beta}{2}\cos\frac{\alpha}{2}\cos\frac{\gamma}{2} + \sin\frac{\beta}{2}\sin\frac{\alpha}{2}\sin\frac{\gamma}{2}\right) \\ &+ i\left(\cos\frac{\beta}{2}\sin\frac{\alpha}{2}\cos\frac{\gamma}{2} + \sin\frac{\beta}{2}\cos\frac{\alpha}{2}\sin\frac{\gamma}{2}\right) \\ &+ j\left(\sin\frac{\beta}{2}\cos\frac{\alpha}{2}\cos\frac{\gamma}{2} - \cos\frac{\beta}{2}\sin\frac{\alpha}{2}\sin\frac{\gamma}{2}\right) \\ &+ k\left(\cos\frac{\beta}{2}\cos\frac{\alpha}{2}\sin\frac{\gamma}{2} - \sin\frac{\beta}{2}\sin\frac{\alpha}{2}\cos\frac{\gamma}{2}\right), \end{aligned} \tag{7.1}$$

where α, β, and γ are the elevation, in-plane rotation, and azimuth angles respectively. With the new viewpoint annotations, we train the regression model to predict unit quaternions.

7.2.2 Model architecture

The models used to regress viewpoint are based on a classification model pre-trained on ImageNet, namely VGG-16. The VGG-16 model is used to compare the results with a baseline method proposed by Mahendran et al. [56]. This pre-trained model is used to extract image features, which are then passed to a regression block made

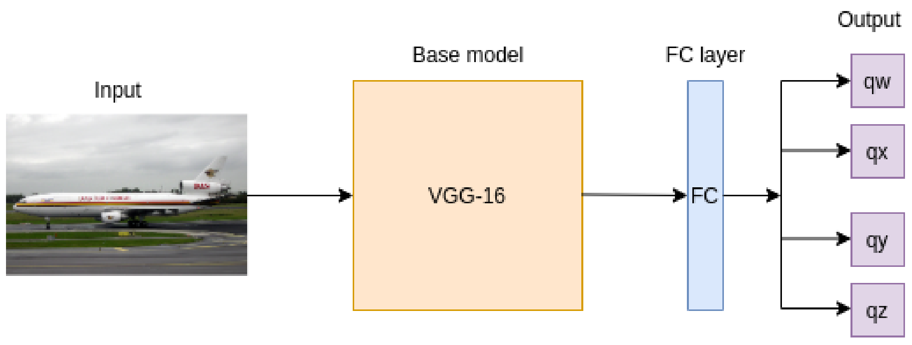


Figure 7.2: Block diagram of the CNN-based regression model.

of two fully-connected (FC) layers. The last FC layer is made of four neurons, representing the components of the quaternion, without a non-linear activation function. Furthermore, the output of the model is passed to an L2 normalization function to obtain unit quaternions. A total of twelve category-specific models with the same architecture were trained for the twelve object categories in the PASCAL3D+ dataset. Figure 7.2 illustrates the architecture of the CNN-based regression model proposed.

7.2.3 Model training

The training data is made of the original training set from PASCAL3D+ with pose jittering data augmentation, as described in Section 7.1.2. This training data is also augmented with synthetic images. We use 20% of the training data as a validation set for performance monitoring and model selection.

The regression models are trained with a cost function based on the distance between unit quaternions. All category-specific models and the global model are trained using the loss

$$L(q, \hat{q}) = 1 - (q \cdot \hat{q})^2 \,. \tag{7.2}$$

The models are trained using a stochastic gradient descent (SGD) optimizer with a decaying learning rate. During training, the performance of the model is evaluated on the validation set after each epoch, and a checkpoint of the weight values is saved. We use an early-stopping strategy where the performance on the validation set is monitored regularly and the training is stopped when performance on the validation set starts degrading. The best-performing model on the validation set is then restored from the checkpoints, and the generalization performance is evaluated on the test set.

7.2.4 Model evaluation and results

After training, the best-performing model on the validation set is restored and evaluated on the test set which is made of the original validation images of the PASCAL3D+ dataset. We use the accuracy and the median error as evaluation metrics. Table 7.3 summarizes the accuracy and the median errors of the different regression models for all twelve object categories.

Table 7.3: Accuracy at $\theta = \frac{\pi}{6}$ and median error for the regression models on PASCAL3D+.

	aero	**bike**	**boat**	**bottle**	**bus**	**car**	**chair**	**table**	**mbike**	**sofa**	**train**	**tv**	**mean**
Accuracy	0.67	0.62	0.44	0.93	0.90	0.80	0.55	0.69	0.63	0.69	0.80	0.77	0.71
Median error	18.44	24.13	36.33	7.88	5.27	8.72	27.04	19.55	23.70	18.71	7.43	16.23	17.79

7.3 Classification-based viewpoint estimation

In previous chapters, classification-based viewpoint estimation models proved to perform better than regression models for medical images. In this chapter we also investigate the effectiveness of classification models for viewpoint estimation of natural images in the PASCAL3D+ dataset. Here we use the unit quaternion representation for 3D rotations, which is more effective at measuring the distance between rotations. Since the images in PASCAL3D+ were labeled with Euler angles, we first need to convert the initial labels to unit quaternions and then define the viewpoint classes.

7.3.1 Viewpoint class definition

For an effective classification, the training data need to be representative of all possible viewpoints around the objects. Oversampling from a few viewpoints could lead to a high class imbalance in the dataset. We must therefore ensure a fair distribution of the viewpoint classes to avoid class imbalance. To this end, we use the method proposed by Deserno [14] to sample 20,000 unit quaternions uniformly. Figure 7.3 shows the distribution of the sampled viewpoints on a sphere using the azimuth and elevation angles. As can be seen, the viewpoints are fairly well distributed on the sphere, which avoids high class imbalance.

The sampled unit quaternions are used as viewpoint class representations. For all twelve object categories in the PASCAL3D+ dataset, each image is assigned a viewpoint class label based on its proximity to the sampled unit quaternions using a nearest-neighbor strategy. Thus, each image in the dataset is assigned to its closest viewpoint among the sampled unit quaternions. All sampled viewpoints that are not

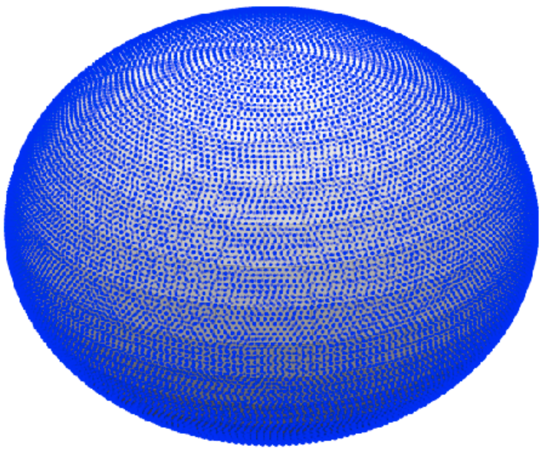


Figure 7.3: Viewpoints sampled on a 3D sphere.

assigned any image in the dataset are discarded. Table 7.4 summarizes the number of viewpoint classes for each object category.

Table 7.4: Number of viewpoint classes per object category. The first column represents the number of viewpoint classes for the dataset of real images, and the second column represents the number of classes for the dataset with synthetic data augmentation.

	Classes (real)	Classes (synthetic)
Aeroplane	10,358	18,603
Bicycle	4,728	14,446
Boat	6,083	13,224
Bottle	6,034	17,579
Bus	1,455	10,698
Car	4,078	10,970
Chair	5,018	13,532
Dining table	2,437	12,522
Motorbike	4,828	12,909
Sofa	2,102	11,994
Train	1,363	10,874
TV monitor	2,887	11,736

7.3.2 Model architecture

Two types of models were used: category-specific models and a single global model.

Category-specific models

For each of the twelve object categories, a CNN-based classification model was trained to predict viewpoints. All category-specific models use a pre-trained VGG-16 model for feature extraction. The extracted features are passed through a classification block made of two fully-connected layers. The first fully-connected layer

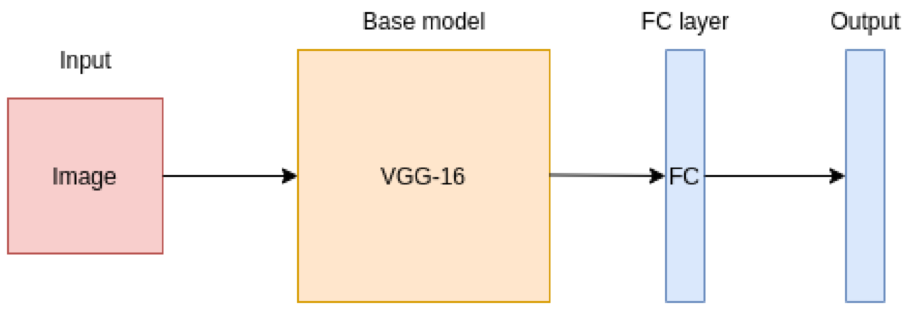


Figure 7.4: Block diagram of the category-specific viewpoint classification model.

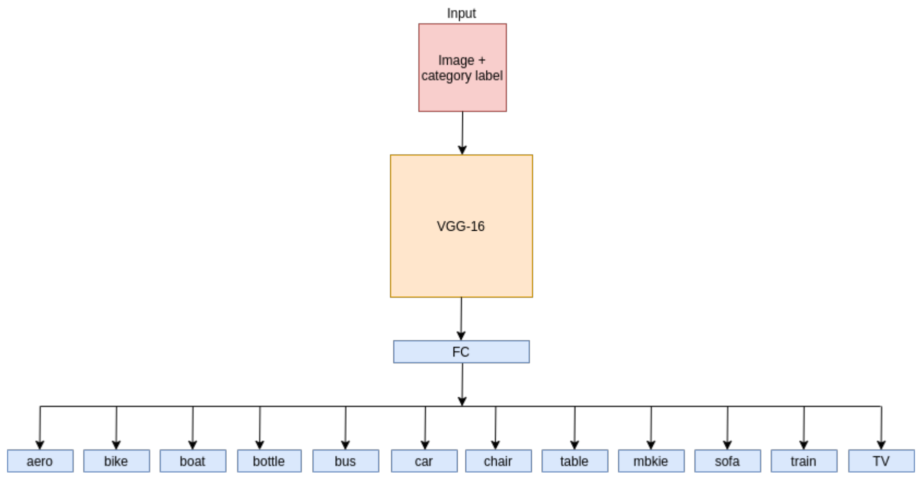


Figure 7.5: Block diagram of the global viewpoint classification model.

is made of 1024 units. The number of units in the last FC layer depends on the number of viewpoint classes, which is specific to each object category. Figure 7.4 illustrates the architecture of the category-specific models.

Global model

Training a separate viewpoint estimation model for each object category is time-consuming and computationally inefficient. In fact, all twelve models have the same architecture and can be trained simultaneously for efficiency. We build a global model that can predict the viewpoint of any of the twelve object categories in the dataset. The global model has a single input layer with the same feature extraction and classification blocks as the category-specific models. However, the global model has twelve outputs that represent the prediction for each object category. Given an input image with an unknown object category, the model outputs twelve candidate viewpoints for all the possible object categories. The actual prediction is retrieved from the candidate predictions based on the object category of the input image. Figure 7.5 illustrates the architecture of the global model.

7.3.3 Evaluation and results

The best-performing models on the validation set are restored for inference and performance evaluation. We evaluate the performance of classification models using the median error metric. The median errors for all classification models are summarized in Table 7.5.

Table 7.5: Median error on PASCAL3D+.

Method	**aero**	**bike**	**boat**	**bottle**	**bus**	**car**	**chair**	**table**	**mbike**	**sofa**	**train**	**tv**	**mean**
3D Pose-Reg. [56] (Real)	16.24	26.81	46.35	**8.47**	4.15	8.76	32.90	26.71	22.20	28.91	**6.36**	17.85	20.48
3D Pose-Reg. [56] (Real + syn.)	14.53	22.55	35.78	9.29	4.28	8.06	19.11	30.62	18.80	13.22	7.32	16.01	16.61
Ours-Class-specific (Real)	12.17	18.52	24.59	9.63	3.94	**5.93**	14.30	**7.69**	22.37	23.14	6.80	16.18	13.77
Ours-Class-specific (Real + syn.)	**10.74**	**14.45**	**22.61**	8.70	**3.69**	5.93	**9.72**	7.74	**15.71**	**9.92**	6.56	**13.83**	**10.80**
Ours-Class-global (Real)	15.66	18.52	33.58	9.11	4.28	6.50	21.61	25.30	19.29	20.90	7.02	15.18	16.41
Ours-Class-global (Real + syn.)	14.42	16.49	34.70	9.96	4.22	6.67	13.59	14.11	17.87	11.61	6.71	15.12	13.79

For most object categories, the category-specific model performs better than the global model. However, this improvement comes at a higher computational cost since twelve models must be trained to account for all object categories. We also compare the median error of the proposed regression models to the baseline model proposed by Mahendran et al. [56]. The results show that our proposed models perform better than the baseline model.

7.4 Comparison of the different methods

In this section we compare the performance of the methods used. In particular, we compare the performances of the classification and regression models. Moreover, the effect of synthetic data augmentation on the model performance is analyzed. Finally, we compare the performance of the category-specific models to the performance of the global model.

7.4.1 Classification and regression models

We compare the performance of the CNN-based classification and regression models. For both methods, the object viewpoints are represented as unit quaternions. We measure the viewpoint estimation error by calculating the angular distance between the predicted quaternion and the ground truth quaternion for all test images. We report the median errors for each object category. In general, the classification models perform better than the regression models, with the exception of a few object categories (dining table, motorbike, and sofa). Figure 7.6 shows a boxplot of the median errors of the classification model and the regression model for all object categories. Although the viewpoint space is continuous by nature, the classification models appear to perform better than regression models. The success of the classification method could be attributed to the CNN architecture used for the

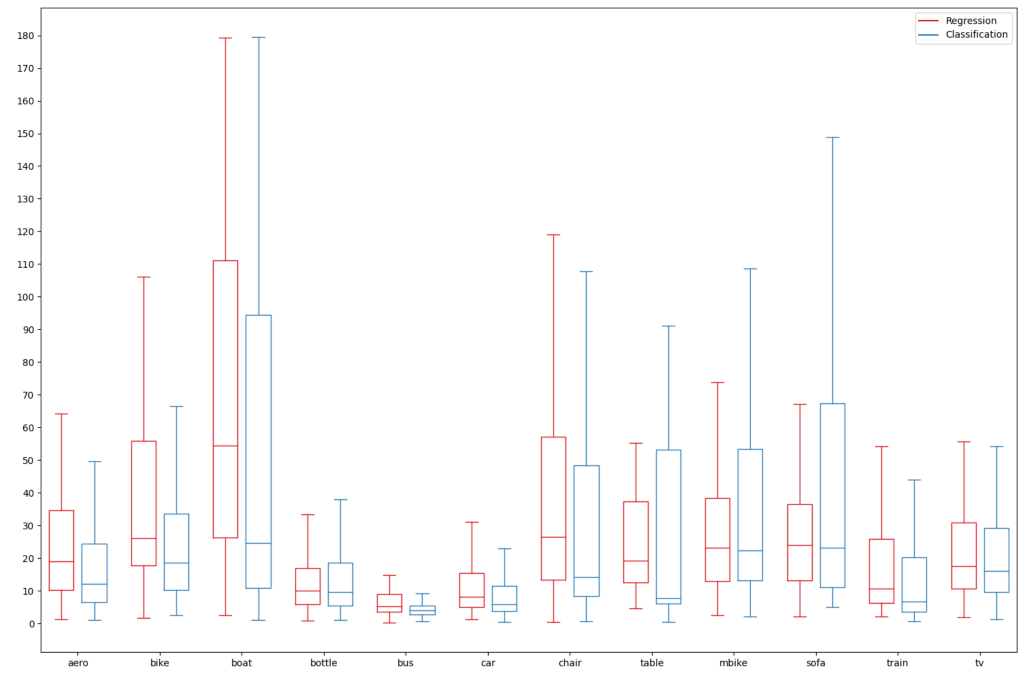


Figure 7.6: Median errors of the classification and regression models.

classification model, and the geometry-aware loss that accounts for the similarity between nearby viewpoints.

7.4.2 Real images and synthetic data augmentation

Here we investigate the influence of synthetic data augmentation on the viewpoint estimation performance. To this end, we compare the median errors of the classification model trained using the real images from PASCAL3D+ only and the classification model trained using the real images augmented with the synthetic images. Figure 7.7 displays the boxplots of the median errors for models trained using only real images, and for models trained using real images with synthetic data augmentation. The results show that the model trained using synthetic data augmentation outperforms the model trained on real images only, for most object categories. On average, the synthetic data augmentation reduces the median error by 3°. However, this data augmentation did not improve the performance of the boat and dining table object categories.

7.4.3 Category-specific and global models

We compare the performance of the category-specific models with the global model. For each object category, we compute the viewpoint error as the angular distance between the predicted viewpoint and the ground truth viewpoint. Figure 7.8 shows the boxplots of the median errors for the category-specific models and the global model.

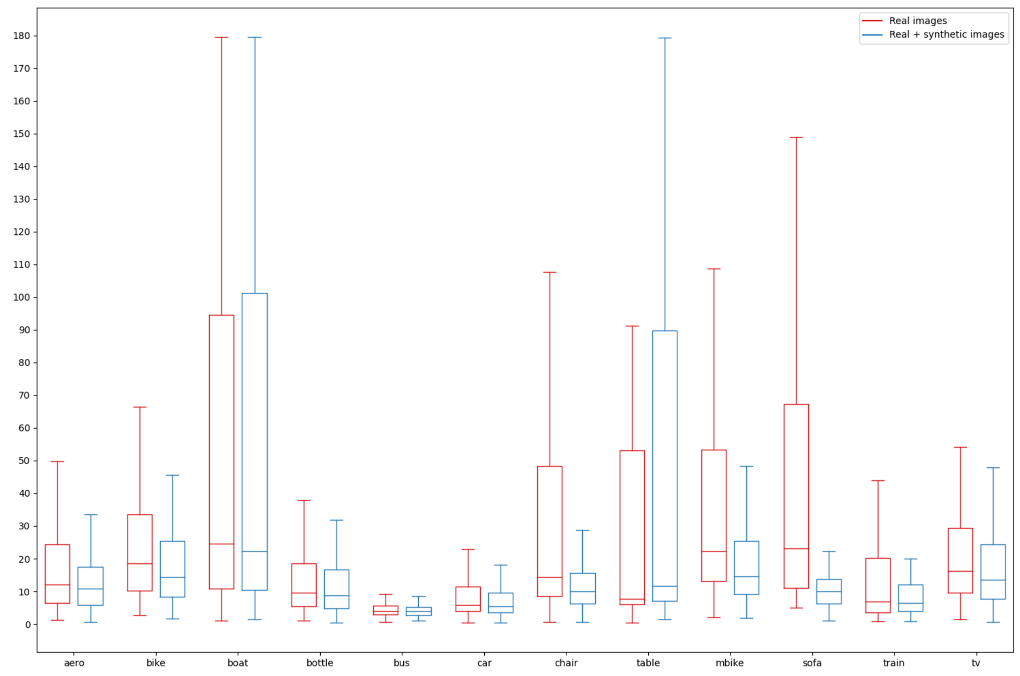


Figure 7.7: Median errors of the models trained on real images and models trained with synthetic data augmentation.

In general, the category-specific models perform better than the global model. However, the global model is more computationally efficient than the category-specific ones since the base model is shared between all the object categories. The good performance of the category-specific models could be attributed to the fact the viewpoints are more consistent within the same object category than across multiple categories. On the other hand, since the global model is trained on all twelve object categories, the viewpoint label might be ambiguous to the model.

7.4.4 Comparison with a state-of-the-art method

In this section we compare the results obtained using the category-specific models and the global model with the 3D regression method proposed by Mahendran et al. [56]. Using the median error as an evaluation metric, we compute the performance of each of the three methods. The models in all three methods are trained on the real images of PASCAL3D+ augmented with synthetic data. An illustration of the results obtained is displayed in Figure 7.9. The results show that our CNN-based classification models using quaternion representation outperform the baseline method for most object categories. In particular, the category-specific models give the best performance. These results confirm the effectiveness of the CNN-based classification model using quaternion representation at solving viewpoint estimation for natural images.

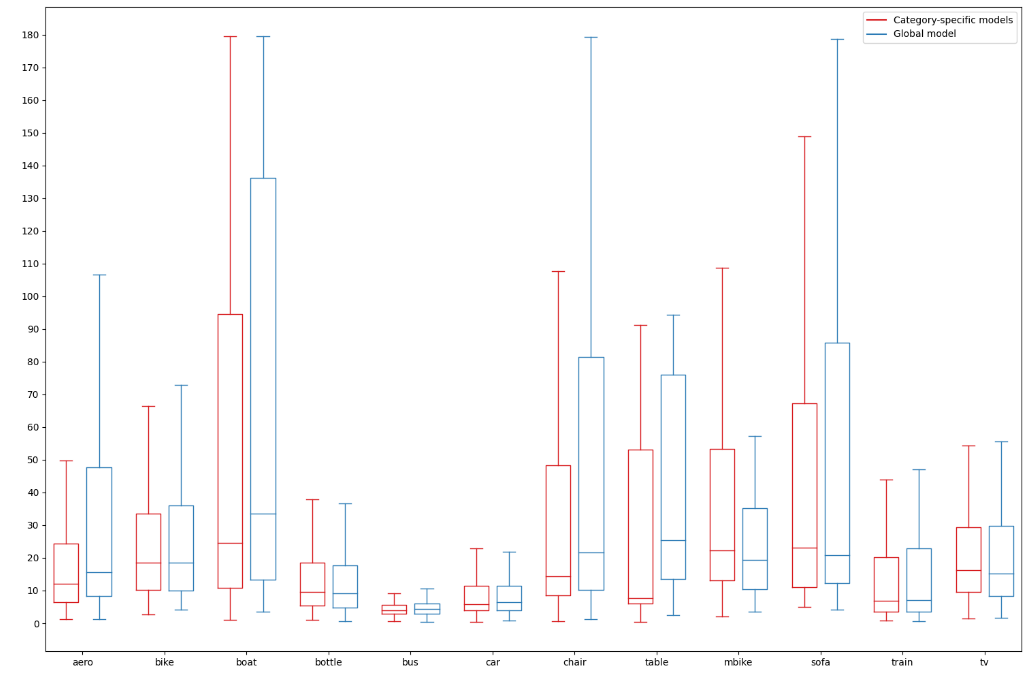


Figure 7.8: Median errors of the global models and the category-specific models.

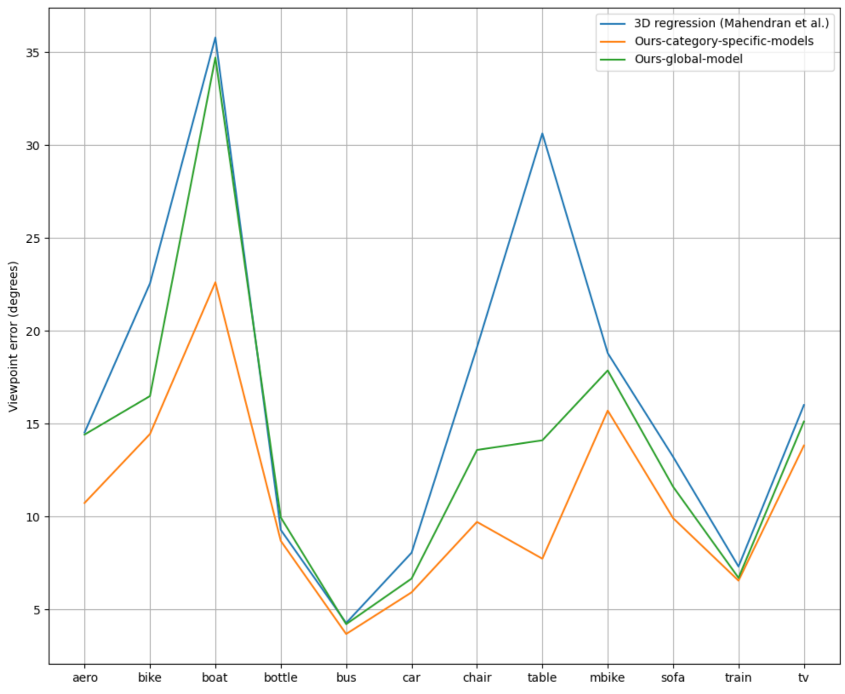


Figure 7.9: Viewpoint estimation errors for models trained on real PASCAL3D+ images and synthetic image augmentation.

7.5 Summary

In this chapter we extended the methods proposed for viewpoint estimation in medical imaging to natural images. Using the PASCAL3D+ dataset, we were able to train CNN-based regression and classification models to predict the viewpoint of twelve object categories. The results show that classification-based models using a unit quaternion representation are effective at estimating the viewpoints in natural images.

Chapter 8

Conclusion

The problem of viewpoint estimation in medical imaging using machine learning was investigated in this project. We particularly focused on developing convolutional neural network models to predict the orientation of a human body from a DRR. The problem was first simplified by considering the one-dimensional viewpoint estimation. We later extend the problem to the determination of full 3D orientation. In the first part of this chapter, we recapitulate the different methods proposed to solve the viewpoint estimation problem. In the second part of the chapter, we explore possible extensions of the project for future research.

8.1 Summary

Training a deep learning model for viewpoint estimation requires a large amount of data with accurate viewpoint labels. Thus, the first step during the investigation of this project was to generate such a dataset. In Chapter 3, we proposed methods based on voxel intensity projection to generate 2D images of digitally reconstructed radiographs (DRR) from a set of CT scan volumes. A vanilla method using the sum of the voxel intensities, called additive projection (AP), was used to generate orthogonal projections of the CT volumes onto a 2D plane for varying orientations of the CT volumes. However, this method proved to be very inefficient since it requires the rotation of the whole 3D volume before each projection. To overcome this, we used another projection technique based on the Fourier volume rendering. The efficiency of the Fourier volume rendering method is due to the fact that it only requires the extraction of 2D slices from the 3D volume for each projection orientation. This data generation strategy allowed us to gather DRRs with accurate viewpoint labels to train the viewpoint estimation models proposed in this project.

In Chapter 4, we discussed the one-dimensional case of the viewpoint estimation problem. We explored several types of models, which are either classification-based or regression-based. For the classification models, the viewpoint space was discretized into bins at 1-degree intervals. Each bin represents a viewpoint class. This model was trained using the usual cross-entropy loss with one-hot encoded label vectors. The CNN-based classification model was found to outperform the baseline model based on a k-nearest neighbor classifier. However, the one-hot encoding seems to ignore the geometric nature of the viewpoint estimation problem. Hence we later replace the one-hot encoding labels with soft label encoding. Moreover, a CNN-based regression model was used to investigate the effectiveness of regression-based methods. The results showed that the regression model does not perform as well as the CNN-based classification method.

The performance of the CNN-based classification method was improved using a “soft” version of the viewpoint labels in place of the one-hot encoding labels as presented in Chapter 5. The soft label encoding allows us to give non-zero weights to similar classes to the ground truth class to account for the similarity between viewpoint classes. This soft-encoding of the viewpoint classes significantly improves the performance of the CNN-based classification model. Furthermore, the performance of the classification model was improved using a geometric structure-aware loss in place of the usual cross-entropy loss. This cost function implicitly incorporates the label softening into the loss using the distance between viewpoint classes. Thus the geometric structure-aware cost function allows the trained model to capture the continuity of the viewpoint space.

In Chapter 6, we extended the viewpoint estimation problem to three-dimensional rotation angle estimation. We explored various representations for 3D rotations: Euler angles, axis angles, and quaternions. The quaternion representation turned out to be the most effective. We trained a CNN-based regression model to directly predict unit quaternions. This model proved to be more adapted to the viewpoint estimation problem than the regression of Euler angles. By sampling unit quaternions uniformly on the viewpoint space, we were able to train a CNN-based classification model to predict viewpoints using the quaternion representation. This classification model with unit quaternion representation performs better than the regression method.

We finally extended the viewpoint estimation problem to natural images in Chapter 7. We proposed CNN-based regression and classification methods for viewpoint estimation on the PASCAL3D+ dataset using quaternion representation. For the regression model, we convert the original Euler angle viewpoint labels to unit quaternions. We then trained a CNN-based regression model to predict the unit quaternion representing the viewpoint of each image. This regression model is trained by minimizing a cost function based on the distance between unit quaternions. As for the classification model, we first sampled unit quaternion uniformly distributed on the viewpoint space. The images in the dataset were assigned a viewpoint class based on their proximity with the sampled unit quaternions. The results show that the CNN-based classification and regression models outperform baseline work in the literature for viewpoint estimation on PASCAL3D+. Given that the PASCAL3D+ dataset includes images of twelve object categories, we used two CNN architectures for the viewpoint estimation models. In the first method, we trained a separate model for each of the twelve object categories. In the second method, we trained a global model that predicts the viewpoint regardless of the object category. The category-specific models perform better than the global model. However, they have a higher computational cost than the global model. The performance of the proposed methods for viewpoint estimation on PASCAL3D+ shows that the methods proposed in this thesis are not limited to medical imaging.

8.2 Recommendations and future work

We use Fourier volume rendering (FVR) for DRR generation in this work due to the computational efficiency of this method. FVR has a relatively low computation cost, which allows us to investigate online data generation and data augmentation during model training. This would not have been possible with slower rendering techniques. One limitation of FVR is that it uses parallel projection to generate DRRs. Thus FVR does not accurately model the process of real X-ray image generation, which uses perspective or fan-beam projection instead.

For applications that require the estimation of viewpoints from real X-ray images, we recommend the models proposed in this work be trained using DRR generated with rendering techniques such as DeepDRR, RealDRR, or other methods based on perspective or fan-beam projection when computation resources are available. Since the models presented in this work are not dependent on the data generation technique used, we expect similar viewpoint estimation results with other DRR generation methods such as DeepDRR or RealDRR.

Moreover we suggest a few research avenues that could be explored to extend the results presented in this thesis:

- **Extension to other body parts and other image modalities:** The viewpoint estimation methods proposed in this project focus on chest image views from DRRs generated from CT scans. This could be extended to include other body parts with DRR generated from a different image modality such as an MRI scan.

- **Application to content-based image retrieval:** The viewpoint estimation results could be used to develop a method for content-based image retrieval. This can be applied to a database of medical images to help radiologists retrieve similar cases for quicker and better diagnosis.

- **Application to 2D/3D image registration:** The viewpoint estimation methods proposed in this work output the estimated orientation of a human body from a projected DRR. This viewpoint estimate can be used as an initialization for a 2D/3D registration process, which would be useful to fuse information from a pre-operative image and an intra-operative image during image-guided surgery.

Bibliography

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. *TensorFlow: A system for Large-scale Machine Learning*. 2016. arXiv: 1605.08695 [cs.DC].

[2] Hidetaka Arimura, Shigehiko Katsuragawa, Qiang Li, Takayuki Ishida, and Kunio Doi. "Development of a computerized method for identifying the posteroanterior and lateral views of chest radiographs by use of a template matching technique". In: *Medical physics* 29.7 (2002), pp. 1556–1561.

[3] Bastian Bier, Mathias Unberath, Jan-Nico Zaech, Javad Fotouhi, Mehran Armand, Greg Osgood, Nassir Navab, and Andreas Maier. "X-ray-transform invariant anatomical landmark detection for pelvic trauma surgery". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2018, pp. 55–63.

[4] John M Boone, Greg S Hurlock, J Anthony Seibert, and Richard L Kennedy. "Automated recognition of lateral from PA chest radiographs: saving seconds in a PACS environment". In: *Journal of Digital Imaging* 16.4 (2003), pp. 345–349.

[5] John M Boone, Sadananda Seshagiri, and Robert M Steiner. "Recognition of chest radiograph orientation for picture archiving and communications systems display using neural networks". In: *Journal of Digital Imaging* 5.3 (1992), pp. 190–193.

[6] Anna Bosch, Andrew Zisserman, and Xavier Munoz. "Representing shape with a spatial pyramid kernel". In: *Proceedings of the 6th ACM International Conference on Image and Video Retrieval*. 2007, pp. 401–408.

[7] Sofien Bouaziz and Mark Pauly. “Dynamic 2D/3D Registration for the Kinect”. In: *ACM SIGGRAPH 2013 Courses*. SIGGRAPH ’13. Anaheim, California: Association for Computing Machinery, 2013. ISBN: 9781450323390. DOI: 10.1145/2504435.2504456. URL: https://doi.org/10.1145/2504435.2504456.

[8] Michael M Bronstein, Alexander M Bronstein, Fabrice Michel, and Nikos Paragios. “Data Fusion Through Cross-Modality Metric Learning Using Similarity-Sensitive Hashing”. In: *2010 IEEE computer society conference on computer vision and pattern recognition*. IEEE. 2010, pp. 3594–3601.

[9] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. “ShapeNet: An Information-Rich 3D Model Repository”. In: *CoRR* abs/1512.03012 (2015). arXiv: 1512.03012. URL: http://arxiv.org/abs/1512.03012.

[10] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. “Deeplab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.4 (2017), pp. 834–848.

[11] Xi Cheng, Li Zhang, and Yefeng Zheng. “Deep Similarity Learning for Multimodal Medical Images”. In: *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization* 6.3 (2018), pp. 248–252.

[12] Chen-Rui Chou, C Brandon Frederick, Sha X Chang, and Stephen M Pizer. “A learning-based patient repositioning method from limited-angle projections”. In: *Brain, Body and Machine: Proceedings of an International Symposium on the Occasion of the 25th Anniversary of the McGill University Centre for Intelligent Machines*. Springer. 2010, pp. 83–94.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2009, pp. 248–255.

[14] Markus Deserno. “How to generate equidistributed points on the surface of a sphere”. In: *If Polymerforshung (Ed.)* 99.2 (2004).

[15] Jennifer Dhont, Dirk Verellen, Isabelle Mollaert, Verdi Vanreusel, and Jef Vandemeulebroucke. “RealDRR–Rendering of realistic digitally reconstructed radiographs using locally trained image-to-image translation”. In: *Radiotherapy and Oncology* 153 (2020), pp. 213–219.

[16] Gilad Divon and Ayellet Tal. "Viewpoint Estimation—Insights & Model". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 252–268.

[17] Shane Dunne, Sandy Napel, and Brian Rutt. "Fast Reprojection of Volume Data". In: *[1990] Proceedings of the First Conference on Visualization in Biomedical Computing*. IEEE Computer Society. 1990, pp. 11–12.

[18] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. "The Pascal visual object classes (VOC) challenge". In: *International Journal of Computer Vision* 88.2 (2010), pp. 303–338.

[19] Mark Everingham and John Winn. "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Development Kit". In: *Pattern Analysis, Statistical Modelling and Computational Learning, Tech. Rep* 8 (2011).

[20] Xiang Fang, Leah Harris, Wei Zhou, and Donglai Huo. "Generalized radiographic view identification with deep learning". In: *Journal of Digital Imaging* 34.1 (2021), pp. 66–74.

[21] Javad Fotouhi, Bernhard Fuerst, Alex Johnson, Sing Chun Lee, Russell Taylor, Greg Osgood, Nassir Navab, and Mehran Armand. "Pose-aware C-arm for automatic re-initialization of interventional 2D/3D image registration". In: *International Journal of Computer Assisted Radiology and Surgery* 12 (2017), pp. 1221–1230.

[22] Dongshan Fu and Gopinath Kuduvalli. "A fast, accurate, and automatic 2D–3D image registration for image-guided cranial radiosurgery". In: *Medical physics* 35.5 (2008), pp. 2180–2194.

[23] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 580–587.

[24] Vivek Gopalakrishnan and Polina Golland. "Fast auto-differentiable digitally reconstructed radiographs for solving inverse problems in intraoperative imaging". In: *Workshop on Clinical Image-Based Procedures*. Springer. 2022, pp. 1–11.

[25] Alexander Grabner, Peter M Roth, and Vincent Lepetit. "3D Pose Estimation and 3D Model Retrieval for Objects in the Wild". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3022–3031.

[26] Matthias Grimm, Javier Esteban, Mathias Unberath, and Nassir Navab. "Pose-dependent weights and domain randomization for fully automatic X-ray to CT registration". In: *IEEE Transactions on Medical Imaging* 40.9 (2021), pp. 2221–2232.

[27] Wenhao Gu, Cong Gao, Robert Grupp, Javad Fotouhi, and Mathias Unberath. "Extended capture range of rigid 2D/3D registration by estimating Riemannian pose gradients". In: *Machine Learning in Medical Imaging: 11th International Workshop, MLMI 2020, Held in Conjunction with MICCAI 2020, Lima, Peru, October 4, 2020, Proceedings 11*. Springer. 2020, pp. 281–291.

[28] William Rowan Hamilton. "On a new system of imaginaries in algebra". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 33.219 (1848), pp. 58–60.

[29] JiaXin Han, TianYu Fu, JingFan Fan, QiaoLing Deng, and Jian Yang. "2D/3D US-tO-MRI Rigid Registration by Deep Learning". In: *2021 3rd International Conference on Intelligent Medicine and Image Processing*. 2021, pp. 33–38.

[30] Grant Haskins, Jochen Kruecker, Uwe Kruger, Sheng Xu, Peter A Pinto, Brad J Wood, and Pingkun Yan. "Learning Deep Similarity Metric for 3D MR-TRUS Registration". In: *arXiv preprint arXiv:1806.04548* (2018).

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778.

[32] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural networks* 2.5 (1989), pp. 359–366.

[33] Srikrishna Jaganathan, Maximilian Kukla, Jian Wang, Karthik Shetty, and Andreas Maier. "Self-Supervised 2D/3D Registration for X-Ray to CT Image Fusion". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2023, pp. 2788–2798.

[34] Srikrishna Jaganathan, Jian Wang, Anja Borsdorf, and Andreas Maier. "Learning the update operator for 2D/3D image registration". In: *Bildverarbeitung für die Medizin 2021: Proceedings, German Workshop on Medical Image Computing, Regensburg, March 7-9, 2021*. Springer. 2021, pp. 117–122.

[35] E-Fong Kao, Chungnan Lee, Twei-Shiun Jaw, Jui-Sheng Hsu, and Gin-Chung Liu. "Projection profile analysis for identifying different views of chest radiographs". In: *Academic Radiology* 13.4 (2006), pp. 518–525.

[36] E-Fong Kao, Wei-Chen Lin, Jui-Sheng Hsu, Ming-Chung Chou, Twei-Shiun Jaw, and Gin-Chung Liu. "A computerized method for automated identification of erect posteroanterior and supine anteroposterior chest radiographs". In: *Physics in Medicine & Biology* 56.24 (2011), p. 7737.

[37] Yueying Kao, Weiming Li, Zairan Wang, Dongqing Zou, Ran He, Qiang Wang, Minsu Ahn, and Sunghoon Hong. "An Appearance-and-Structure Fusion Network for Object Viewpoint Estimation". In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18.* International Joint Conferences on Artificial Intelligence Organization, July 2018, pp. 4929–4935. DOI: 10 . 24963 / ijcai . 20 @ inproceedings18 / 684. URL: https://doi.org/10.24963/ijcai.2018/684.

[38] Yueying Kao, Weiming Li, Zairan Wang, Dongqing Zou, Ran He, Qiang Wang, Minsu Ahn, and Sunghoon Hong. "An Appearance-and-Structure Fusion Network for Object Viewpoint Estimation." In: *IJCAI.* 2018, pp. 4929–4935.

[39] Jack Kiefer and Jacob Wolfowitz. "Stochastic estimation of the maximum of a regression function". In: *The Annals of Mathematical Statistics* (1952), pp. 462–466.

[40] Michael Kistler, Serena Bonaretti, Marcel Pfahrer, Roman Niklaus, and Philippe Büchler. "The virtual skeleton database: an open access repository for biomedical research and collaboration". In: *Journal of Medical Internet Research* 15.11 (2013), e2930.

[41] Julian Krebs, Tommaso Mansi, Hervé Delingette, Li Zhang, Florin C Ghesu, Shun Miao, Andreas K Maier, Nicholas Ayache, Rui Liao, and Ali Kamen. "Robust Non-rigid Registration Through Agent-Based Action Learning". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention.* Springer. 2017, pp. 344–352.

[42] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in Neural Information Processing Systems* 25 (2012).

[43] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[44] Daewon Lee, Matthias Hofmann, Florian Steinke, Yasemin Altun, Nathan D Cahill, and Bernhard Scholkopf. "Learning Similarity Measure for Multi-modal 3D Image Registration". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition.* IEEE. 2009, pp. 186–193.

[45] Thomas Martin Lehmann, Mark Oliver Gueld, Daniel Keysers, Henning Schubert, Andrea Wenning, and Berthold B Wein. “Automatic detection of the view position of chest radiographs”. In: *Medical imaging 2003: image processing.* Vol. 5032. SPIE. 2003, pp. 1275–1282.

[46] Thomas Martin Lehmann, Henning Schubert, Daniel Keysers, Michael Kohnen, and Berthold B Wein. “The IRMA code for unique classification of medical images”. In: *Medical Imaging 2003: PACS and Integrated Medical Information Systems: Design and Evaluation.* Vol. 5033. SPIE. 2003, pp. 440–451.

[47] Marc Levoy and Pat Hanrahan. “Light field rendering”. In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques.* 1996, pp. 31–42.

[48] Rui Liao, Shun Miao, Pierre de Tournemire, Sasa Grbic, Ali Kamen, Tommaso Mansi, and Dorin Comaniciu. “An Artificial Agent for Robust Image Registration”. In: *Thirty-First AAAI Conference on Artificial Intelligence.* 2017.

[49] Shuai Liao, Efstratios Gavves, and Cees GM Snoek. “Spherical Regression: Learning Viewpoints, Surface Normals and 3D Rotations on n-Spheres”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2019, pp. 9759–9767.

[50] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. “Feature Pyramid Networks for Object Detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2017, pp. 2117–2125.

[51] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully Convolutional Networks for Semantic Segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2015, pp. 3431–3440.

[52] David G Lowe. “Distinctive image features from scale-invariant keypoints”. In: *International Journal of Computer Vision* 60.2 (2004), pp. 91–110.

[53] Hui Luo, Wei Hao, David H Foos, and Craig W Cornelius. “Automatic image hanging protocol for chest radiographs in PACS”. In: *IEEE Transactions on Information Technology in Biomedicine* 10.2 (2006), pp. 302–311.

[54] Kai Ma, Jiangping Wang, Vivek Singh, Birgi Tamersoy, Yao-Jen Chang, Andreas Wimmer, and Terrence Chen. “Multimodal Image Registration with Deep Context Reinforcement Learning”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention.* Springer. 2017, pp. 240–248.

[55] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. “Rectifier nonlinearities improve neural network acoustic models”. In: *Proc. ICML*. Vol. 30. 1. Atlanta, GA. 2013, p. 3.

[56] Siddharth Mahendran, Haider Ali, and René Vidal. “3D Pose Regression using Convolutional Neural Networks”. In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2017, pp. 2174–2182.

[57] Tom Malzbender. “Fourier Volume Rendering”. In: *ACM Transactions on Graphics (TOG)* 12.3 (1993), pp. 233–250.

[58] George Marsaglia. “Choosing a point from the surface of a sphere”. In: *The Annals of Mathematical Statistics* 43.2 (1972), pp. 645–646.

[59] Francisco Massa, Renaud Marlet, and Mathieu Aubry. “Crafting a Multi-task CNN for Viewpoint Estimation”. In: *arXiv preprint arXiv:1609.03894* (2016).

[60] Shun Miao, Sebastien Piat, Peter Fischer, Ahmet Tuysuzoglu, Philip Mewes, Tommaso Mansi, and Rui Liao. “Dilated FCN for multi-agent 2D/3D medical image registration”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 2018.

[61] Shun Miao, Z Jane Wang, and Rui Liao. “A convolutional neural network approach for 2D/3D medical image registration”. In: *CoRR abs/1507.07505* (2015).

[62] Fabrice Michel, Michael Bronstein, Alex Bronstein, and Nikos Paragios. “Boosted Metric Learning for 3D Multi-modal Deformable Registration”. In: *2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*. IEEE. 2011, pp. 1209–1214.

[63] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. “V-net: Fully convolutional neural networks for volumetric medical image segmentation”. In: *2016 fourth international conference on 3D vision (3DV)*. IEEE. 2016, pp. 565–571.

[64] Vinod Nair and Geoffrey E Hinton. “Rectified linear units improve restricted Boltzmann machines”. In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. 2010, pp. 807–814.

[65] J Anthony Parker, Robert V Kenyon, and Donald E Troxel. “Comparison of interpolating methods for image resampling”. In: *IEEE Transactions on Medical Imaging* 2.1 (1983), pp. 31–39.

[66] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019.

[67] Sébastien Piat, Shun Miao, Rui Liao, Tommaso Mansi, and Jiannan Zheng. *Dilated Fully Convolutional Network for Multi-Agent 2D/3D Medical Image Registration*. US Patent App. 16/103,196. Feb. 2019.

[68] Mark R Pickering, Abdullah A Muhit, Jennie M Scarvell, and Paul N Smith. "A new multi-modal similarity measure for fast gradient-based 2d-3d image registration". In: *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE. 2009, pp. 5821–5824.

[69] Ewa Pietka and HK Huang. "Orientation correction for chest images". In: *Journal of Digital Imaging* 5.3 (1992), pp. 185–189.

[70] Michael JD Powell. "The BOBYQA algorithm for bound constrained optimization without derivatives". In: *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge* 26 (2009).

[71] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. Curran Associates, Inc., 2015, pp. 91–99. URL: http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf.

[72] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.

[73] Daniel B Russakoff, Torsten Rohlfing, Kensaku Mori, Daniel Rueckert, Anthony Ho, John R Adler, and Calvin R Maurer. "Fast generation of digitally reconstructed radiographs using attenuation fields with application to 2D-3D image registration". In: *IEEE Transactions on Medical Imaging* 24.11 (2005), pp. 1441–1454.

[74] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. *ImageNet Large Scale Visual Recognition Challenge*. 2015. arXiv: 1409.0575 [cs.CV].

[75] Seyed Sadegh Mohseni Salehi, Shadab Khan, Deniz Erdogmus, and Ali Gholipour. "Real-time deep pose estimation with geodesic loss for image-to-template rigid registration". In: *IEEE Transactions on Medical Imaging* 38.2 (2018), pp. 470–481.

[76] Robert L Siddon. "Fast calculation of the exact radiological path for a three-dimensional CT array". In: *Medical physics* 12.2 (1985), pp. 252–255.

[77] Martin Simonovsky, Benjamin Gutierrez-Becker, Diana Mateus, Nassir Navab, and Nikos Komodakis. "A deep metric for multimodal registration". In: *Medical Image Computing and Computer-Assisted Intervention-MICCAI 2016: 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part III 19*. Springer. 2016, pp. 10–18.

[78] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-scale Image Recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[79] Hao Su, Charles R Qi, Yangyan Li, and Leonidas J Guibas. "Render for CNN: Viewpoint Estimation in Images using CNNs Trained with Rendered 3D Model Views". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2686–2694.

[80] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1–9.

[81] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2818–2826.

[82] Engin Tola, Vincent Lepetit, and Pascal Fua. "Daisy: An efficient dense descriptor applied to wide-baseline stereo". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.5 (2009), pp. 815–830.

[83] Takashi Totsuka and Marc Levoy. "Frequency Domain Volume Rendering". In: *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. 1993, pp. 271–278.

[84] Shubham Tulsiani and Jitendra Malik. “Viewpoints and Keypoints”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2015, pp. 1510–1519.

[85] Mathias Unberath, Jan-Nico Zaech, Cong Gao, Bastian Bier, Florian Goldmann, Sing Chun Lee, Javad Fotouhi, Russell Taylor, Mehran Armand, and Nassir Navab. “Enabling Machine Learning in X-ray-based Procedures via Realistic Simulation of Image Formation”. In: *International Journal of Computer-Assisted Radiology and Surgery (IJCARS)* (2019).

[86] Mathias Unberath, Jan-Nico Zaech, Sing Chun Lee, Bastian Bier, Javad Fotouhi, Mehran Armand, and Nassir Navab. “DeepDRR–a catalyst for machine learning in fluoroscopy-guided procedures”. In: *Medical Image Computing and Computer Assisted Intervention–MICCAI 2018: 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part IV 11.* Springer. 2018, pp. 98–106.

[87] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. “Convolutional pose machines”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2016, pp. 4724–4732.

[88] Wolfgang Wein, Barbara Röper, and Nassir Navab. “2D/3D registration based on volume gradients”. In: *Medical Imaging 2005: Image Processing.* Vol. 5747. SPIE. 2005, pp. 144–150.

[89] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. “Beyond PASCAL: A benchmark for 3D object detection in the wild”. In: *IEEE Winter Conference on Applications of Computer Vision.* IEEE. 2014, pp. 75–82.

[90] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. “SUN Database: Large-scale Scene Recognition from Abbey to Zoo”. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* IEEE. 2010, pp. 3485–3492.

[91] Zhiyun Xue, Daekeun You, Sema Candemir, Stefan Jaeger, Sameer Antani, L Rodney Long, and George R Thoma. “Chest x-ray image view classification”. In: *2015 IEEE 28th International Symposium on Computer-Based Medical Systems.* IEEE. 2015, pp. 66–71.

[92] Jiana Yang, Shilin Wang, and Gongshen Liu. “Viewpoint Estimation in Images by a Key-Point Based Deep Neural Network”. In: *2019 IEEE International Conference on Image Processing (ICIP).* IEEE. 2019, pp. 2551–2555.

[93] Sergey Zagoruyko and Nikos Komodakis. “Learning to Compare Image Patches via Convolutional Neural Networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2015, pp. 4353–4361.