Automatic Vertebrae Localization, Identification, and Segmentation Using Deep Learning and Statistical Models

by

Amin Suzani

B.Sc. Computer Engineering, Sharif University of Technology, 2012

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Applied Science

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL
STUDIES
(Electrical and Computer Engineering)

The University Of British Columbia
(Vancouver)

October 2014

Abstract

Automatic localization and identification of vertebrae in medical images of the spine are core requirements for building computer-aided systems for spine diagnosis. Automated algorithms for segmentation of vertebral structures can also benefit these systems for diagnosis of a range of spine pathologies. The fundamental challenges associated with the above-stated tasks arise from the repetitive nature of vertebral structures, restrictions in field of view, presence of spine pathologies or surgical implants, and poor contrast of the target structures in some imaging modalities.

This thesis presents an automatic method for localization, identification, and segmentation of vertebrae in volumetric computed tomography (CT) scans and magnetic resonance (MR) images of the spine. The method makes no assumptions about which section of the vertebral column is visible in the image. An efficient deep learning approach is used to predict the location of each vertebra based on its contextual information in the image. Then, a statistical multi-vertebrae model is initialized by the localized vertebrae from the previous step. An iterative expectation maximization technique is used to register the statistical multi-vertebrae model to the edge points of the image in order to achieve a fast and reliable segmentation of vertebral bodies. State-of-the-art results are obtained for vertebrae localization in a public dataset of 224 arbitrary-field-of-view CT scans of pathological cases. Promising results are also obtained from quantitative evaluation of the automated segmentation method on volumetric MR images of the spine.

Preface

This thesis is primarily based on several manuscripts, resulting from the collaboration between multiple researchers. All publications have been modified to make the thesis coherent. Ethical approval for conducting this research has been provided by the UBC Research Ethics Board, certificate numbers: H13-02361.

A version of Chapter 2 has been published in:

- Amin Suzani, Abtin Rasoulian, Sidney Fells, Robert N. Rohling, and Purang Abolmaesumi. Semi-automatic segmentation of vertebral bodies in volumetric MR images using a statistical shape+pose model. In *SPIE Medical Imaging*, pages 90360P–90360P. International Society for Optics and Photonics, 2014.

The contribution of the author was in developing, implementing, and evaluating the presented framework. The statistical model of the vertebral bodies along with the implementation of the model registration was provided by Dr. Rasoulian. Profs. Abolmaesumi, Rohling, and Fels helped with their valuable suggestions for improving the methodology. All co-authors contributed to the editing of the manuscript.

Table of Contents

List of Tables

List of Figures

Glossary

CT	Computed Tomography
MR	Magnetic Resonance
PACS	Picture Archiving and Communication System
SVM	Support Vector Machine
SVD	Singular Value Decomposition
PCA	Principal Component Analysis
HMM	Hidden Markov Model
RF	Regression Forest
GPU	Graphics Processing Units
SGD	Stochastic Gradient Descent
RELU	Rectified Linear Unit
HF	Hessian-free
KDE	Kernel Density Estimation
STD	Standard Deviation
CF	Classification Forest
CUDA	Compute Unified Device Architecture

CPU	Central Processing Units
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
ROI	Region Of Interest

Acknowledgments

First and foremost, I would like to thank my supervisor, Purang Abolmaesumi, for guiding me throughout this journey. He has always been a wonderful example for me both in research and professional behaviour.

Special thanks to Robert Rohling, Abtin Rasoulian, and Weiqi Wang for their insightful feedback and sharing their knowledge and experience in the fields of medical image analysis and GPU programming.

I also would like to thank all my colleagues and friends in the Robotics and Controls Lab for providing a very friendly and pleasant working environment during my two-year stay.

Finally, I would not be where I am today without the unconditional love and support from my parents, Tahereh and Hasan.

Chapter 1

Introduction

1.1 Clinical Background

The human spine (also referred to as vertebral column) normally consists of 33 vertebrae. The upper 24 vertebrae are articulating and separated from each other by intervertebral discs. The lower nine are fused in the sacrum and the coccyx. The articulating vertebrae are grouped into three regions: 1) seven *cervical* vertebrae of the neck, 2) twelve *thoracic* vertebrae of the middle back, and 3) five *lumbar* vertebrae of the lower back. The order of the regions and also the numbering of each region are from top to bottom. The number of vertebrae in each region are slightly variable in the population. Figure 1.1 illustrates the regions of the vertebral column and their conventional numbering notation [71].

The cervical region is located in the neck. It has generally smaller vertebrae compared to other regions. The shape of the first two cervical vertebrae (called the atlas and the axis) differs from the rest. These two vertebrae are used for rotating the neck. The thoracic vertebrae, on the other hand, are connected to the ribs and have limited movement. This section forms the steady mid back of the human body. The lumbar vertebrae are generally larger than the vertebrae of other regions. This section bears most of the weight and also movements (bending, twisting, etc.) of the body. Lower back pain, one of the most common types of pain, usually occurs in the lumbar region [62].

A typical vertebra includes a vertebral body in the front and a vertebral arch

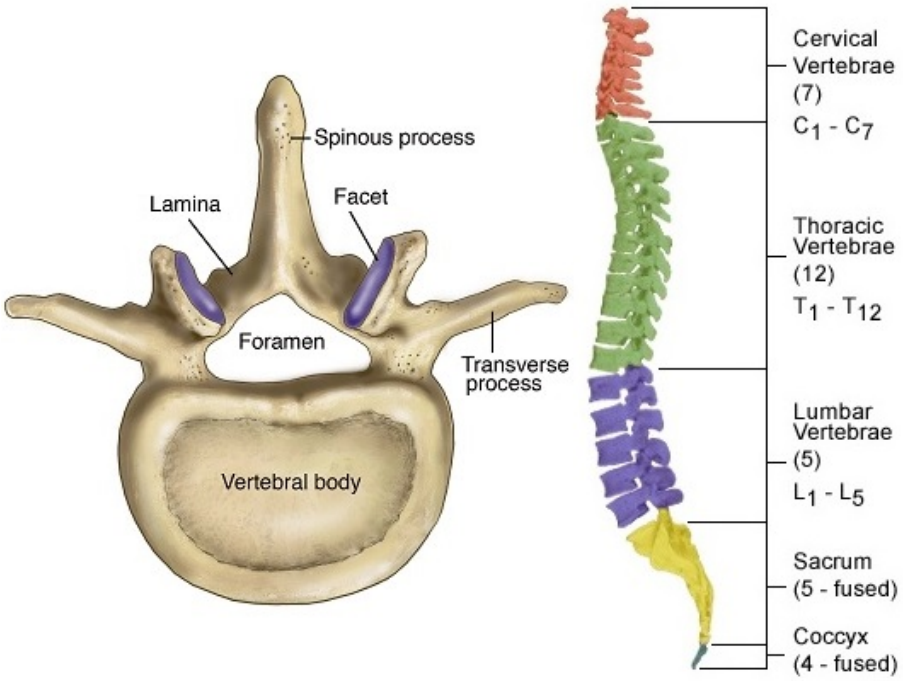

Figure 1.1: Left: Main parts of a typical vertebra, Right: The regions of the human vertebral column. (From http://www.compelvisuals.com, http://www.studyblue.com)

in the back which encloses the vertebral foramen. One spinous process, two transverse processes, and a pair of laminae are some remarkable parts of the vertebral arch. Figure 1.1 shows the main parts of a typical vertebra.

The spine is commonly viewed by X-rays, Computed Tomography (CT), and Magnetic Resonance (MR) imaging. X-rays and CT are more affordable and generally provide better contrast of bony structures. However, their capability in viewing soft tissues is very limited. On the other hand, MR depicts much better contrast of soft tissues like nerve roots and intervertebral discs [54]. In addition, unlike CT and X-rays, MR imaging does not expose the patient to ionizing radiations [12, 55].

1.2 Thesis Objectives

The global objective of this thesis is to facilitate building of computer-aided systems for spinal disease diagnosis, surgical planning, and post-operative assessments. To this end, we investigate and develop techniques for automatic vertebrae detection, localization, identification, and segmentation within a clinically acceptable time-frame.

Localization and identification of the vertebrae and discs are essential steps in the analysis of spine images. Radiologists report the diagnosis after detecting and identifying the vertebrae present in the image [1]. In particular, lower back pain, one of the most common types of pain, can be caused by a wide range of spine diseases and pathologies (herniated disc, degenerative disc, and spondylolisthesis to name a few). Reliable computer-aided diagnosis systems can massively help the physicians to find the cause of lower back pain. One of the core requirements for building such systems is developing a reliable method for automated localization and identification of vertebrae. Once the target vertebrae are localized, the system will be able to perform any subsequent diagnosis on them.

Segmentation of vertebral structures in medical images of the spine can also benefit computer aided systems for measuring the deformations caused by different spinal pathologies and diseases. These measurements can be used for diagnosis, pre-operative planning and also post-operative assessments. For instance, scoliosis is a spine pathology in which the patient's spine is curved from side to side. This pathology can be diagnosed from the shapes and orientations of the vertebrae. Therefore, a reliable segmentation of vertebral structures are necessary. In addition, automatic spine segmentation as well as vertebrae localization can benefit a Picture Archiving and Communication System (PACS) by allowing the system to store more meaningful data than the raw images.

1.3 Thesis Structure

The rest of this thesis is subdivided into five chapters as outlined below:

Chapter 2 describes a semi-automatic method for segmenting vertebral bodies in typical volumetric MR images of the spine, where the slice spacing is large. Using a graphical user interface, the user clicks on each vertebra that needs to be

segmented. The method takes advantage of the correlation between shape and pose of different vertebrae in the same patient by registering a statistical multi-vertebrae model to the image.

Chapter 3 presents an automatic method based on deep learning for simultaneous localization and identification of vertebrae in general CT scans, where the field-of-view is unknown and spine pathologies may be present.

Chapter 4 combines the last two chapters to propose a fully-automatic approach for localization, labeling, and segmentation of vertebral bodies in multi-slice MR images. The need for user interaction in Chapter 2 is replaced with a modified version of the automatic localization method presented in Chapter 3.

Chapter 5 describes two main contributions for speeding up the methods proposed in previous chapters by vectorization and GPU acceleration. A vectorization approach for the feature extraction algorithm and a GPU-accelerated method for the statistical model registration are proposed and compared against their sequential versions.

Chapter 6 concludes the thesis with a short summary followed by remarking the major contributions along with the direction of future work on the subject.

Chapter 2

Semi-automatic Segmentation in MRI

2.1 Introduction

Segmentation of vertebral structures in volumetric medical images of the spine is an essential step in the design of computer aided diagnosis systems for measuring the deformations caused by different spinal pathologies such as osteoporosis, spinal stenosis, spondylolisthesis and scoliosis. When making decisions for diagnosis and therapy of these pathologies, physicians often use MR images for the assessment of soft spinal structures such as inter-vertebral discs and nerve roots. However, the established methods for segmentation of vertebral structures have been mainly developed for CT. On the other hand, CT requires radiation exposure and does not depict soft tissue as well as MR images. The availability of a fast and reliable spine segmentation of MR images may eliminate the need for CT and benefit patient care.

Segmentation of vertebral structures in MR images is challenging, mainly due to poor contrast between bone surfaces and surrounding soft tissue. Another challenge is relatively large inter-slice gap (more than 3 *mm*) in typical clinical MR images compared to CT, which is normally sub-millimeter in recent generation of scanners. In addition, spatial variations in surrounding soft tissue contrast and magnetic field inhomogeneities increase the complexity of the segmentation task in MR images.

For spine segmentation in MR images, several methods have been proposed in the literature. Most of them are 2D approaches that are applied on manually identified cross-sections which contain the target vertebrae [8, 12, 23, 60]. Peng et al. [48] search for the best slice automatically and then perform a 2D segmentation approach. A smaller number of methods have been proposed for MR segmentation of vertebral structures in 3D. The method from Hoad et al. [22] has a labour-intensive initialization step and also a post-processing step for correcting the segmentation. Štern et al. [64, 65] perform the segmentation by optimizing the parameters of a geometrical model of vertebral bodies. The method from Neubert et al. [41] uses statistical shape models and registration of grey level intensity profiles for vertebral body segmentation. Zukic et al. [78] use multiple-feature boundary classification and mesh inflation. Methods from Hoad et al. and Štern et al. have long execution times (at least more than five minutes for high-resolution volumetric images). All above-stated 3D approaches segment each vertebra independently. Consequently, the correlation between shapes of different vertebrae in a patient is disregarded. In addition, the methods from Hoad et al., Štern et al., and Neubert et al. are evaluated on volumetric images with an inter-slice gap of at most 1.2 *mm* which is not commonly used in clinical practice. Recently, a method was proposed by Kadoury et al. [28] for spine segmentation using manifold embeddings. Multiple vertebrae are treated as a whole shape in this work. However, it is still evaluated on MR images with less than 1.2 *mm* slice spacing, and no computation time was reported.

In a previous work of our research group [50], we have successfully segmented lumbar vertebrae in volumetric CT scans using a statistical multi-vertebrae anatomical shape+pose model. In this thesis, we aim to find simple and fast pre-processing steps to overcome the MR segmentation challenges explained above, and consequently obtain edge points of vertebral bodies from widely spaced slices of routine volumetric MR images. Thereafter, we register our multi-vertebrae anatomical model to the extracted edge points with the goal of achieving a fast and reliable segmentation of lumbar vertebral bodies in MR images.

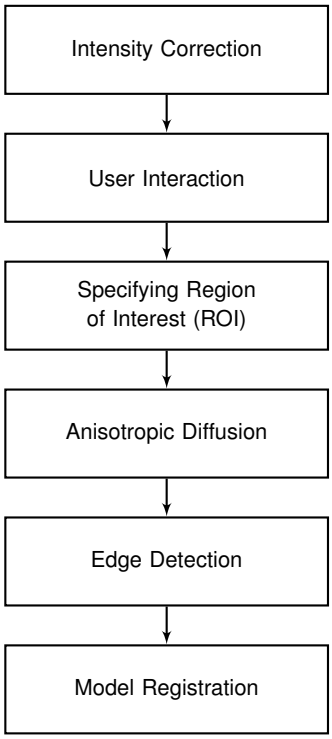

Figure 2.1: Flowchart of the semi-automatic framework for segmentation of vertebral bodies.

2.2 Materials

The performance of the proposed method was evaluated on T1-weighted MR images of nine patients (via SpineWeb [72]). The in-plane resolution is 0.5×0.5 mm^2 with a slice spacing between 3.3 to 4.4 mm. Each series of images consists of slices with 512×512 pixels. The number of slices from each patient ranges from 12 to 18. The multi-vertebrae anatomical model was constructed from manually segmented volumetric CT scans of an independent group of 32 subjects [50, 52]. Ground truth segmentations were obtained manually from raw images using ITK-SNAP [74].

2.3 Methods

2.3.1 Intensity Correction

The presence of intensity inhomogeneity in spinal MR images can adversely influence the ability to extract edges. Hence, we first apply an intensity correction algorithm on MR images to reduce this inhomogeneity. The bias field correction function provided in Statistical Parametric Mapping software package (SPM12b, Wellcome Department of Cognitive Neurology, London, UK) is used for this purpose. Figures 2.2(a) and 2.2(b), respectively, show the mid-sagittal slice of a sample MR image before and after intensity correction.

2.3.2 Anisotropic Diffusion

We then apply a conventional 3D anisotropic diffusion [49] filter to the intensity-corrected image. Anisotropic diffusion is an image smoothing algorithm that attempts to preserve edges. Figures 2.2(b) and 2.2(c), respectively, show the image before and after this step. The function below (proposed by Perona and Malik [49]) is used for the diffusion coefficient in order to privilege high-contrast edges over low-contrast ones.

$$c(x,y,z,t) = e^{-(||\nabla I||/\kappa)^2}, \tag{2.1}$$

∇I denotes the image gradient, and the conductance parameter, κ, controls the sensitivity to edges. Weak edges can block the conductance when κ is too low, and strong edges may not be preserved when κ is too high. In all experiments κ and the integration constant, Δt, were set to 50 and $1/7$, respectively. The variables x, y, and z represent the spatial coordinates, while the variable t enumerates iteration steps. The algorithm was run with 20 iterations for each volumetric image.

2.3.3 Canny Edge Detection

At this step, the intensity-corrected mid-sagittal slice of the image is shown to the user. The user clicks on each of the lumbar vertebrae (L1 to L5) in the image (five points in total). For some irregular image acquisitions where those vertebrae are not visible in the mid-sagittal slice, the user could select a slice manually to

be able to click on the vertebrae. However, we do not observe any sensitivity to the slice selection. Only the clicked points are used for the rest of the procedure. After user interaction, a 2D Canny edge detection algorithm [7] is applied to each sagittal slice to find the edge map of vertebral bodies in potential regions of the volumetric image. These regions are five cubic bounding boxes centered at the clicked points. The scales of the cubes are obtained from the average distance of consecutive user clicks. The sensitivity threshold of the Canny edge detector is obtained automatically based on the maximum value of the gradient magnitude of the region. Figure 2.3 shows the result of edge detection in mid-sagittal slice of three different volumetric MR images. The statistical model is registered to the edge map obtained from this step.

2.3.4 Segmentation Using the Multi-vertebrae Model

Variations of shapes and poses of lumbar vertebrae were independently extracted from previously acquired 32 manually-segmented volumetric CT images [50]. This analysis was performed on all five lumbar vertebrae combined, taking into account the correlation between shapes of different vertebrae. The extracted variations are then represented in a statistical model. An iterative expectation maximization registration technique presented by Rasoulian et al. [50] is used for aligning the model to the edge points extracted from MR images in the previous step. The structure of the model and the registration technique are described in more detail in [50].

2.4 Results and Discussions

Our segmentation results are quantitatively evaluated by using the manual segmentation as the ground truth. The mean error and Hausdorff distance are reported for each vertebra in a volumetric image (Table 2.1). For each point of the 3D surface mesh of manual segmentation we find its distance to the closest point in the registered model. The average of these distances is considered as the mean error and the maximum of them is considered as the Hausdorff distance. In order to make our results comparable with other approaches that only focus on 2D segmentation [8, 12, 23, 60], we also report the segmentation errors for the 2D mid-sagittal slice of each image in Table 2.2. Some examples of our segmentation results are

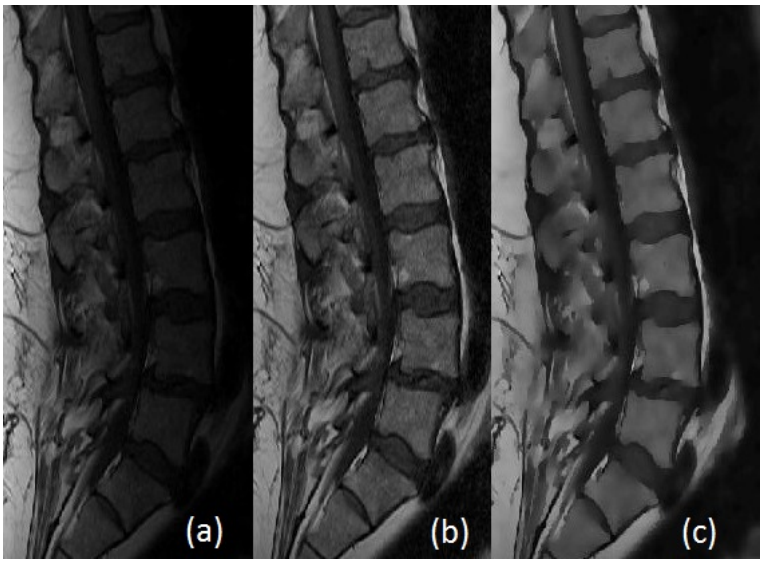

Figure 2.2: (a) Mid-sagittal slice of original image, (b) after intensity-correction, (c) after intensity correction and anisotropic diffusion.

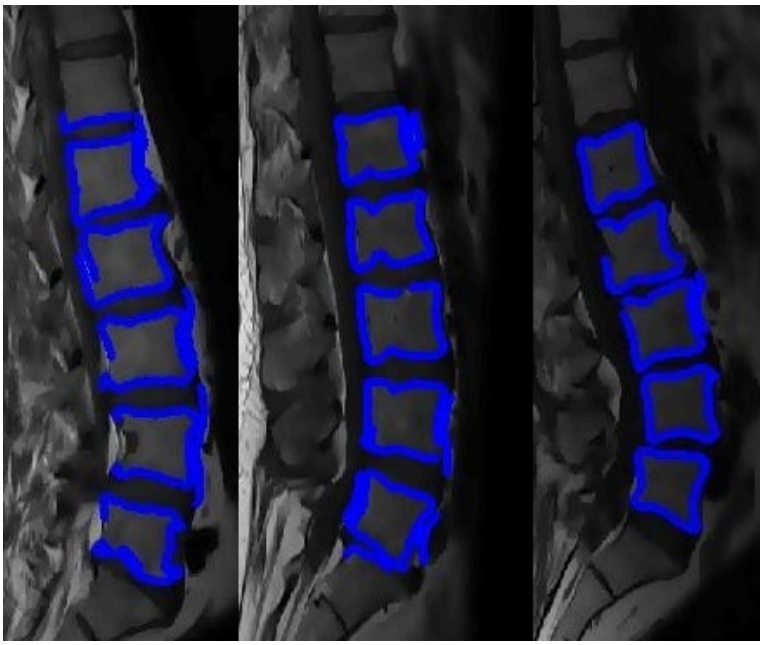

Figure 2.3: Edge extraction results in the mid-sagittal slice of three different volumes.

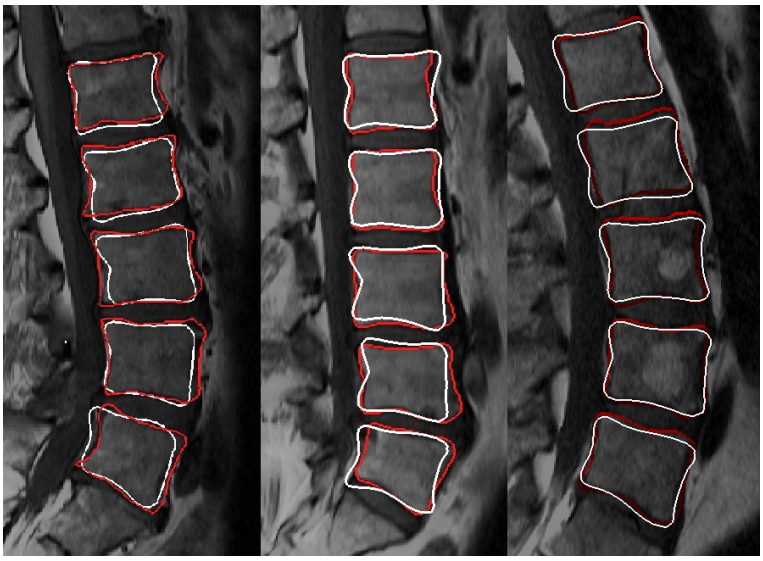

Figure 2.4: Examples of segmentation results in three different subjects. White contours show our segmentation results, while manual segmentation is shown with red contours.

shown in Figure 2.4. Although the slice spacing in our MR images is relatively high (between 3.3 and 4.4 *mm*), the quantitative results show that our method can segment the lumbar vertebral bodies in MR images with a mean error of ~ 3 *mm* and standard deviation of ~ 0.8 *mm*. The overall segmentation for each image takes less than 2 minutes using our implementation in MATLAB (The MathWorks, Inc., Natick, MA) on a 2.5 GHz Intel core i5 machine.

Table 2.1: Mean error and maximum distance (Hausdorff) of 3D segmentation in each multi-slice image are reported (in mm) for each vertebral body.

Subject	L1		L2		L3		L4		L5		All lumbar vertebrae	
	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean $\pm$ std	Max
1	2.6	6.5	2.2	6.7	2.1	5.9	2.2	7.7	2.6	8.3	2.3 ± 0.2	8.3
2	2.9	9.8	2.8	8.1	2.4	7.3	3.1	11.3	2.9	9.1	2.8 ± 0.3	11.3
3	2.8	7.1	2.1	5.7	2.0	6.5	2.8	6.0	3.0	8.5	2.5 ± 0.4	8.5
4	2.7	7.5	3.7	9.6	3.8	9.6	4.0	10.3	3.7	12.4	3.6 ± 0.5	12.4
5	2.6	7.4	2.4	8.5	2.5	8.8	2.6	9.3	3.3	11.1	2.7 ± 0.4	11.1
6	1.9	6.4	2.3	7.0	2.9	10.2	3.2	12.3	3.0	12.1	2.7 ± 0.5	12.3
7	3.6	11.6	4.1	12.0	3.7	11.1	4.4	11.4	5.1	13.2	4.2 ± 0.6	13.2
8	2.5	6.7	3.3	11.7	5.9	16.3	4.0	12.8	2.6	7.2	3.7 ± 1.4	16.3
9	2.0	6.3	2.3	6.6	3.0	9.5	3.3	11.7	2.7	8.9	2.7 ± 0.5	11.7
Average	2.6	7.7	2.8	8.4	3.1	9.4	3.3	10.3	3.2	10.1	3.0 ± 0.8	11.7

Table 2.2: Mean error and maximum distance (Hausdorff) of 2D segmentation in the mid-sagittal slice of each image are reported (in mm) for each vertebral body.

Subject	L1		L2		L3		L4		L5		All lumbar vertebrae	
	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean $\pm$ std	Max
1	1.6	3.7	1.6	4.2	1.4	2.9	1.6	2.8	1.3	2.5	1.5 ± 0.1	4.2
2	1.6	3.6	2.1	4.3	1.7	3.5	2.3	4.2	2.0	4.1	2.0 ± 0.3	4.3
3	1.4	2.5	1.4	2.4	1.6	2.6	2.1	4.4	3.0	5.8	1.9 ± 0.7	5.8
4	1.7	4.4	1.8	4.3	1.7	4.6	2.1	4.8	2.3	5.3	1.9 ± 0.3	5.3
5	2.2	5.4	2.4	5.3	2.3	7.2	2.7	9.5	2.8	7.1	2.5 ± 0.3	9.5
6	1.6	4.0	1.5	2.6	1.5	3.6	1.8	5.0	1.7	3.6	1.6 ± 0.1	5.0
7	2.1	4.1	1.7	4.6	1.6	3.6	1.9	4.7	2.5	7.1	2.0 ± 0.3	7.1
8	1.9	4.9	2.3	5.7	1.9	5.2	2.8	7.4	2.2	5.1	2.2 ± 0.4	7.4
9	1.9	4.5	1.5	2.7	1.5	3.1	2.5	5.5	1.5	2.7	1.8 ± 0.4	5.5
Average	1.8	4.1	1.8	4.0	1.7	4.0	2.2	5.4	2.2	4.8	1.9 ± 0.4	6.0

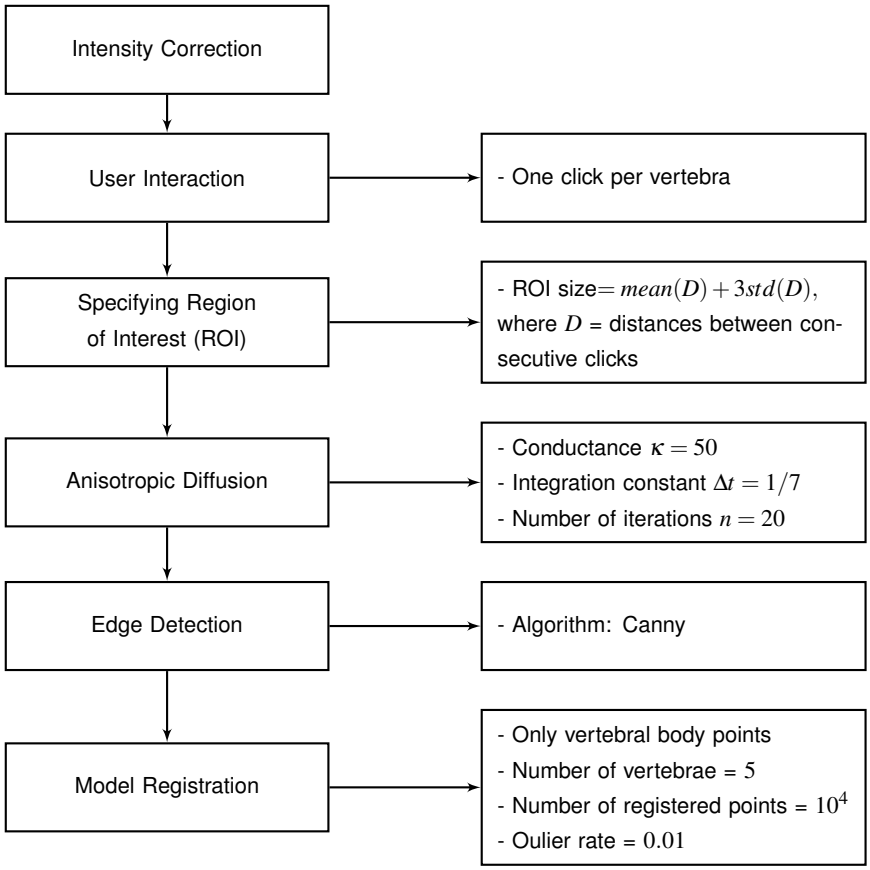

Figure 2.5: Summery of the parameters of the semi-automatic framework for segmentation of vertebral bodies. The values of the main parameters of each step are shown in the boxes at the rightmost column.

2.5 Summary

A semi-automatic segmentation algorithm based on registering a statistical model is applied on nine 3D MR images of the spine. Adding simple MR pre-processing steps allows the multi-vertebrae shape+pose model developed previously for CT scans to work on volumetric MR images. The statistical model can accommodate large inter-slice gaps (about 4 *mm*). In addition, it is fast because it exploits the fact that neighbouring vertebrae have similar shapes and poses. The segmentation error will determine the range of clinical applications of this method.

Chapter 3

Deep Learning for Automatic Vertebrae Localization in CT

3.1 Introduction

Automatic vertebra localization and identification (sometimes referred to as labeling) in spinal imaging is a crucial component for image-guided diagnosis, surgical planning, and follow-up assessment of spine disorders such as disc/vertebra degeneration, vertebral fractures, scoliosis, and spinal stenosis.

The main challenges associated with building an automated system for robust localization and identification of vertebrae in spine images arise from: 1) restrictions in field of view; 2) repetitive nature of spinal column; 3) high inter-subject variability in spine curvature and shape due to spine disorders and pathologies; and 4) image artifacts caused by metal implants.

Several methods have been proposed in the literature for automatic vertebra localization and labeling in CT [15, 16, 20, 32, 36, 39, 53], MR [1, 44, 48, 58], and X-ray [63, 75] images. Most of the previous works either concentrate on specific region or make an assumption about the visible part of vertebral column in the image. A few studies claimed handling arbitrary-field-of-view scans in a fully-automatic system [15, 16, 32, 53]. The methods proposed in [15] and [53] rely on a generative model of shape and/or appearance of vertebrae. This may cause these methods to struggle with pathological subjects, especially when metal im-

plants produce artifacts in the image. The computational cost for handling general scans is high in [32] and [53]. The recently proposed method in [16] has led to state-of-the-art results in the dataset we are using in this work. The key idea in that work is that the points of the image are probabilistically classified as being a particular vertebra centroid. Based on predicted probabilities, the centroid of these points are obtained using mean-shift. Although promising results are obtained on a challenging dataset, their method requires an additional post-processing step for removing false positives. This step adds up to the computation time and may not be robust when the spine images are not tightly cropped. In addition, their approach for centroid estimation approach based on mean-shift requires certain parameters and thresholds which may require manual tuning. Most importantly, the fastest of the above-stated methods [16] have a computation time of about a minute. This computation time limits the application of these methods to be used as preprocessing steps for wide range of image-guided tasks in clinical basis.

In this work, we aim to find a faster and more robust solution to the problem of vertebra localization in general clinical CT scans by using deep neural networks [3, 57]. No assumptions are made about which and how many vertebrae are visible in images. Our method is extensively evaluated on a public dataset, which consists of 224 three-dimensional CT scans, and is compared to recently proposed state-of-the-art approaches.

3.2 Materials

The performance of our method is evaluated on a publicly-available large dataset consists of 224 spine-focused three-dimensional CT scans of patients with varying types of pathologies. The pathologies include fractures, high-grade scoliosis, and kyphosis. Many cases are post-operative scans where severe image artifacts are caused by surgical implants. Different sections of spine are visible in different images. The field-of-view is mostly limited to 5-15 vertebrae while the whole spine is visible in a few cases. It is the same dataset used in [16]. Figure 3.1 illustrates several cases of this challenging dataset.

Statistical multi-vertebrae shape+pose model mentioned in Section 3.5 was constructed from manually segmented three-dimensional segmented CT images of

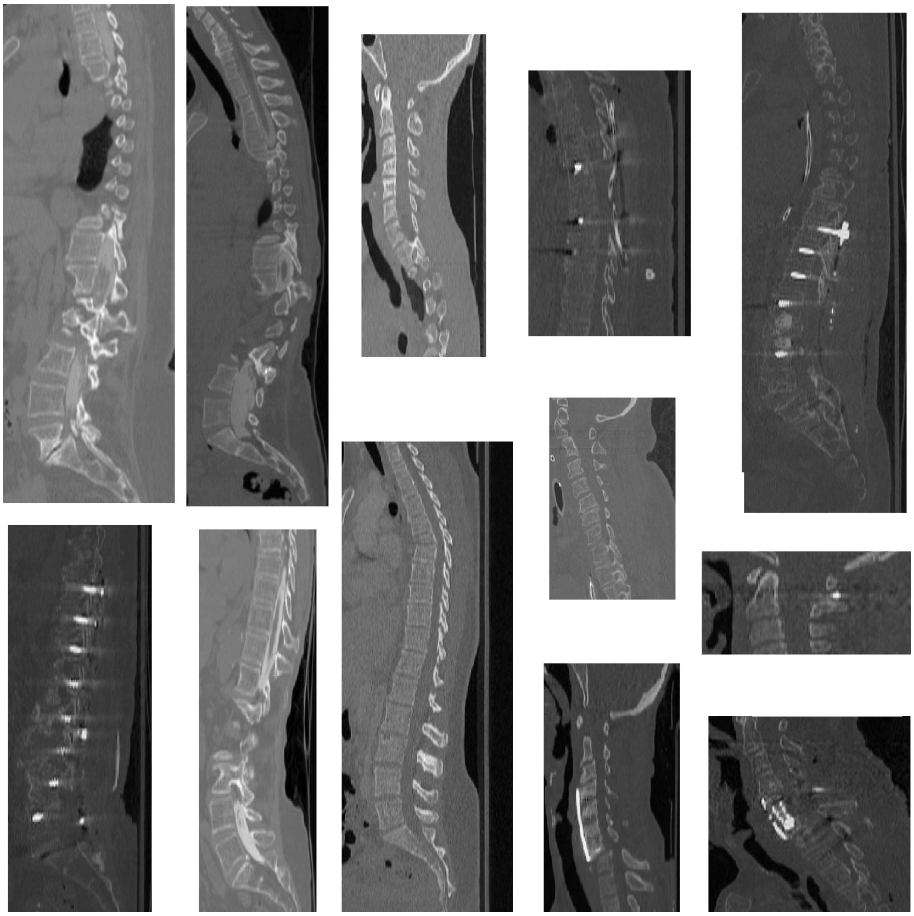

Figure 3.1: The mid-sagittal slice of several images from the dataset. The dataset consists of 224 three-dimensional CT scans of patients with varying types of pathologies. In many cases, severe image artifacts are caused by metal implants. Different regions of the spine are visible in different images.

an independent group of 32 subjects which were mostly non-pathological [50]. The manual segmentation is done using ITK-SNAP [74].

3.3 Simultaneous Localization and Labeling

3.3.1 Intensity-based Features

Hundreds of intensity-based features are extracted from each voxel of the volumetric image. The value of each feature is the mean intensity over a three-dimensional cuboid displaced with respect to the reference voxel position. The cuboid dimensions and the displacement of each feature are chosen randomly. For the reference voxel p, the feature vector $v(p) = (v_1, ..., v_j, ..., v_n)$ is computed as follows:

$$v_j = \frac{1}{|F_{p;j}|} \sum_{q \in F_{p;j}} I(q), \tag{3.1}$$

where $I(q)$ is the image intensity at position q in the image, and $q \in F_{p;j}$ are the image voxels within the cuboid. $F_{p;j}$ denotes the feature cuboid i displaced in respect to pixel p. Figure 3.2 shows a visualization of our intensity-based features. Similar types of featuers are used in [9, 10, 14, 61].

Extracting mean intensity over cuboidal regions can be computed very quickly by using the idea of the integral image (introduced in [69]). The integral image is an intermediate representation for the original image. By definition

$$ii(x,y,z) = \sum_{x' \leq x, y' \leq y, z' \leq z} i(x', y', z'), \tag{3.2}$$

where $i(x,y,z)$ is the original image and $ii(x,y,z)$ is the integral image. A 2D representation of the integral image is shown in Figure 3.3. Each pixel in this integral image contains the sum of intensities above and to the left of itself, inclusive. For computing the sum of intensities in the box D, we only need to compute $II_4 - II_2 - II_3 + II_1$ which is only three operations (in 3D it will be seven operations). The integral image can be constructed recursively from the original image using a few operations per voxel. Once constructed, any of our cuboidal features (regardless of location and size) can be computed in constant time.

Extracting hundreds of above-mentioned features from an image voxel provides a meaningful description of the area around the reference voxel. This description is then used to train a learning system for vertebra localization.

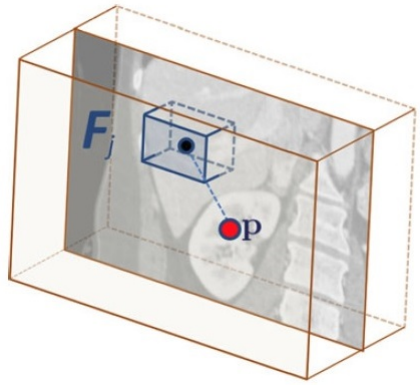

Figure 3.2: The value of each feature is the mean intensity over a three-dimensional cuboid displaced with respect to the reference voxel position. From [10].

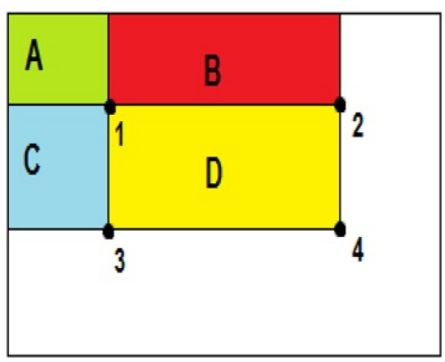

Figure 3.3: A 2D example of the integral image approach. Each pixel in the integral image contains the sum of intensities above and to the left of itself, inclusive. Therefore, sum of intensities in rectangle D of the original image can be computed quickly by two subtractions and one addition using the integral image: $II(4) - II(3) - II(2) + II(1) = \sum I(A \cup B \cup C \cup D) - \sum I(A \cup C) - \sum I(A \cup B) + \sum I(A) = \sum I(D)$.

Before training, we randomly generate the parameters of hundreds of random features. Then the same parameters are used in all steps of training and testing. Generating feature parameters involves randomly choosing the values of cuboid dimensions vector S and displacement vector D, where $0 < S_i \leq max\_size$ and $0 \leq D_i \leq max\_displace$ for $i \in \{1,2,3\}$. For the experiments of this chapter, we used $max\_size = 32\,mm$, $max\_displace = 12\,mm$ to extract 500 features from each voxel.

3.3.2 Parametrizing Localization Problem As a Regression

Vertebra localization problem is parametrized as a multi-variate non-linear regression. As explained above, hundreds of the cuboidal intensity-based features are extracted from each voxel. The targets of the regression are the relative distance between the center of each vertebral body and the reference voxel. In other words, the vector from the reference voxel to the center of the vertebral body is considered as one target in the regression.

The number of observations (samples) in our regression problem is equal to the number of voxels that are present in the image. The number of features are equal to 500 for the experiments of this chapter. Since the images of our CT dataset are labeled with 26 landmarks (26 vertebral bodies), the target vector of our regression includes 26 three-dimensional vectors per observation. Therefore, the vertebra localization problem is parametrized as multi-variate regression with 500 features and $26 \times 3 = 78$ targets.

For the vertebrae that are not present in an image in training set, we make a rough guess of the location of them outside of the field of view of the image. This rough guess is made by rigid registration of simple model to the last three present landmarks in the image. The model is build by averaging the position of the same vertebrae from all images of the dataset when they are rigidly registered together. Figure 3.5 shows a rough guess of the location of the non-visible cervical vertebral bodies in an image in training set.

3.3.3 Deep Neural Networks for Regression

For decades, shallow neural networks (with zero or one hidden layers) have been used for machine learning. In 1990s, introduction of newer machine learning mod-

Figure 3.4: The vertebrae localization problem is parametrized as a regression problem. The targets of the regression are the vectors from the reference voxel to the center of each vertebral body (The orange arrow). The reference voxel is described by 500 intensity-based features from the area around itself (Shown in red).

els such as Support Vector Machine (SVM) and Random Forests caused loss of popularity for neural networks among researchers. However, Since 2006 some techniques have been developed that enables effective training of deep neural network (with multiple hidden layers). Recent research shows that these deep networks can outperform other learning models in several applications. During last few years, state-of-the-art results have been produced by applying deep learning to various tasks in computer vision [33, 67].

Network structure

In the experiments of this chapter, a deep feed-forward neural network with six layers is used for solving the multi-variate non-linear regression problem. The structure of our neural network is demonstrated in Figure 3.6. The intensity-based

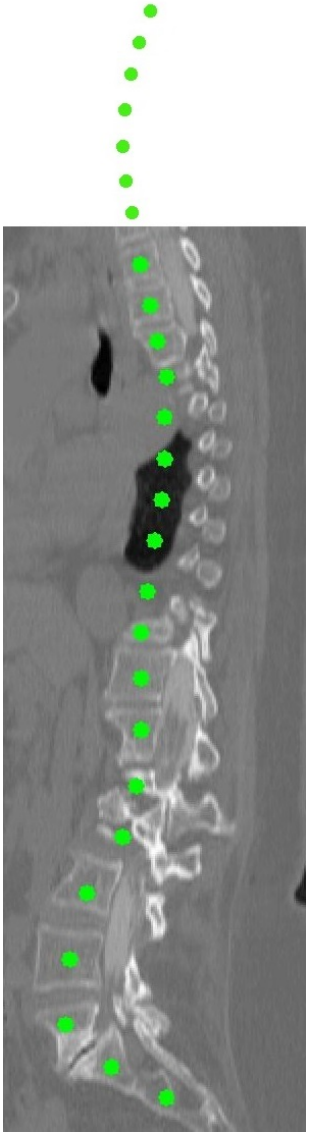

Figure 3.5: The landmarks that are used for training and also evaluation are the center of vertebral bodies. For the training set, we make a rough guess of the location of the vertebrae that are not visible in the image. Rigid registration of a simple model is used for this purpose.

features are given to the network via the first layer (input layer). Then, layers are activated consecutively from left to right. After activation of all units in the network, the outputs will be present in the last layer (output layer). Any layer between input layer and output layer is called a hidden layer. Training a neural network involves using training data to assign values to the connection weights between layers. Once the weights are determined, units of each layer can be activated for test data as follows:

$$a_i^{(l+1)} = g(\sum_{j=0}^{N_l} \Theta_{ij}^{(l)} a_j^l), \tag{3.3}$$

where $a_i^{(l)}$ denotes the activation of unit i in layer l, and $\Theta_{ij}^{(l)}$ is the weight of connection from unit j of layer l to unit i of layer $l+1$. $g(x)$ is the activation function. We have used Rectified Linear Unit, $g_{hidden}(x) = max(0,x)$, for hidden layers and linear function, $g_{output}(x) = x$, for output layer. A visualization of these functions is brought in Figures 3.7 and 3.8.

Cost function

For training the network, we define a cost function as a measure of how far away our solution is from the optimal solution. Our cost function is defined as:

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log h_\Theta(x^{(i)})_k + (1-y_k^{(i)}) \log(1-h_\Theta(x^{(i)})_k)\right], \tag{3.4}$$

where $h_\Theta(x^{(i)})_k$ is the hypothesis for the kth output using the input x from sample i of the training data. $y_k^{(i)}$ denotes the ground truth for kth output for the ith sample. K is the number of outputs (equal to 78 in experiments of this chapter). m is the number of training samples in the mini-batch that we are computing the cost function. In particular, the cost function is a measure that how far each hypothesis $h_\Theta(x^{(i)})$ is from its corresponding ground truth $y_k^{(i)}$. The above-stated function is preferred over possible trivial cost functions like $-\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K}(h_\Theta(x^{(i)})_k - y_k^{(i)})^2\right]$, because it is known to be more convex and consequently less prone to falling into local minima during the optimization process. However, the cost function is still not guaranteed to be a pure convex function. Figure 3.9 illustrates a sample of cost function with respect of two parameters (two connection weights of the network).

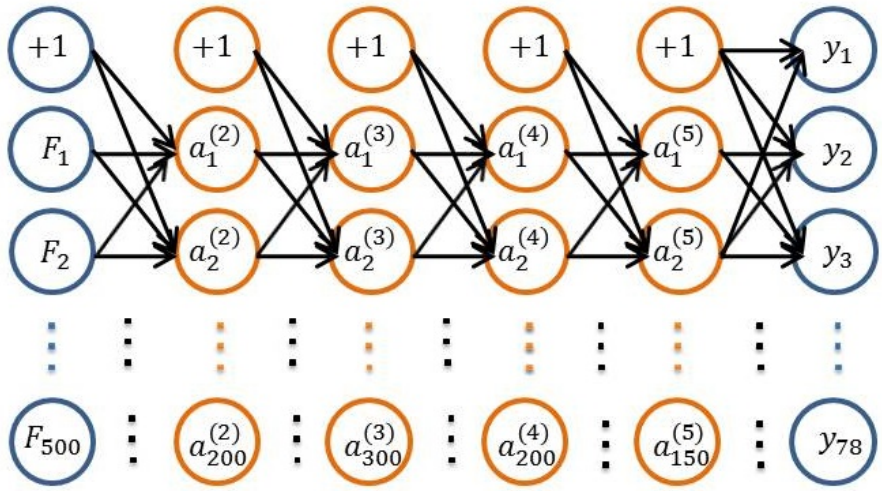

Figure 3.6: The deep neural networks consists of six layers that are trained using layerwise pre-training and stochastic gradient descent. Hidden units are shown in orange while input and output layers are shown in blue.

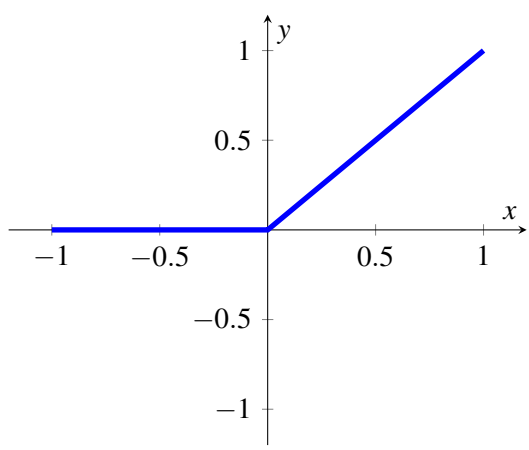

Figure 3.7: Rectifier functions are used as the activation functions for hidden units.

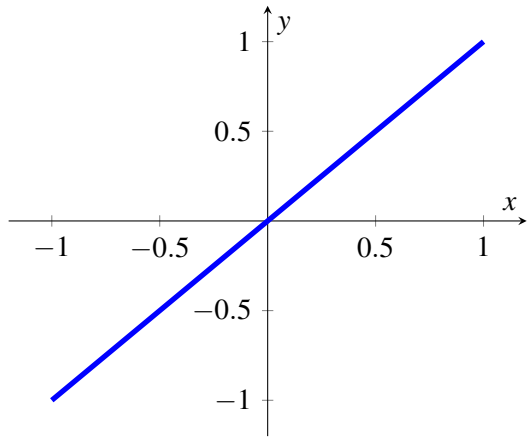

Figure 3.8: Linear functions are used for the output layer activation.

A typical approach for training neural networks is to iteratively use back-propagation algorithm [34, 70] to compute partial derivatives of cost function with respect to each connection weight of all layers, $\frac{\delta}{\delta\Theta_{ij}^{(l)}}J(\Theta)$, and then use Stochastic Gradient Descent (SGD) to optimize the cost function by changing the connection weights. Nevertheless, this approach may lose effectiveness when we have a deep network with two or more hidden layers. The main reason is the significant propagation of error in computing partial derivatives using conventional back-propagation over the whole network.

Layerwise pre-training

In recent years, it has been shown that deep neural networks can be trained much more effectively using layerwise pre-training [4, 17]. This method involves breaking down the deep network into several shallow networks and training each of them greedily.

Assume that we have K hidden layers ($K = 4$, in the experiments of this chapter). First, we disregard all hidden layers except the first one. It means that we build a network including the input layer, first hidden layer, and the output layer. We train this shallow network using conventional back-propagation and stochastic

gradient descent. Once the weights of the first hidden layer are trained, we add the next layer. Therefore, second network is built using the input layer, first hidden layer, second hidden layer, and the output layer. We train the weights of the second hidden layer re-using the trained weights of the previous layer, and so forth. We continue this approach until all hidden layers 1 to $K-1$ are trained. The output layer is present in all steps because we need to compare it against the ground truth and compute the cost function. Once the hidden layers 1 to $K-1$ are trained, we use back-propagation and SGD over the whole network to determine the weights of the last hidden layer as well as fine-tuning the pre-trained weights of the previous layers.

Second neural network for the z coordinates of outputs

As mentioned earlier, the outputs of our network are the relative distance vector between the center of vertebral bodies and the the reference voxel. As we have 26 landmarks, 26 out of all $26 \times 3 = 78$ outputs are the z coordinates of these distance vectors. Due to asymmetric structure of the spine (tall and thin structure) the range of the Z outputs is much wider than the range of x and y outputs. The Z outputs vary in range $[-800, 800]$ *mm* which is the order of the spine height, whereas the x and y outputs vary in rage $[-200, 200]$ *mm* which is the order of maximum width of the CT images of spine. However, all these outputs are scaled linearly to the same range $[-1, 1]$. This means that the network will emphasize more on optimizing x and y outputs than the z outputs, while the z outputs are more important to us for vertebra identification (adjacent vertebrae have similar x and y coordinates, but different z coordinate). To alleviate this issue, we train a separate feed-forward neural network for only z outputs. The structure of this additional network is similar to the main network except that it has only 26 units in its output layer.

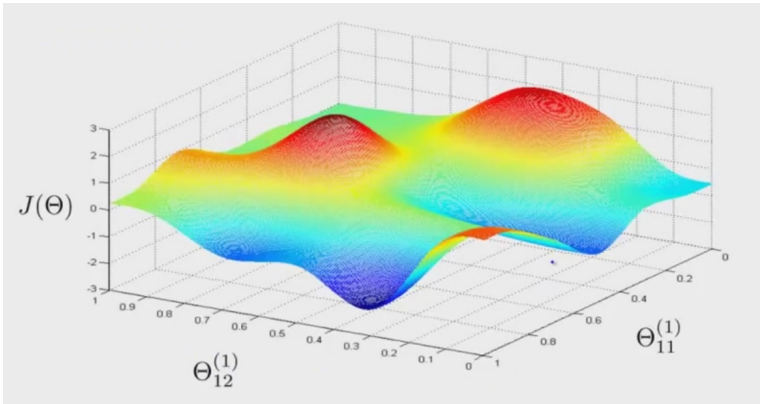

Figure 3.9: A sample visualization of the cost function with respect to two arbitrary parameters of the network. SGD aims to converge to the global minimum while the cost function is not guaranteed to be convex.

3.3.4 Kernel Density Estimation for Vote Aggregation

Procedure on a test image

On a 3D test image, we first extract hundreds of features from each voxel. Then, we use the deep neural network to predict the relative distance vector of each vertebral body with respect to the reference voxel. Knowing the location of the reference voxel and the relative distance vector to the center of a specific vertebral body, we can compute the predicted absolute location of the vertebral body on the test image. This absolute location is considered the vote of that specific voxel for the location of that specific vertebral body. Note that these votes might be either inside or outside of the scope of image. Each voxel in the image votes for the location of each vertebral body, so for each specific vertebral body (say L1) we end up with thousands of votes acquired from different voxels. We need to aggregate these votes to obtain a single prediction for the location of the vertebral body. Mean, median, or histogram mode of the votes can be used for aggregation, but a more robust and accurate method is using Kernel Density Estimation (KDE).

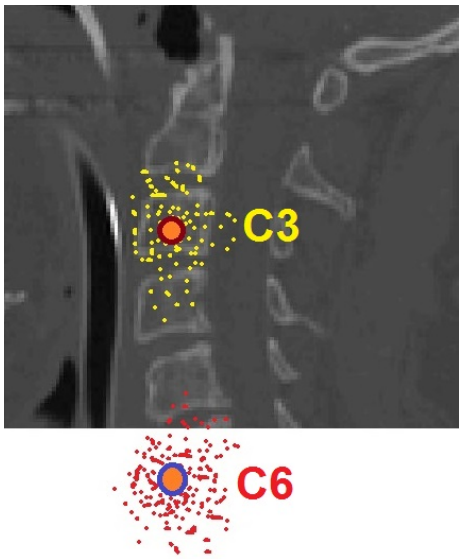

Figure 3.10: Thousands of voxels of the image vote for the location of each vertebra. The centroid of these votes are estimated using KDE. Based on the votes, a vertebra may be localized inside or outside of the field of view of the image.

Kernel density estimation

Kernel Density Estimation (KDE) is a non-parametric approach to estimate the probability density function of a random variable. The simplest, most familiar non-parametric density estimator is a histogram. However, there are several issues with the histograms to be used in certain applications: 1) Histograms are not smooth; 2) They depend on end points of bins; 3) They depend on the width of bins. On the other hand, kernel density estimators are smooth, and have no end points. Figure 3.11 shows an example of a Kernel density estimation versus a histogram.

Kernel density estimators still depend on a parameter called *bandwidth*. A density estimation is said to be undersmoothed when we choose a bandwidth that is too small. Conversely, an overly large value for the bandwidth will cause oversmoothing. An automatic data-driven bandwidth selection approach as well as an adaptive kernel density estimator based on linear diffusion processes is presented

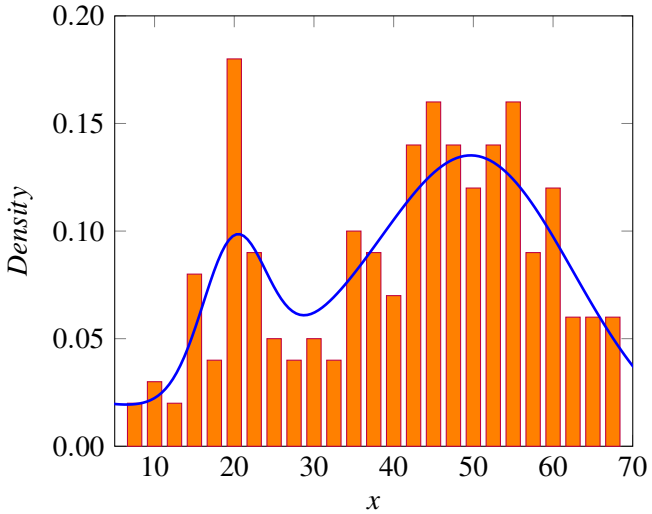

Figure 3.11: Kernel density estimation of a sample set of points. The global maximum of this estimation is used as the centroid of the votes. In multimodal distributions like this sample, the estimated centroid from KDE may significantly differ from mean, median, or mode of the distribution.

by Botev et al. in [6]. Their approach is less sensitive to outliers than the popular Gaussian kernel density estimator because it is locally adaptive. It also has shown better performance in terms of speed and reliability than most of other adaptive approaches [19, 26, 47, 68]. The advantage of their automatic data-driven bandwidth selection over the other most popular approaches [25, 59] is that it does not use reference rules [27] and consequently does not make assumptions about normality of the input data.

In our experiments, we used the kernel density estimation method presented by Botev et al. to obtain a fast and reliable density function for the voxel votes for each vertebral body. The global maximum of this density function is considered as the predicted location of the vertebral body on the image.

3.4 Hyper-Parameters Optimization

One of the main challenges of building a deep learning system is the presence of plenty of hyper-parameters which need to be adjusted. Since the training time for deep networks is usually high (several hours to several weeks), extensive evaluation on all parameters can be impractical. Consequently, the learning system may end up being a sub-optimal solution. In this section, we briefly discuss the effect of some of the most important parameters in each phase of our deep learning system in the context of vertebra localization and identification.

3.4.1 Parameters in Feature Extraction

Downsample rate

When we described the method in Section 3.3, we explained how we extract features from all voxels of the image. While it is possible to do this for all voxels, it would need a training time in the order of weeks, and testing time in the order of a few minutes. However, we desire to decrease the testing time of the deep network to less than a second. In our experiments, we extract these features from all over the 3D image with a downsample rate of 12 *mm* in each direction. Note that it is different from downsampling the image with this rate and then extracting features from all voxels. In the experiments of this chapter, we first downsample the images with the rate of 1 *mm* in each direction. Then, we extract high-quality features from the voxels that have the distance of 12 *mm* to each other in each direction. The number of these voxels is still over a thousand on average for each image. This approach brings down the training time from weeks to two days, and the testing time from a few minutes to less than a second. The feature extraction implementation is explained in more detail in Chapter 5. Figure 3.12 shows that smaller downsample rates do not improve the localization results. In other words, it shows that the number of voxels obtained from downsampling rate of 12 *mm* are sufficient for the localization framework.

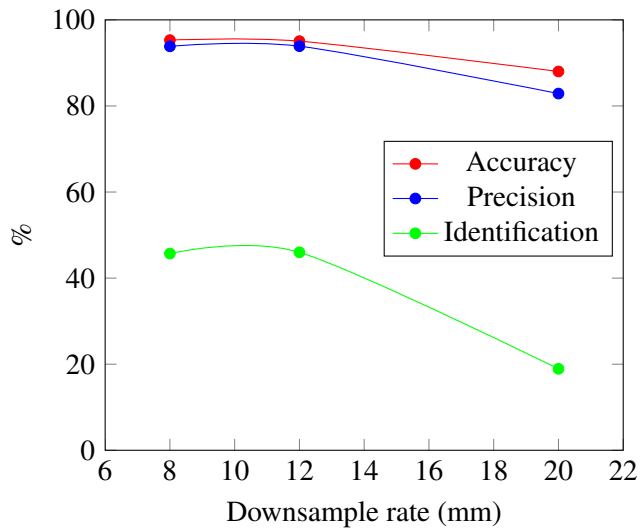

Figure 3.12: The effect of downsample rate on accuracy, precision, and identification rate is plotted. Downsampling with the rate of 12 *mm* has provided sufficient points for robust estimation. Further decrease of the downsampling rate shows no more accuracy gain.

Size and displacement of feature boxes

As explained in Section 3.3, each feature is a 3D box with a displacement from the reference voxel. Each dimension of the box is randomly chosen from the interval $[1, max\_size]$. Each dimension of the displacement vector is randomly chosen from the interval $[-max\_displace, max\_displace]$. The parameters max_size and $max\_displace$ need to be large enough to provide meaningful description of the area around the reference voxel. On the other hand, the value of the feature will be undefined when the distance of the voxel to the edge of the image is less than $max\_displace + max\_size/2$. We disregard the votes of these voxels that have some meaningless feature values. Hence, too large values for these parameters may cause meaningless feature values for significant number of voxels. In our experiments, we used $max\_displace$ = 12 *mm*, and max_size = 32 *mm*. The values are obtained experimentally. Figure 3.13 shows the effect of maximum box size on the perfor-

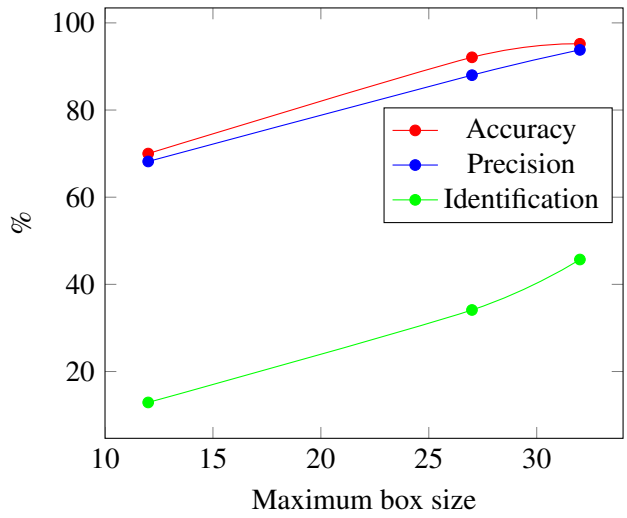

Figure 3.13: Increasing the maximum box size up to 32 *mm* improved the performance. A larger value for this parameter causes it to go out of bound in several small images of the dataset. Going out of bound means that all votes are disregarded and no prediction is made.

mance of the system.

PCA whitening

Principal Component Analysis (PCA) is a dimensionality reduction algorithm that aims to convert a set of possibly correlated variables into a set of linearly uncorrelated variables [24]. In machine learning, PCA is widely used for enhancing unsupervised feature learning by reducing the dimension and also removing the correlation between features [2, 30, 56]. There is a closely related step called *whitening* that aims to reduce redundancy of the input data. More formally, PCA uses Singular Value Decomposition (SVD) of the data and keeps the most significant singular vectors to project the data to a lower dimensional space. When we use PCA whitening the principal components are divided by the singular values to ensure uncorrelated outputs with unit variance. It has been shown that whitening can improve the accuracy of the downstream estimator in several applications [35, 42].

In our experiments, using PCA whitening for removing correlation and redundancy among the features caused better convergence of the training (better minimization of the cost function). However, on test data it showed poor performance. This observation may be a sign of overfitting. In machine learning, overfitting occurs when a function is too closely fit to a limited set of data points. It means that the model is describing the noise and random error of the training set instead of the underlying relationship that is expected to be seen in the test data. Figure 3.14 demonstrates the performance of the deep neural network for different regions of spine when using whitening as a preprocessing step. The results show slightly better identification rate for lumbar region (where we have more training examples), but poor performance in other sections.

Deep neural networks are prone to overfitting because of their complex structure and their high number of parameters. However, several methods such as regularization and dropouts, are proposed to alleviate the overfitting issue in deep learning [11, 21, 73]. A future work may involve trying to improve the localization results by using PCA whitening along with regularization or dropouts techniques which avoid overfitting.

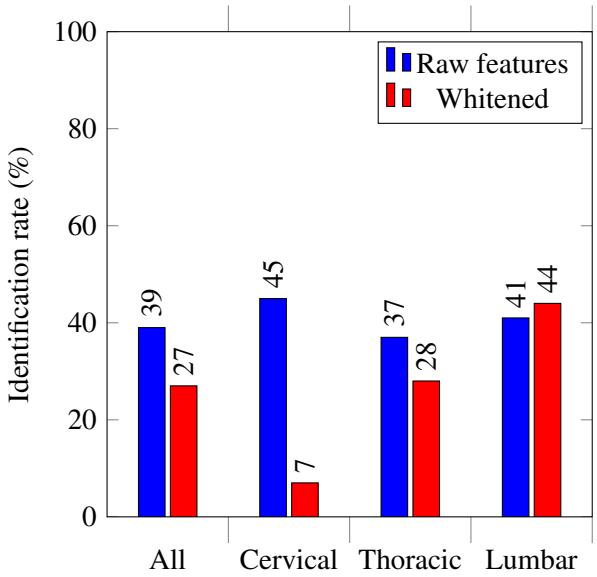

Figure 3.14: Using PCA whitening as an additional preprocessing step caused overfitting. The cost function is better minimized at the training time, but poor performance is observed on test data. The results are slightly better in lumbar region where there has been more training examples.

3.4.2 Parameters in Training Deep Neural Network

Neural network structure

One of the main challenges of training a deep neural network is to optimize the number of hidden layers, and the number of hidden units in each layer. More layers and more hidden units increase the ability of the network to estimate more complex relationships in the data, while make the network more prone to overfitting. Our network structure (shown in Figure 3.6) is obtained experimentally. Because of the long training time (about two days in our experiments) sweeping through all parameters was not feasible. Other than trial and error, no reliable shortcuts were found in the literature for this purpose. Informal tutorials suggest that one should start with one hidden layer with a number of units in the order of the number

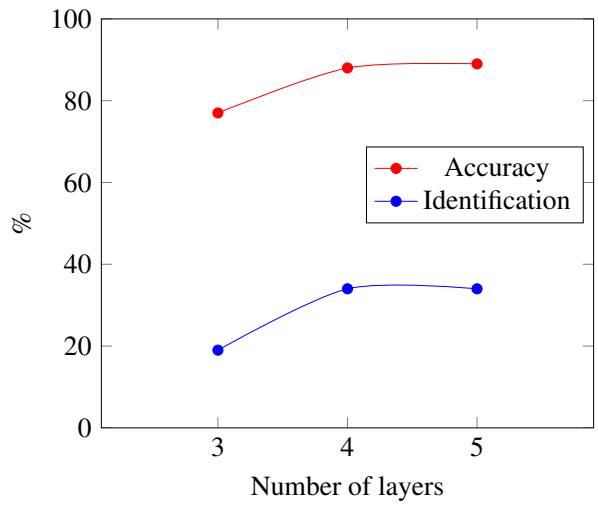

Figure 3.15: The accuracy and identification rates obtained from using different numbers of hidden layers. Increasing the number of hidden layers from 4 to 5 did not have a large effect on the final results.

of inputs, and then use trial and error to optimize these parameters. Figure 3.15 and 3.16 illustrate the effect of number of layers and neurons per layer in our experiments, respectively.

Activation function

Logistic sigmoid, and hyperbolic tangent were widely used as the activation function of the hidden units in traditional neural networks [34]. In recent years, it has been shown that rectifier functions can perform better than conventional functions in terms of estimation accuracy and the speed of convergence [11, 40, 76]. A rectifier function is simply defined as $f(x) = max(0,x)$ where x is the input of the hidden unit. A neuron that uses this function for activation is called a *rectified linear unit*. training a neural network of rectified linear units is computationally efficient because of: 1) the simplicity of the function 2) the sparse activation (in a randomly initialized network about half of the hidden units have zero output). It is argued in [18] that this sparsity also makes these units more biologically plausible.

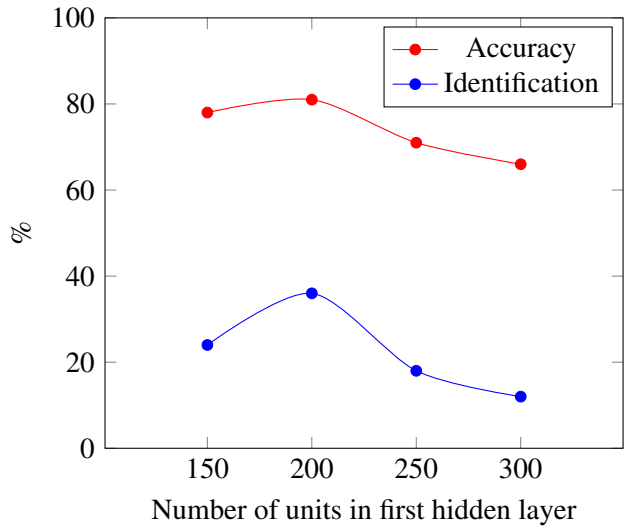

Figure 3.16: The accuracy and identification rates obtained from using different numbers of neurons in the first hidden layer. The system showed high sensitivity to this hyper-parameter.

Using rectified linear units in our deep neural networks led to higher accuracy and also faster convergence. Figures 3.17 shows a comparison of the activations functions in the context of experiments of this chapter.

Optimization method

Stochastic Gradient Descent (SGD) [31] is widely used for optimizing the cost function in training neural networks. In our experiments, cost function is defined as in Equation 3.4 for each sub-network that are trained greedily (explained in Section 3.3). The gradient of cost function with respect to each connection weight is computed using backpropagation algorithm [70], and SGD is used for optimization.

In early experiments, when not using layerwise pre-training, Hessian-free (HF) optimization [37] performed significantly better than SGD for training the whole network altogether on a small subset of data. However, this approach was too time-consuming and consequently impractical for training on a large training set. On the

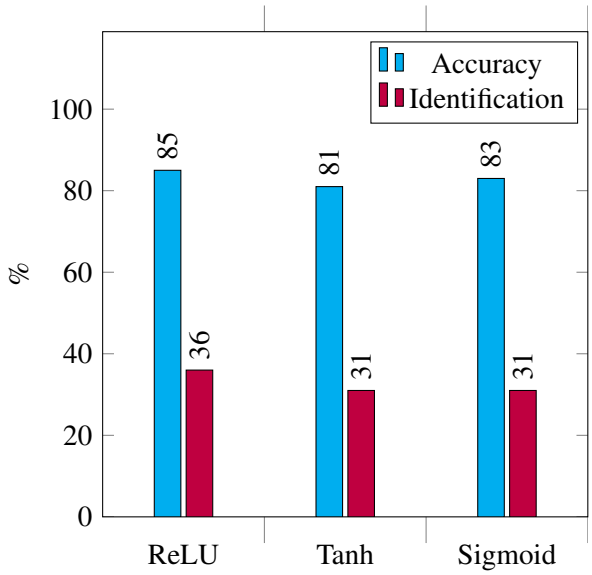

Figure 3.17: The accuracy and identification rates obtained from using different activation functions for hidden layers. Using RELU was faster and also led to better results.

other hand, SGD led to desirable convergence along with layerwise pre-training on larger datasets.

Stochastic gradient descent has several parameters that need to be tuned for best convergence. *Learning rate* is the step size when changing parameters. *Momentum* is used to increase speed when the surface of cost function is highly non-spherical. *Batch size* is the size of the subset of training data that used to optimize the cost function at each iteration. Above-stated hyper-parameters are obtained empirically. A typical approach is to plot the cost function versus SGD iterations, and choose the parameters that lead to rough consistency in decreasing the cost function over time. Figure 3.18 plots the convergence of SGD over time for each steps of the deep network training.

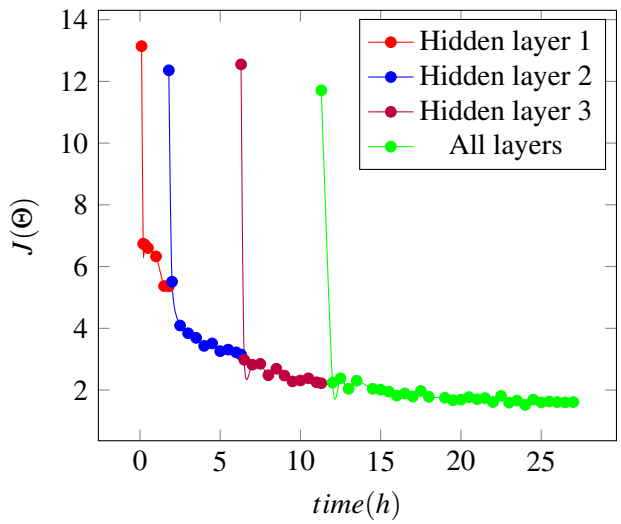

Figure 3.18: Convergence of the deep neural network. SGD is used along with layerwise pre-training. Adding a layer in each step led to further minimization of the cost function.

3.4.3 Parameters in Aggregating Votes

Once we have the votes of thousands of voxels of the image for a location of a specific vertebra, we aggregate them using the Kernel Density Estimation (KDE) approach presented in [6] to obtain a single prediction for the location of the vertebra. Figure 3.19 demonstrate the importance of vote aggregation by comparing the accuracy of the KDE approach against the basic measures of central tendency like mean, trim-mean, and median. Trim-mean means that we remove first and last 20% of data and then get the mean over the rest.

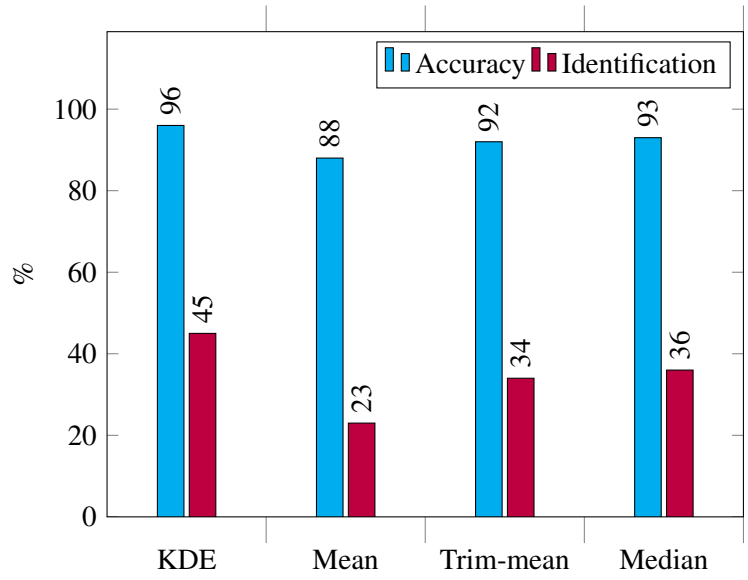

Figure 3.19: The accuracy and identification rates obtained from using different approaches for aggregating votes.

3.5 Results and Discussion

Two-fold cross-validation is performed on two non-overlapping sets of volumetric CT scans with 112 images each. The results on data from fold 1 were obtained after training the deep neural network on fold 2, and vice versa. The folds are selected exactly as in [16] to enable comparing the results.

After aggregating the votes of the voxels, we may conclude that a specific vertebra is outside of the scope of the image. If expert annotation (ground truth) confirms that the vertebra is not visible in the image, we consider it as a True Negative (TN). Otherwise, if the image contains the vertebra it will be a False Negative (FN). Similarly, if our system concludes that the vertebra in inside the scope of the image and the expert annotations confirm, it is considered as a True Positive (TP). Otherwise, it will be a False Positive (FP). Based on these observations the predictions are evaluated in terms of accuracy, precision, sensitivity, and

specificity which are defined as follows:

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN}, \tag{3.5}$$

$$Precision = \frac{TP}{TP+FP}, \tag{3.6}$$

$$Sensitivity = \frac{TP}{TP+FN}, \tag{3.7}$$

$$Specificity = \frac{TN}{TN+FP}. \tag{3.8}$$

Localization error is defined as Euclidean distance between the estimated centroid of a vertebral body and its expert annotation. A vertebra is defined as correctly identified if the localization error for that vertebra is less than 2 *cm*, and the closest expert annotation to its predicted vertebra location is the correct one. The same evaluation criteria are used in [16] and [15].

3.5.1 Separate Network for Z Coordinate

In the coordinate system of our experiments, the z coordinate is the direction of the length of vertebral column. In a single image, centroid of adjacent vertebrae have almost similar x and y, but different z coordinate. Therefore, the range for the z coordinate of the distance vectors (which are the targets of the deep neural network) is much larger than x and y coordinates. On the other hand, for an efficient optimization, we need to scale all the targets of the neural network to the same range. This will cause the optimization approach to concentrate more on minimizing the error in x and y coordinates whereas the z coordinate is more important for vertebra identification. Hence, we added another deep neural network for predicting only the z coordinate of the targets. The effect of adding this additional neural network is summarized in Table 3.1.

3.5.2 Point Selection Using Canny Edge Detector

In our method, each voxel of the image equally contribute to the predicted location of a certain vertebra. However, some images have some dark area that no information about the vertebrae can be obtained from the area around them. Presence of these points in an image may reduce the accuracy of the method. Hence, we

Table 3.1: Localization errors (in mm) for using a single neural network for all coordinates vs. using a separate neural network for the z coordinate.

Region	Single neural network			Separate network for z coord.		
	Mean	Std	Id. rates	Mean	Std	Id. rates
All	23.0	12.6	37.1%	20.7	11.9	44.8%
Cervical	22.6	11.2	26.7%	17.9	9.3	41.4%
Thoracic	22.9	13.2	37.3%	20.8	12.5	45.3%
Lumbar	23.4	13.0	46.3%	23.2	12.3	44.9%

Table 3.2: Localization errors (in mm) for using a Canny edge detector for point selection vs. performing the analysis on the votes of all voxels without any type of point selection.

Region	With point selection			Without point selection		
	Mean	Std	Id. rates	Mean	Std	Id. rates
All	20.7	11.9	44.8%	21.5	12.5	40.1%
Cervical	17.9	9.3	41.4%	19.3	9.9	35.0 %
Thoracic	20.8	12.5	45.3%	20.8	12.9	45.1 %
Lumbar	23.2	12.3	44.9%	23.8	12.4	41.6 %

attempt to alleviate this issue by only using the voxels that are close to detected edges. Only the votes of these voxels are aggregated for predicting the location of vertebrae. Table 3.2 summarizes the capability of this point selection step in improving the localization results. We expect this step to be more effective when the images are not tightly cropped.

3.5.3 Refinement by Local Vote Aggregation

Once we have the predictions for each vertebra in the image, we refine the predictions by aggregating only the votes which are close to the predicted location. For each visible (according to the prediction) vertebra in the image, the points around itself and its adjacent vertebra are aggregated using Kernel Density Estimation. The previous predicted point is then replaced by the point that is obtained from

Table 3.3: Localization errors (in mm) before and after using local vote aggregation to refine the predictions.

Region	Before Refinement			After Refinement		
	Mean	Std	Id. rates	Mean	Std	Id. rates
All	20.7	11.9	44.8%	18.2	11.4	54%
Cervical	17.9	9.3	41.4%	17.1	8.7	43.1%
Thoracic	20.8	12.5	45.3%	17.2	11.8	55.3%
Lumbar	23.2	12.3	44.9%	20.3	12.2	56.9%

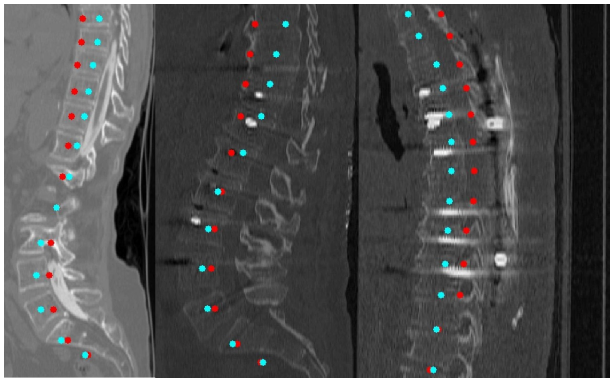

Figure 3.20: Visual results of the local vote aggregation step are shown on the mid-sagittal plane of three example images. The localization points before refinement are shown in red while the points after refinement by local vote aggregation are shown in cyan. Refined points have a better representation of the spine curvatures.

this step. Table 3.3 and Figure 3.20 demonstrate the effect of this refinement step on our results. The visual results show that after this step the predictions are usually fit better to the curvature of spinal column. It is mostly because of the fact that the curvature in one part of spine cannot be predicted from the points on other parts of the column.

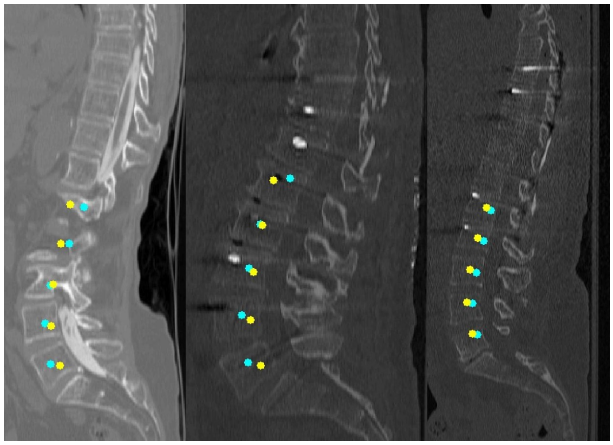

Figure 3.21: Visual results of the shape+pose model refinement step are shown on the mid-sagittal plane of three example images. The localization points before model refinement are shown in cyan while the points after model refinement are shown in yellow. The improvement is minimal and may not be worth the computational expense.

3.5.4 Capability of Our Shape+pose Model for Further Refinement

In this step, we attempt to initialize our shape+pose model of lumbar spine by the predictions of our method, register the model to the edges of the image in order to refine the localization and also provide a vertebrae segmentation. Although this approach improved the identification rate by 1%, the results were not satisfactory (See Figure 3.21). The main reason is that the presence of image artifacts (mostly caused by surgical implants) do not allow us to obtain a high-quality edge map for model registration. Due to presence of these artifacts, the method fails to automatically find a suitable sensitivity threshold for the Canny edge detector. Figure 3.22 illustrates three examples of Canny edge detector output. The other reason is that the model (trained on mostly healthy subjects [50]) cannot accommodate pathological spines in this dataset.

We do not include this step in our final results, since it causes little improvement with the expense of high computation cost (about two minutes per image).

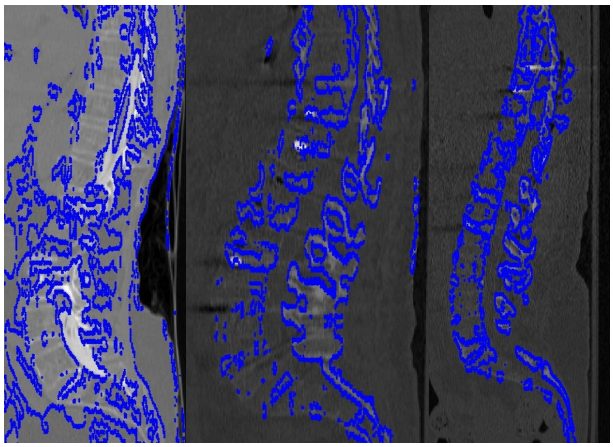

Figure 3.22: The edge map that is obtained by Canny edge detector in the mid-sagittal plane of three example images. Image artifacts and unclear boundaries adversely affect the quality of the extracted edge maps.

3.5.5 Final Results

Our final localization and identification results are presented in Table 3.4. The results show the capability of our method for determining the visible part of spine and also identifying each visible vertebra. Since the width of vertebrae are generally smaller in the cervical region, similar mean error may cause lower identification rate in this region. Because our method is computationally fast (Potentially less than 3 seconds on a desktop computer), it can be used as a preprocessing step in a wide range of clinical procedures.

3.5.6 Comparison to State-of-the-art

Our results are compared to [15] and [16] which are the state-of-the-art methods for vertebra localization and identification in CT scans. All three approaches are evaluated on the same dataset with a two-fold cross validation with exactly the same fold separation.

Our approach is the most similar to [15]. In this work, they first use a voting

Table 3.4: Localization results for different regions of the vertebral column. Mean error and STD are in mm.

Region	Accuracy	Precision	Sensitivity	Specificity	Mean	STD	Id. rates
All	96.0%	94.4%	97.2%	95.0%	18.2	11.4	53.6%
Cervical	96.0%	91.2%	97.8%	95.0%	17.1	8.7	43.1%
Thoracic	95.1%	93.9%	95.9%	94.5%	17.2	11.8	55.3%
Lumbar	98.1%	97.5%	99.4%	96.1%	20.3	12.2	56.9%

framework based on a Regression Forest (RF) to obtain a rough localization, and then they use a graphical model based on a Hidden Markov Model (HMM) to refine the predictions. The results of their method on this public dataset is provided in [16]. They achieved state-of-the-art results on a dataset of non-pathological cases. However, the performance of their method degrades significantly in this public pathological dataset (which we also used). A possible reason is that the graphical model cannot accommodate high variations in pathological cases and consequently fails to refine the predictions in this dataset. The results of our method is compared to theirs in Table 3.5. The comparison demonstrates that using deep learning can eliminate the need for model-refinement which has relatively high computational cost and is not as robust in pathological cases.

On this dataset, the state-of-the-art localization results are recently presented in [16] using a method based on a Classification Forest (CF). A comparison between our results and theirs is summarized in Table 3.6. While their approach has a higher identification rate, our method outperforms theirs in terms of accuracy, precision, sensitivity, and specificity. In addition, our algorithm can be computationally faster.

The algorithms presented in [15] and [16] take about 2 minutes and 1 minute per image, respectively. In [15] a joint shape and appearance model in several scales is used to refine the predictions. In [16] centroid density estimates based on vertebra appearance are combined with a shape support term to remove the false positive predictions from all over the image. Our method does not require above-stated computationally-intensive steps. Our reported results are generated using a vectorized MATLAB implementation for feature extraction which takes

Table 3.5: Comparison of the localization results of our method and the method presented in [15] which uses regression forests followed by a refinement using Hidden Markov Models. The same training and test sets are used in evaluations of both methods. Mean error and STD are in mm.

Region	Mean		STD		Identification	
	Ours	RF+HMM	Ours	RF+HMM	Ours	RF+HMM
All	**18.2**	20.9	**11.4**	20.0	**54%**	51%
Cervical	17.1	**17.0**	**8.7**	17.7	43%	**54%**
Thoracic	**17.2**	19.0	**11.8**	20.5	55%	**56%**
Lumbar	**20.3**	26.6	**12.2**	19.7	**57%**	42%

about 1 minute per image. However, an efficient version of the feature extraction step is recently implemented in C++ (Bill Liu, Vancouver, BC) which works in a fraction of a second per image. The descriptions and complexity analyses of both implementations are provided in Section 5.1. After integration with the C++ implementation of the feature extraction step, the computation time for our method will be reduced to less than 3 seconds per image. This makes our method a better candidate than the methods proposed in [15] and [16] for a range of image-guided procedures that cannot tolerate long execution time.

For deep neural network part, we used Theano-nets package (Leif Johnson, Austin, TX) built on top of Theano library [5] in Python. The other parts of the algorithm are implemented in MATLAB. Testing time for two deep neural networks is less than a second per image. Two levels of vote aggregation (first prediction and then local refinement) take about one second per image. Feature extraction takes about one minute using the MATLAB implementation. However, the C++ implementation of the feature extraction step works in less than one second.

Table 3.6: Comparison of the localization results of our method and another method based on classification forests which is considered as state-of-the-art for this dataset [16]. The same training and test sets are used in evaluations of both methods. Mean error and STD are in mm.

	Accuracy	Precision	Sensitivity	Specificity	Mean	STD	Id. rate
Ours	**96.0%**	**94.4%**	**97.2%**	**95.0%**	18.2	**11.4**	53.6%
CF	93.9%	93.7%	92.9%	94.7%	**12.5**	12.6	**70.6%**

3.6 Summary

A fully-automatic method based on deep learning is developed for simultaneous localization and identification of vertebrae in three-dimensional CT scans. The results show that the method can handle arbitrary-field-of-view scans where it is initially unknown that which part of vertebral column is visible in the image and to what extent. State-of-the-art results are achieved on a large public dataset of pathological cases. The fact that the proposed method is computationally faster than the previous work make it potentially more appealing to be used in computer-aided systems for spine diagnosis and therapy.

Chapter 4

Automated Localization, Identification, and Segmentation in MRI

4.1 Introduction

Localization and labeling of vertebrae is a crucial step in diagnosis and therapy of spinal diseases such as slipped vertebra, disk/vertebra degeneration, and spinal stenosis. Vertebra segmentation is also a pre-processing step in diagnosis of spine pathologies like scoliosis and osteoporosis. Hence, building a computer-based system for spine diagnosis and therapy requires automatic localization, labeling and segmentation of vertebral structures.

Lower back pain, one of the most common types of pain, is usually caused by lumbar region of the spine. Lumbar vertebrae are mostly viewed by X-rays, MR , or CT. Among these three, MR imaging shows the soft tissue better and also does not expose the patient to ionizing radiations. This has led to increased interest in MR technologies for imaging the spine in recent years. Automatic labeling and segmentation of vertebral bodies in MR images is challenging because of: 1) variation among images in terms of field-of-view, 2) repetitive nature of vertebral column, 3) lower contrast between bony structures and soft tissue compared to CT,

4) larger inter-slice gap in clinical MR images compared to CT, and 5) presence of field inhomogeneities in MR images.

Several researchers have investigated the localization and labeling problem in CT [15, 20, 32, 36, 39, 51] and MR [1, 44, 48, 58] images. Most methods make assumptions about which vertebrae are visible in the scan. Alomari et al. [1] assumed approximate alignment between scans. Klinder et al. [32], Glocker et al. [15], and Rasoulian et al. [53] claimed handling general scans. However, all three of these methods are applied on CT scans and not MR images. In addition, Glocker et al. only provide labeling and do not provide segmentation. Klinder et. al. and Rasoulian et al. provide segmenation, but their algorithms have high computation time for arbitrary-field-of-view scans.

In Chapter 2 [66], we proposed a semi-automatic segmentation approach for MR images based on a multi-vertebrae statistical shape+pose model. It required one user click for each vertebra to be segmented. In this work, we propose a method based on deep learning for simultaneous localization and identification of vertebral bodies in MR images. Then, we initialize our previous segmentation method with the localized points in order to obtain a fully-automatic segmentation.

4.2 Materials

The proposed method was evaluated on the same dataset that is used in Chapter 2. The dataset consists of nine T1-weighted multi-slice MR images. All images contain lumbar region, but the extent of visibility of thoracic and sacrum regions varies. No pathology was observed in this dataset. Slice thickness ranges between 3.3 to 4.4 *mm* among images. The size of all slices are 512×512 pixels with in-plane resolution of 0.5 *mm*. The number of slices of each volumetric image varies from 12 to 18. The statistical model (used for segmentation) was constructed from manually segmented multi-slice CT scans of 32 patients [50]. Segmentation ground truth meshes (also the same as Chapter 2) were prepared manually using ITK-SNAP [74] from raw images. The center of gravity of each manually segmented vertebral body is used as localization landmarks.

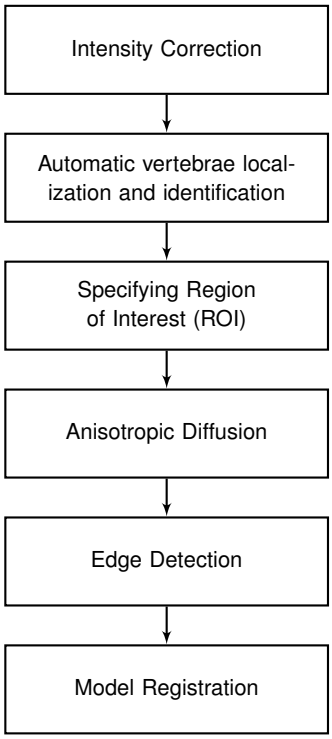

Figure 4.1: Flowchart of the automatic framework for segmentation of vertebral bodies.

4.3 Methods

4.3.1 Automatic Localization and Identification

Pre-processing: bias field correction

The presence of intensity inhomogeneities in MR images can adversely influence the quality of intensity-based feature extraction. Hence, we first apply a bias field correction algorithm on MR images to reduce this inhomogeneity. Statistical Parametric Mapping package is used for bias field correction (SPM12b, Wellcome Department of Cognitive Neurology, London, UK).

Localization and identification by deep learning

The localization task is parametrized as a multi-variate regression problem. Each voxel of the image is described by hundreds of intensity-based features. Each feature is the difference between the mean intensity over two cuboids displaced with respect to the reference voxel position. Dimensions and displacement of each feature are generated randomly. These features are fast to compute using the integral image idea proposed by Viola et al. [69]. The targets of the multi-variate regression are the relative distances between the reference voxel and the centers of lumbar vertebral bodies (Parametrization of image localization task as a regression problem, and also randomly generated cubic features are explained in more detail by Criminisi et al. [10]). A feed-forward neural network with three hidden layers is trained for solving the multi-variate regression problem. Stochastic gradient descent along with layerwise pre-training is used for optimization. On a test image, we first extract features from all voxels, then we use the neural network to predict the relative distance of the labels with the respect to each voxel. Each of these relative distances are converted to absolute positions of the labels and considered as the vote of that specific voxel for the position of each vertebral body. The votes of all pixels are aggregated using Kernel Density Estimation [6] to obtain a robust prediction of the center of each lumbar vertebral body. No assumptions are made about presence of the target vertebrae in the volumetric image. The voxels may vote outside of the scope of the image if the target vertebrae are not visible.

Refinement by local thresholding

In most of the cases, the predicted points from the deep learning approach are located inside of the target vertebral bodies, but not exactly centered. We attempt a simple approach to bring the predicted points to the center of vertebral bodies in order to improve identification and initialization of our statistical model for segmentation. By Otsu thresholding [46] in the region of predicted points, we find the large components that may be vertebral bodies. If a large component is found close to the prediction, we refine the prediction by replacing it with the center of that component. Figure 4.3 illustrates an example of the refinement step.

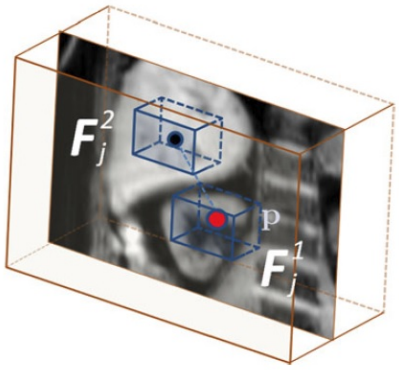

Figure 4.2: The value of each feature is the difference between the mean intensity over two cuboids displaced with respect to the reference voxel position. From [10].

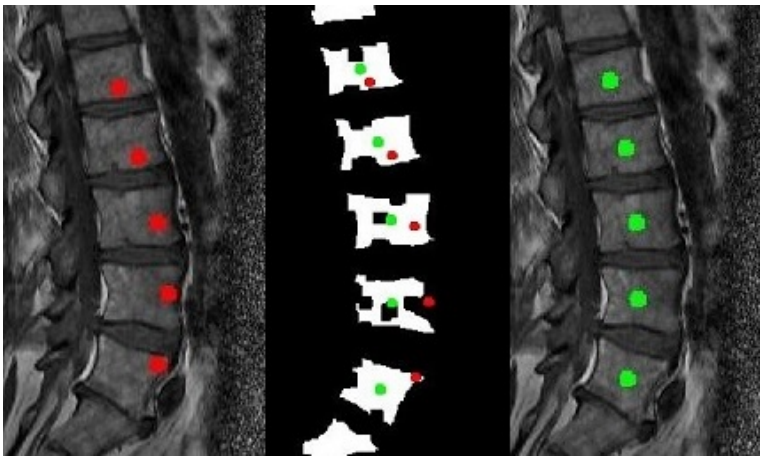

Figure 4.3: Refining localization points by replacing them with the center of the closest large component, obtained from local thresholding. Left: Localized points before refinement. Middle: Refinement using components obtained from local thresholding. Right: Localized points after refinement.

4.3.2 Segmentation

Pre-processing: anisotropic diffusion

A three-dimensional anisotropic diffusion [49] filter is applied to the bias-field corrected image for image smoothing while preserving edges. This pre-processing step highly improves the quality of edge detection. For speed optimization, we only apply the filter to the region of spinal column detected by previous steps. More details of the pre-processing steps are described in Section 2.3.

Statistical model registration

We use Canny edge detection algorithm to obtain the edges in the detected spinal column region. The maximum value of the gradient magnitude of the region is used to automatically obtain the sensitivity threshold of the edge detector algorithm. Then, we use an iterative Expectation Maximization registration method (introduced by Rasoulian et al. [50]) to register a statistical model to Canny edges in order to obtain a robust segmentation. The statistical multi-vertebrae model integrates variations of shapes and poses of lumbar vertebrae that are separately extracted from 32 manually-segmented 3D CT scans [50]. The multi-vertebrae registration of the model allows us to avoid mis-segmentation in the area between vertebrae by simultaneous registration of all visible vertebrae and taking into account the correlation between shapes and poses of different vertebral bodies.

4.4 Results and Discussion

Localization and identification results on nine subjects are reported in Table 4.1. The centroids of five lumbar vertebral bodies (L1 to L5) are predicted using leave-one-out cross validation. The distance to the ground truth landmarks are computed. The average and standard deviation of these distances as well as identification rate are reported. We consider a vertebral body to be identified when its distance to ground truth landmark is less than 2 *mm* and the closest landmark is the correct one. The results illustrate 100% identification after refinement step for 45 lumbar vertebral bodies.

Three-dimensional segmentation results are shown in Table 4.2. The closest

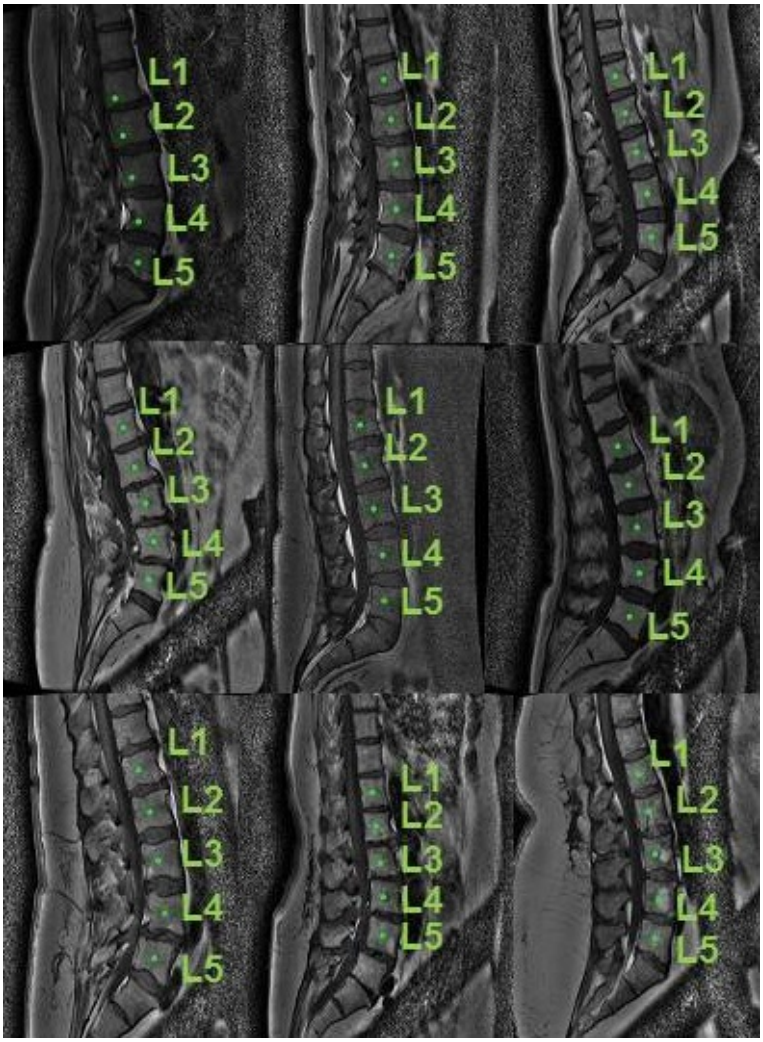

Figure 4.4: Localization and labeling results on the mid-sagittal slice of bias-field corrected images.

Table 4.1: Localization and identification results for lumbar vertebral bodies in nine volumetric MR images.

	Mean error	Standard deviation	Identification
Deep Learning Localization	11.9 mm	6.3 mm	91%
After Refinement	3.0 mm	2.4 mm	100%

distance to the registered model is found for each point in the manual segmentation. The average and maximum of these distances are reported as respectively mean error and Hausdorff distance. These results demonstrate that we can automatically segment the lumbar vertebral bodies in volumetric MR images with mean error of below 2.8 *mm* which shows improvement over our previous semi-automatic method. Some examples of our identification and segmentation results are shown in Figures 4.4 and 4.5.

The localization and identification step (including refinement) takes less than 20 seconds on a desktop computer. The whole process takes less than 3 minutes using an inefficient MATLAB (The MathWorks, Inc., Natick, MA) implementation for segmentation. Training a deep neural network takes about one day which can be adjustable by downsampling the training data. Deep learning part is implemented in Python using Theano-nets package (Leif Johnson, Austin, TX) built on top of the Theano library [5].

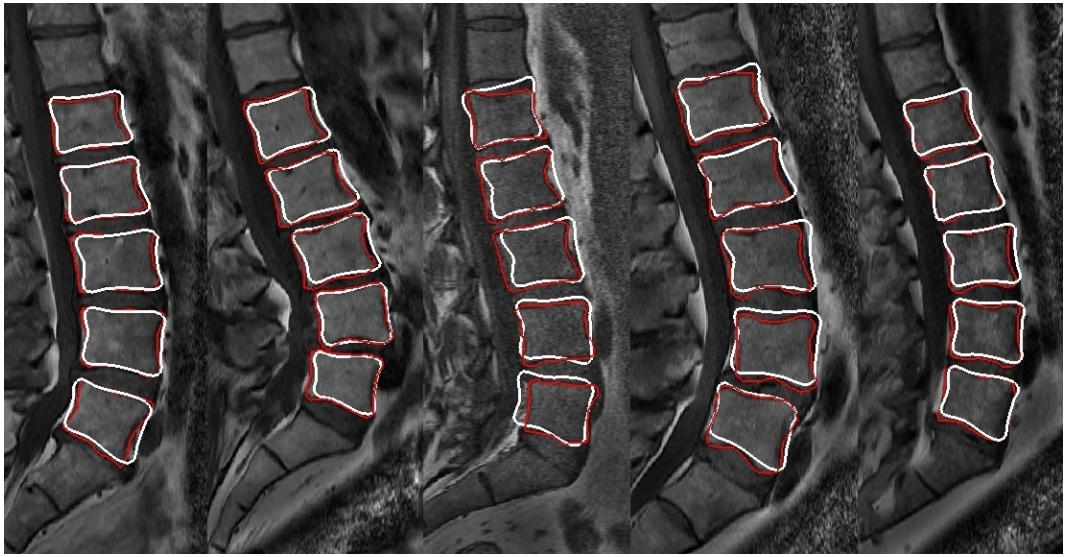

Figure 4.5: Examples of segmentation results in the mid-sagittal slice of five different patients. Our segmentation results are shown with white contours, while red contours show the manual segmentation.

Table 4.2: Mean error and maximum distance (Hausdorff) of segmentation error (in mm) for each vertebral body.

Subject	L1		L2		L3		L4		L5		All lumbar vertebrae	
	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean $\pm$ std	Max
1	2.0	6.1	1.7	5.2	2.8	9.5	4.4	15.7	4.2	16.4	3.0 ± 1.2	16.4
2	2.2	6.5	2.2	8.0	2.0	7.3	1.8	9.6	3.4	16.6	2.3 ± 0.6	16.6
3	3.7	8.6	3.4	8.7	3.6	9.6	4.0	13.6	5.8	19.9	4.1 ± 1.0	19.9
4	2.6	8.1	1.9	7.1	2.6	10.5	1.6	8.3	2.0	9.1	2.1 ± 0.4	10.5
5	2.2	8.0	1.6	5.5	1.6	6.4	2.2	11.0	4.6	17.7	2.5 ± 1.2	17.7
6	2.7	10.3	2.6	9.7	2.8	9.3	2.5	10.4	4.0	16.1	2.9 ± 0.6	16.1
7	1.8	5.5	1.7	5.6	2.2	7.9	2.7	11.9	3.5	15.2	2.3 ± 0.7	15.2
8	3.2	8.4	3.3	9.5	3.2	10.4	3.6	12.7	2.6	11.0	3.2 ± 0.4	12.7
9	2.3	7.3	1.7	6.5	2.0	8.0	1.9	8.8	3.1	12.4	2.2 ± 0.5	12.4
Average	2.5	7.7	2.2	7.3	2.5	8.8	2.7	11.3	3.7	14.9	2.7 ± 0.9	19.9

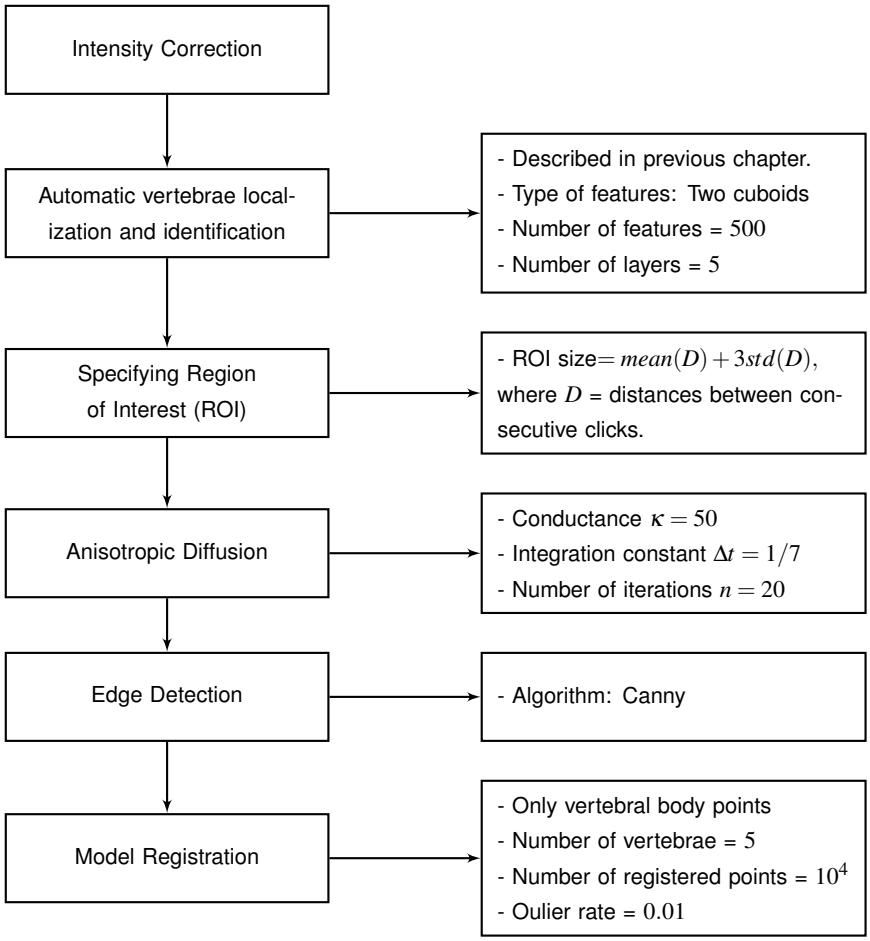

Figure 4.6: Summery of the parameters of the automatic framework for segmentation of vertebral bodies. The values of the main parameters of each step are shown in the boxes at the rightmost column.

4.5 Summary

A fully-automatic approach based on deep learning and statistical models is proposed for localization, labeling and segmentation of vertebral bodies in multi-slice MR images. Although the experiments are done only on the lumbar region, no assumptions are made about presence of specific vertebrae in the method. The multi-vertebrae model can handle large inter-slice gap (about 4 *mm*) in clinical MR images with low computation cost. Results demonstrate that our method can automatically localize, label, and segment the vertebral bodies in MR images with sufficient accuracy and speed for a wide range of clinical applications.

Chapter 5

Speedup by Vectorization and GPU Acceleration

5.1 Vectorized Feature Extraction

While compiled programming languages like C/C++ are widely used for software release, interpreted languages like MATLAB and Python are more popular in research. It is mainly because their tools and built-in math functions enable the researcher to explore multiple approaches and reach a solution faster than traditional compiled languages. MATLAB, in particular, is optimized for operations involving matrices and vectors. On the other hand, it performs much slower than compiled language in executing loops over multi-dimensional data. This is one of the reasons that code vectorization is highly recommended when using such languages. In this context, vectorization means revising loop-based, scalar-oriented code to vector and matrix operations. Other than better performance, a vectorized code is easier to read and less error prone.

As explained in Chapters 3 and 4, for the task of vertebrae localization, we extract a set of features from image voxels to feed our deep learning system. In this section, we explain the key points of our contribution in providing a vectorized implementation of this feature extraction step. The research and development benefits of this approach is analyzed by a comparison against a basic sequential implementation.

5.1.1 Description of Features

For vertebrae localization methods presented in Chapters 3 and 4, hundreds of intensity-based features are extracted from each voxel of the image (with some downsampling for computational speedup). The features describe a short-range area around the reference voxel.

In CT scans, the value of each feature is the mean intensity over a three-dimensional cuboid displaced with respect to the reference voxel position. In MR images, because of the presence of intensity inhomogeneities, the mean intensity over two cuboids are subtracted and used as the value of each feature (See Figures 3.2 and 4.2). The features used in CT and MR are explained in more detail in Sections 3.3 and 4.3. For simplicity, in this section we focus on the implementation alternatives for CT features. MR features are implemented similarly by using two cuboids instead of one.

Computing the sum of intensities over cuboids can be done efficiently using an intermediate representation of the image, called the integral image [69]. Mathematical definition is brought in Equation 3.2 and a 2D representation is shown in Figure 3.3. Having the integral image, the sum of intensities over a cuboid in the original image can be computed in constant time.

The first step is to generate specifications for M features. Each feature box F_i is defined by its displacement $D_i \in \mathbb{R}^3$ and its dimensions $S_i \in \mathbb{R}^3$. The displacement is defined as the center of the feature box in respect with the reference voxel position. D_i and S_i are selected randomly from the ranges $[-max\_displace, max\_displace]$ and $[1\ mm, max\_size]$, respectively. The integral image, $II_{N_1 \times N_2 \times N_3}$ has the same size of the original image denoted as $I_{N_1 \times N_2 \times N_3}$. In our experiments, $M = 500$ features are specified. Note that this number of features need to be extracted for each image voxel. Therefore, for the whole image we may extract billions of features.

5.1.2 Sequential Implementation

In the sequential approach, we simply iterate over image voxels for each feature. Using the integral image, extracting the mean intensity of a feature box can be done in constant time . Assuming M features and an $N \times N \times N$ image, the computational cost for feature extraction per image will be MN^3.

Algorithm 1 Sequential feature extraction

```
for i = 1 → M do                                  ▷ Loop over all features
    for x = 1 → N_1 do                            ▷ Loop over x axis
        for y = 1 → N_2 do                        ▷ Loop over y axis
            for z = 1 → N_3 do                    ▷ Loop over z axis
                F_i ← mean intensity of the feature box   ▷ Constant time
```

5.1.3 Vectorized Implementation

In the vectorized version, we quantize the range of allowed box sizes and displacements, respectively, in sets

$$\mathcal{S} = \{\{2,2,2\}\{2,2,7\}\{2,2,12\},...,\{max\_size, max\_size, max\_size\}\}, \tag{5.1}$$

and

$$\mathcal{D} = \{-max\_displace,...,-8,-4,0,4,8,...,max\_displace\}. \tag{5.2}$$

The values in the above sets are in mm. For each $S_i \in \mathcal{S}$, we extract the mean intensity of feature boxes with size S_i from all voxels of the image. Using element-wise matrix operations on the integral image, we create a 3D matrix with the same size of the original image, where the value of each element of this matrix equals the mean intensity of the box with size S_i centred on the corresponding voxel on the original image. Quantizing displacements allow us to downsample these computed matrices for memory efficiency. Thereafter, the value of each feature F_j for all voxels, can be obtained rapidly by picking the already-computed 3D matrix for S_j and reshaping it according to D_j. Figure 5.1 visually illustrates the steps of this algorithm on a 2D example.

Algorithm 2 Vectorized feature extraction

```
for all S_i ∈ 𝒮 do
    B_i ← mean intensity of box sizes S_i centred on all pixels
for j = 1 → M do                                  ▷ Loop over all features
    F_j ← cropped B_j according to D_j
```

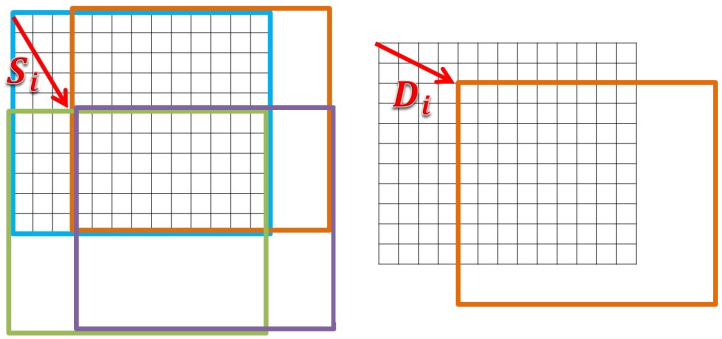

Figure 5.1: The proposed vectorization approach is shown on a 2D example. A feature box with size S_i and displacement vector D_i can be computed for all pixels of an image by element-wise matrix addition and subtraction on the integral image according to S_i (Left), and then cropping the product matrix according to D_i (Right). In the left figure, a number of element-wise matrix operations (*purple-orange-green+blue*) on the integral image results in a matrix in which each element contains the sum of boxes with size S_i started from the corresponding element in the original image. Out-of-bound parts of the matrices can be handled by zero-padding or cropping.

5.1.4 Comparison and Computation Analysis

In the experiments of Chapter 4, the number of features, M, equals 500. Considering downsampling rate of $[4\ mm, 4\ mm, 4\ mm]$ which is used in Chapter 4, the dimension of images in each direction, N, is mostly less than 80 pixels. The proposed sequential algorithm is $O(MN^3)$ which we expect it to be executed using a compiled programming language like C++ in the order of $500 \times 80^3 = 4 \times 10^6$ *cycles*, which should take less than a second on a desktop machine (with a commonplace 3-GHz processor).

Nevertheless, MATLAB is mostly an interpreted language and is relatively slow in running loops over multi-dimensional data. On the other hand, MATLAB is optimized for matrix operations. In our vectorized implementation instead of for-loops over dimensions of the image, we used element-wise matrix operations (addition, subtraction, and reshape). Figure 5.2 illustrates the substantial speedup

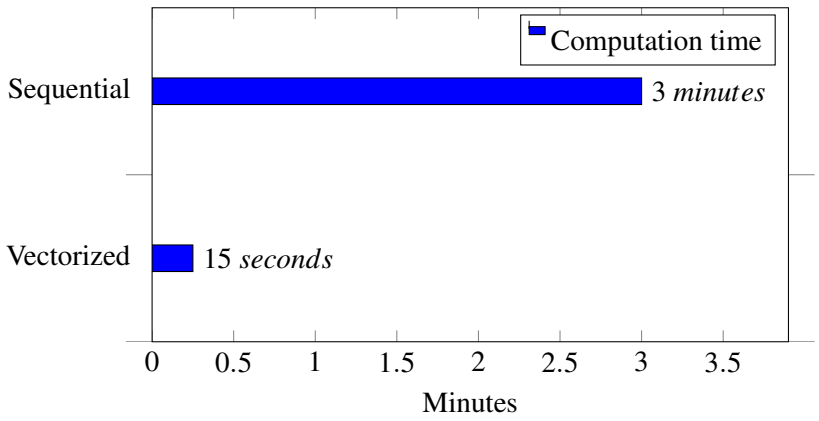

Figure 5.2: Computation time of the vectorized version and the sequential version of the algorithm on a sample image in MATLAB.

that we get from using the vectorized feature extraction algorithm in MATLAB with the setting of the experiments of Chapter 4. Note that the computational time reported for vectorized version is not affected by downsampling. Features are computed for all voxels and the downsampling rate is considered at the end to disregard the significant number of computed features. On the other hand, the computation time of the sequential approach is exponentially affected by downsampling rate ($O(N^3)$ when M is constant). Other than the performance gain, the vectorized implementation also makes the code easier to be read, maintained, and modified. Not being affected by downsample rate also let the researcher explore the effect of different downsample rates without recomputing the features.

5.2 GPU-accelerated Model Registration

The computation time of algorithms is a principal factor in the area of medical image analysis. Algorithms need to meet clinical time requirements to be used in clinics and operating rooms. Many algorithms with high accuracy and robustness cannot be used in clinical setting, because they do not meet clinical requirements in terms of speed. If a method does not work within clinically acceptable time-frame, it would not be valuable in clinical basis and cannot be used in practice. Parallel

computing can be used to make the algorithms to meet these time requirements.

In recent years, there has been a significant surge of interest in parallelizing medical image analysis algorithms on Graphics Processing Units (GPU) [29, 45, 77]. A GPU, typically, has a much higher number of processing elements than traditional single-core or multicore machines. Unlike conventional Central Processing Units (CPU), which did not have more than a handful of cores, GPUs include massively parallel arrays of processing units that can greatly increase the throughput of parallel programming. Using this technology may allow a wide range of image processing algorithms, which previously did not meet hard time requirements of clinical usage, to fit in these constraints.

The work presented in this section, in particular, aims to parallelize a statistical model registration method on processing units of a GPU. The method, proposed in [50] , is used in Chapters 2 and 4 for spine segmentation. This algorithm is parallelized by two different methods: 1) Compute Unified Device Architecture (CUDA) programming on a GPU, and 2) Multicore programming using shared memory. In first method, we take advantage of hundreds of processing elements available in a GPU for parallelizing, while in second method parallelizing is based on a 32 processing units available in a multicore machine.

5.2.1 Spine Segmentation Method

Our segmentation algorithm aims to find boundaries of each vertebra in spinal column in CT or MR images. For this purpose, first, a statistical shape model of spine is generated based on a number of manually segmented images. Then, given a new not-segmented image, we register our previously generated model to the new image to obtain a spine segmentation. In other words, the boundaries of each vertebra in the input image is found by aligning a statistical model to the detected edge points [50]. The concept of statistical models is described in detail in [13].

Improving the performance of this method up to real-time limits can massively expand its application for spine segmentation in CT, MR, and ultrasound images. To this end, we select the most computationally intensive portion of the algorithm and aim to efficiently parallelize its execution.

Computationally intensive part

In the most computationally intensive part of the algorithm, we have a set of N points in D dimensions, $X_{N\times D}$. We also have M Gaussian distributions with standard deviation of σ and different means. The means of these M Gaussian distributions are gathered in each row of matrix $Y_{M\times D}$. We need to build the probability matrix $P_{M\times N}$. The element P_{mn} in this matrix should be set to the probability that point n in $X_{N\times D}$ is generated by the mth Gaussian distribution that its mean is represented in mth row of $Y_{M\times D}$. Each element of P can be computed as follows:

$$P_{mn} = \frac{exp(\frac{-1}{2}||(Y_m - X_n)/\sigma||^2)}{(\sum_{k=1}^{M} exp(\frac{-1}{2}||(Y_k - X_n)/\sigma||^2)) + \varepsilon}, \tag{5.3}$$

where Y_m is the mth row of Y, X_n is the nth row of X, ε is a constant and the operator $||A||$ is defined as follows:

$$||A|| = \sqrt{AA^T}. \tag{5.4}$$

The next step is to find the best transformation that aligns the model (above-mentioned Gaussian distributions) to the data points (matrix $X_{N\times D}$). In order to solve this optimization problem, a loss function is defined. And the goal is to minimize this loss function. This function will be different for different types of transformations (e.g. rigid, rotation, affine, etc.). These different types of loss functions are described in [38]. In all cases, when we get the derivative of the loss function (for minimizing), terms PX, PI_c and $P^T I_c$ are produced, where I_c is a column matrix that all of its elements are ones. PX is P multiplied by X. PI_c means sum of each row of P, and $P^T I_c$ is sum of each column of P. The final outputs of this portion are these three matrices.

In each run of the algorithm, this portion is called about 120 times. In all calls $M = 7000$, but N may vary in the range between $5,000$ and $20,000$. The computation time of each call in the sequential C++ implementation is about 7 seconds for $N = 20,000$. Our goal is to reduce this computation time to below 0.5 seconds in order to meet clinical requirements.

5.2.2 Parallel Computing Approach

Two different parallel computing approaches are used: (1) GPU acceleration using CUDA programming, and (2) multicore CPU acceleration using shared memory. In CUDA programming, we take advantage of the parallel nature of GPU architectures which have hundreds of cores capable of running thousands of parallel threads, while in the shared memory implementation a multicore CPU architecture is used with 32 cores and at most 32 threads. Our objective is to efficiently parallelize the method in each of these platforms and compare the results.

GPU acceleration using CUDA

CUDA is a parallel computing platform created by NVIDIA. It gives the programmer access to instruction set and memory of the parallel computational elements in NVIDIA GPUs. When programming in CUDA, one can use many thread blocks that are scheduled independently. A thread block contains a number of threads (usually between 32 and 512). A kernel function is run in all threads. The threads in each block are executed in parallel, but multiple blocks may be executed in parallel or one after another depending on the resources of the GPU. Threads of each block can have a fast access to a shared memory and can also get synchronized with a low cost. The number of blocks and number of threads in each block should be adjusted based on the data dependency of the program and available hardware resources to achieve the highest possible performance.

Multicore CPU acceleration using shared memory

Pthreads library is used to parallelize our algorithm on a multicore CPU. This library defines a set of C programming language types, functions and constants for parallelizing tasks over multiple cores of a CPU. This enables us to evaluate the effectiveness of GPU acceleration by comparing its speedup gains against multicore CPU parallelization.

5.2.3 Experiment

In this subsection, we explain how the program is parallelized and how tasks are distributed among the processing elements in each platform.

Parallelization on GPU

The most computationally intensive part of the algorithm is parallelized over the processing elements of a GPU. In this part, $X_{N\times D}$ and $Y_{M\times D}$ are given. We would like to build the probability matrix $P_{M\times N}$ using Eq. 5.3 and derive the outputs PX, PI_c and $P^T I_c$. For building the probability matrix P, we need to allocate an $M \times N$ matrix in the GPU's memory. A kernel function needs to be executed on each processing element for computing the numerator of Eq. 5.3. We assign $M \times N$ threads to execute this kernel function for each m and n, where $0 \leq m < M$ and $0 \leq n < N$. After this step, the numerator matrix will be as follows:

$$numerator_{mn} = exp(\frac{-1}{2}||(Y_m - X_n)/\sigma||^2), \tag{5.5}$$

where each P_{mn} is computed by an independent thread in parallel. Thereafter, we need to compute the denumerator of Eq. 5.3. Note that the denumerator has the same value for all elements of each column of P. Also note that the first term of the denumerator of each column is the sum of the corresponding column in $numerator_{M\times N}$ that we have already computed. Hence, for computing the denumerator value of column i, we need to sum up the values of column i in $numerator_{M\times N}$ and then add the constant ε to the sum. Since $numerator_{M\times N}$ has N columns, this can be done by using n parallel threads. After this step, the vector $denumerator$ with size N is built, where:

$$denumerator_n = (\sum_{k=1}^{M} exp(\frac{-1}{2}||(Y_k - X_n)/\sigma||^2)) + \varepsilon. \tag{5.6}$$

Now we need to again use $M \times N$ threads to divide each element of $numerator_{M\times N}$ by the corresponding element in $denumerator_N$:

$$P_{mn} = \frac{numerator_{mn}}{denumerator_n}. \tag{5.7}$$

Once $P_{M\times N}$ is built, we can obtain PX, PI_c and $P^T I_c$, where PI_c is the vector of the sum of each row of P and $P^T I_c$ is the vector of the sum of each column of P. Based on this description, we point out that PI_c and $P^T I_c$ can be implemented as $PI_{N\times 1}$ and $P^T I_{M\times 1}$ where all elements of $I_{N\times 1}$ and $I_{M\times 1}$ are set to one. Therefore, each of

output matrices PX, PI_c and $P^T I_c$ can be computed with one matrix multiplication. An efficient parallel matrix multiplication procedure is implemented in a library in CUDA called CUBLAS [43]. Matrix multiplication routine of this library is used for computing above-stated matrix products.

Parallelization on a multicore CPU

The computationally intensive part of the algorithm is also implemented using shared memory on a multicore machine. Assume that we use K parallel processing cores, in which K is at most 32, based on available resources. For building the probability matrix $P_{M\times N}$, we distribute N columns of P between K processors. In other words, each processor is assigned to compute N/K columns of P. And the processors work in parallel until all elements of P are computed. Consider column j as one of the columns that is assigned to processor i. This processor traverses over each element of column j and computes the numerator value (Eq. 5.5) of each element. It also sums up the elements while traversing over the column. This way, when it reaches the bottom of the column, all numerator values of the column as well as their summation are computed. We obtain the denumerator of that column (Eq. 5.6) by adding the constant ε to the sum of numerators. At this step, we have both the numerators and the denumerator of the column. Hence, the processor needs to traverse over the column again to divide the numerator of each element by the denumerator of the column. During this traverse, it also sums up the products to obtain the jth element of $P^T I_c$. Now, jth column of $P_{M\times N}$ as well as jth element of vector $P^T I_c$ is computed. Then, processor i moves on to its next assigned column (possibly $j+1$) to repeat these steps. The processors work in parallel until all elements of P and $P^T I_c$ are computed. At this point, a synchronization using a *barrier* is necessary to make sure all processes are done before moving on to the next step. For obtaining PX, we distribute the rows of P among processors to perform a parallelized matrix multiplication. This procedure includes traversing over each row of the first matrix P. During this traverse, we also sum up the elements of the row to obtain each element of PI_c. By the end of this step, when all processors are done we will have all outputs PX, PI_c and $P^T I_c$.

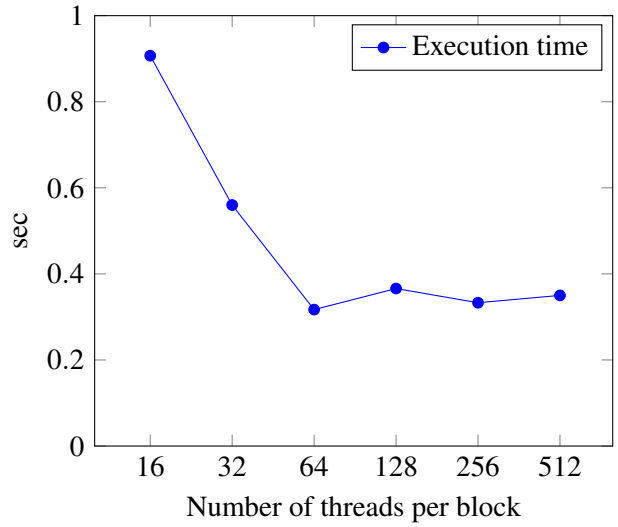

Figure 5.3: Computation time of the CUDA program for different block sizes. M=7,000 and N=20,000.

5.2.4 Results

Block size in GPU programming

Deciding on the optimal number of threads for each block is a challenging step in GPU acceleration. In this experiment, we have changed the block size for the problem size of N=20,000 and reported the observed performance. Results are shown in Figure 5.3. The results illustrate that the best performance is obtained with the block size of 64 threads. Following this observation, all further experiments are performed with the block size of 64 threads.

Speedup gain from GPU-acceleration

The parallelized program is called in each run of the segmentation algorithm about 120 times. In all these calls $M = 7,000$, but N can be varied between $5,000$ and $20,000$. For this reason, we tested our GPU accelerated program for different problem sizes in this range. The results are compared with a sequential efficient pro-

Table 5.1: Computation time and speedup of the GPU-accelerated program for different problem sizes.

Problem size	N=5,000	N=10,000	N=15,000	N=20,000
Sequential program	1400 ms	3700 ms	4900 ms	7200 ms
CUDA program	81 ms	163 ms	241 ms	317 ms
Speedup	17×	22×	20×	22×

gram which has been run on the same machine. Computational times and speedups are reported in Table 5.1. All the timings are obtained from a 2.67 GHz Intel Xeron machine with an NVIDIA Tesla C2050 GPU.

The GPU-accelerated part of the algorithm is integrated in the implementation of the whole statistical model registration which is used in Chapters 2 and 4. The processing time on a sample CT scan is reduced from 15 minutes to 76 seconds. The timing reported for model registration in Chapters 2 and 4 used an approximate implementation of the algorithm. On the same sample CT scan, Our GPU-accelerated method performed 1.45× faster than the approximate implementation. The accuracy of this approach is also not affected by approximation.

Speedup gain from multicore CPU acceleration

For each problem size, we have tested the multicore-accelerated program with different number of cores. Figure 5.4 shows the execution times for the problem size $N = 20,000$ when using different numbers of cores. The results demonstrate that increasing the number of cores up to 8 may improve the performance, but increasing further will reduce the performance due to the overhead of accessing shared memory and synchronization. Table 5.2 shows the execution times and obtained speedups of our shared memory implementation for different problem sizes. Reported results for the sequential program are obtained from using only one core when running the same shared memory implementation. Reported results for the multicore-accelerated program are the best results that are obtained by varying the number of cores for each problem size.

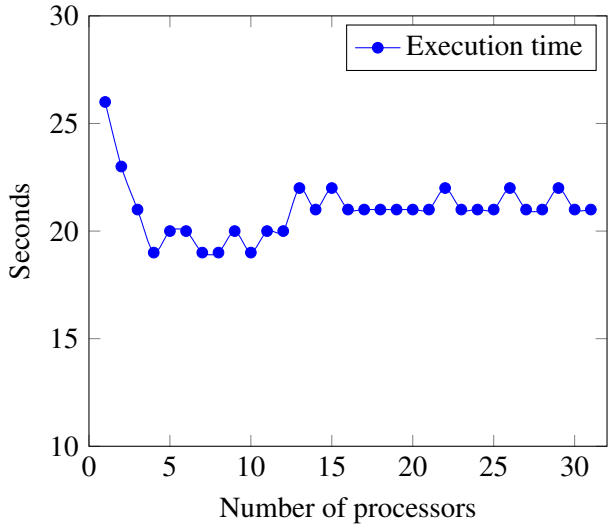

Figure 5.4: Computation time of the multicore-accelerated program for different numbers of processors. M=7,000 and N=20,000.

Table 5.2: Computation time and speedup of the multicore-accelerated program for different problem sizes.

Problem size	N=5,000	N=10,000	N=15,000	N=20,000
Sequential	7 s	14 s	21 s	26 s
Multicore accelerated	5 s	11 s	16 s	19 s
Speedup	1.4×	1.3×	1.3×	1.3×

5.2.5 Discussion

The speedup gain results from the GPU accelerated program (more than 17× in all cases) is much more than the speedup gain from the conventional multicore-accelerated one (less than 1.5× in all cases). This observation shows that the highly parallel structure of GPUs makes them more effective than general-purpose multicore CPUs for our spine segmentation algorithm. This speedup makes the registration algorithm more acceptable for clinical settings.

5.3 Summary

In the first section of this chapter, a vectorized algorithm is proposed for extracting the intensity-based features used in previous chapters, in parallel, from all voxels of a 3D image. Quantitative results showed about $12\times$ speedup over a sequential approach in MATLAB.

In the second section of this chapter, we parallelized our segmentation approach, used in previous chapters, over processing elements of a GPU. The speedup gain from this approach is compared against a conventional multicore-accelerated approach. A speedup around $17\times$ was obtained from the GPU-accelerated method which was significantly higher than the speedup gain from multicore acceleration. This speedup can determine the potential of GPU-acceleration for fitting the proposed automatic segmentation method in the acceptable time-frame of various clinical tasks.

Chapter 6

Conclusion and Future Work

Designing an effective computer-aided system for spine diagnosis and therapy requires a method for automatic vertebra localization and identification. An automatic segmentation method can also directly benefit such system for diagnosing spine pathologies. To be accepted in clinical practice, these methods need to be fast, accurate, and robust.

In this thesis, we proposed and evaluated several methods for vertebra localization, identification, and segmentation. In Chapter 2, we successfully adapted a semi-automatic segmentation approach, which was previously proposed for CT scans, to volumetric MR images. We showed that this method can handle intrinsic MR segmentation challenges such as intensity inhomogeneities and large inter-slice gaps. In Chapter 3, we introduced a method for automatic localization, and labeling of vertebrae in three-dimensional CT scans. An extensive evaluation showed that this method can efficiently solve the localization problem in general scans where the field of view is restricted and spine pathologies may be present. The methods from Chapters 2 and 3 are combined in Chapter 4 to build an integrated automatic system for all tasks of vertebra localization, labeling, and segmentation. Evaluation on MR images of the lumbar region shows promising performance for the integrated system. In Chapter 5, the potential of some parallel computing approaches is evaluated for improving the performance of the methods presented in previous chapters.

6.1 Contributions

The major contributions of this thesis are summarized as follows:

- A semi-automatic method is developed for extracting the edges of vertebral bodies in volumetric MR images of the spine. A bias-field correction algorithm is applied to reduce the intensity inhomogeneities. Then, a simple user interaction is used for determining the Region Of Interest (ROI). A 3D anisotropic diffusion filter is applied to the ROI for smoothing the image while preserving the edges. Then, a Canny edge detector is used to extract the desired edges. The edge map is then used for statistical model registration in order to segment the vertebral bodies.

- A novel technique is developed for automatic vertebra localization and identification based on training deep neural networks. Separate estimators are used for different coordinates. The voxels of the image that are far from Canny edges are disregarded in the estimation. A recently proposed Kernel Density Estimation [6] method is used for fast and robust vote aggregation.

- A novel localization refinement approach is proposed and integrated in the automatic localization system. The initial prediction for location of a vertebra is obtained from the votes of voxels from all over the image. At this point, the prediction is refined by aggregating the votes of the voxels around the initial prediction. The votes that are far from the initial prediction are considered as outliers and disregarded. This additional step improves the results of the method on a large dataset of pathological cases, where unpredicted curvatures may be present in different parts of the spine, while it does not add significant computation time to the localization system.

- A fully-automatic system is developed by integrating an automatic vertebrae localization algorithm and a semi-automatic segmentation approach. A novel technique, based on local thresholding, is developed for refining the vertebrae localization predictions in MR images. This technique is used as an intermediate step for integrating the above-stated systems.

- The registration of a multi-vertebrae statistical model of the spine is parallelized over the computing elements of a Graphical Processing Unit (GPU). The computational time is compared against the pre-existing sequential implementation.

- A vectorization technique is developed for computing the mean intensity of hundreds of displaced cuboids for all pixels of a 3D image in parallel. These values are used as features for building an automatic vertebrae localization system for CT scans and MR images.

6.2 Future Work

As a continuation of the work presented in this thesis, a number of interesting research projects can be suggested as follows:

- In Chapters 2 and 4, we focused on segmenting the vertebral body part of each vertebra in MR images. Because of the large slice spacing in typical MR images of the spine, the other parts such as transverse processes are not depicted as well as vertebral bodies. However, a sub-anatomical labeling of vertebral arch may be possible by registering the statistical model of the whole vertebrae after convergence of the vertebral body parts.

- The segmentation methods presented in Chapters 2 and 4 are evaluated on a small dataset of MR images. In addition, the ground truth meshes are provided by a single non-expert. Future work may involve better evaluation of the method by applying it to a larger dataset that are manually segmented by multiple experts. Since the manual segmentation task on MR images is a subjective task, considering the variability of manually segmented meshes provided by multiple experts can significantly increase the reliability of the reported segmentation results.

- The vertebrae localization approach presented in Chapter 3 is extensively evaluated on a large dataset of arbitrary-field-of-view CT scans that most of them have spine pathologies. However, the evaluation on MR images has

been done on a small set of images that are all captured from lumbar region. Future work may involve a more extensive evaluation on MR images.

- In the vertebrae localization approach proposed in Chapter 3, we used hundreds of intensity-based features to feed deep neural networks. A future work may involve including the feature extraction step in the deep network, using convolutional neural networks or Boltzmann machines.

- The vertebrae localization approach proposed in Chapter 3, does not rely on the image modality or the nature of the spine anatomy. Future work may involve using this method for localizing other anatomies like prostate or liver in different image modalities such as CT scans, MR images, and ultrasound images.

Bibliography

[1] R. S. Alomari, J. J. Corso, and V. Chaudhary. Labeling of lumbar discs using both pixel-and object-level features with a two-level probabilistic model. *Medical Imaging, IEEE Transactions on*, 30(1):1–10, 2011. → pages 3, 16, 50

[2] P. Baldi and K. Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2(1): 53–58, 1989. → pages 33

[3] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends® in Machine Learning*, 2(1):1–127, 2009. → pages 17

[4] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. *Advances in Neural Information Processing Systems*, 19:153–168, 2007. → pages 26

[5] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, 2010. → pages 47, 56

[6] Z. Botev, J. Grotowski, D. Kroese, et al. Kernel density estimation via diffusion. *The Annals of Statistics*, 38(5):2916–2957, 2010. → pages 30, 39, 52, 76

[7] J. Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698, 1986. → pages 9

[8] J. Carballido-Gamio, S. J. Belongie, and S. Majumdar. Normalized cuts in 3-D for spinal MRI segmentation. *Medical Imaging, IEEE Transactions on*, 23(1):36–44, 2004. → pages 6, 9

[9] A. Criminisi, J. Shotton, and S. Bucciarelli. Decision forests with long-range spatial context for organ localization in CT volumes. *Proc MICCAI Workshop on Probabilistic Models for Medical Image Analysis*, pages 69–80, 2009. → pages 19

[10] A. Criminisi, D. Robertson, O. Pauly, B. Glocker, E. Konukoglu, J. Shotton, D. Mateus, A. M. Möller, S. Nekolla, and N. Navab. Anatomy detection and localization in 3D medical images. In *Decision Forests for Computer Vision and Medical Image Analysis*, pages 193–209. Springer, 2013. → pages xi, xiii, 19, 20, 52, 53

[11] G. E. Dahl, T. N. Sainath, and G. E. Hinton. Improving deep neural networks for LVCSR using rectified linear units and dropout. In *Acoustics, Speech and Signal Processing, IEEE International Conference on*, pages 8609–8613. IEEE, 2013. → pages 34, 36

[12] J. Egger, T. Kapur, T. Dukatz, M. Kolodziej, D. Zukić, B. Freisleben, and C. Nimsky. Square-cut: a segmentation algorithm on the basis of a rectangle shape. *PloS One*, 7(2):e31064, 2012. → pages 2, 6, 9

[13] A. Field. *Discovering statistics using SPSS*. Sage Publications Limited, 2009. → pages 66

[14] J. Gall and V. Lempitsky. Class-specific hough forests for object detection. In *Decision Forests for Computer Vision and Medical Image Analysis*, pages 143–157. Springer, 2013. → pages 19

[15] B. Glocker, J. Feulner, A. Criminisi, D. R. Haynor, and E. Konukoglu. Automatic localization and identification of vertebrae in arbitrary field-of-view CT scans. In *Medical Image Computing and Computer-Assisted Intervention*, pages 590–598. Springer, 2012. → pages viii, 16, 41, 45, 46, 47, 50

[16] B. Glocker, D. Zikic, E. Konukoglu, D. R. Haynor, and A. Criminisi. Vertebrae localization in pathological spine ct via dense classification from sparse annotations. In *Medical Image Computing and Computer-Assisted Intervention*, pages 262–270. Springer, 2013. → pages ix, 16, 17, 40, 41, 45, 46, 47, 48

[17] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010. → pages 26

[18] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, volume 15, pages 315–323, 2011. → pages 36

[19] P. Hall, T. C. Hu, and J. S. Marron. Improved variable window kernel estimates of probability densities. *The Annals of Statistics*, pages 1–10, 1995. → pages 30

[20] S. Hanaoka, K. Fritscher, B. Schuler, Y. Masutani, N. Hayashi, K. Ohtomo, and R. Schubert. Whole vertebral bone segmentation method with a statistical intensity-shape model based approach. In *SPIE Medical Imaging*, pages 796242–796242. International Society for Optics and Photonics, 2011. → pages 16, 50

[21] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012. → pages 34

[22] C. Hoad and A. Martel. Segmentation of MR images for computer-assisted surgery of the lumbar spine. *Physics in Medicine and Biology*, 47(19):3503, 2002. → pages 6

[23] S.-H. Huang, Y.-H. Chu, S.-H. Lai, and C. L. Novak. Learning-based vertebra detection and iterative normalized-cut segmentation for spinal MRI. *Medical Imaging, IEEE Transactions on*, 28(10):1595–1605, 2009. → pages 6, 9

[24] I. Jolliffe. *Principal component analysis*. Wiley Online Library, 2005. → pages 33

[25] C. Jones, J. Marron, and S. Sheather. Progress in data-based bandwidth selection for kernel density estimation. *Computational Statistics*, (11): 337–381, 1996. → pages 30

[26] M. Jones, I. McKay, and T.-C. Hu. Variable location and scale kernel density estimation. *Annals of the Institute of Statistical Mathematics*, 46(3): 521–535, 1994. → pages 30

[27] M. C. Jones, J. S. Marron, and S. J. Sheather. A brief survey of bandwidth selection for density estimation. *Journal of the American Statistical Association*, 91(433):401–407, 1996. → pages 30

[28] S. Kadoury, H. Labelle, and N. Paragios. Spine segmentation in medical images using manifold embeddings and higher-order MRFs. *Medical Imaging, IEEE Transactions on*, 32(7):1227–1238, 2013. → pages 6

[29] S. Khallaghi, P. Abolmaesumi, R. H. Gong, E. Chen, S. Gill, J. Boisvert, D. Pichora, D. Borschneck, G. Fichtinger, and P. Mousavi. GPU accelerated registration of a statistical shape model of the lumbar spine to 3D ultrasound images. In *SPIE Medical Imaging*, pages 79642W–79642W. International Society for Optics and Photonics, 2011. → pages 66

[30] K. I. Kim, K. Jung, and H. J. Kim. Face recognition using kernel principal component analysis. *Signal Processing Letters, IEEE*, 9(2):40–42, 2002. → pages 33

[31] K. C. Kiwiel. Convergence and efficiency of subgradient methods for quasiconvex minimization. *Mathematical Programming*, 90(1):1–25, 2001. → pages 37

[32] T. Klinder, J. Ostermann, M. Ehm, A. Franz, R. Kneser, and C. Lorenz. Automated model-based vertebra detection, identification, and segmentation in CT images. *Medical Image Analysis*, 13(3):471–482, 2009. → pages 16, 17, 50

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012. → pages 22

[34] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural Networks: Tricks of The Trade*, pages 9–48. Springer, 2012. → pages 26, 36

[35] H. Lee, P. Pham, Y. Largman, and A. Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in Neural Information Processing Systems*, pages 1096–1104, 2009. → pages 33

[36] J. Ma, L. Lu, Y. Zhan, X. Zhou, M. Salganicoff, and A. Krishnan. Hierarchical segmentation and identification of thoracic vertebra using learning-based edge detection and coarse-to-fine deformable model. In *Medical Image Computing and Computer-Assisted Intervention*, pages 19–27. Springer, 2010. → pages 16, 50

[37] J. Martens. Deep learning via hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning*, pages 735–742, 2010. → pages 37

[38] A. Myronenko and X. Song. Point set registration: coherent point drift. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(12): 2262–2275, 2010. → pages 67

[39] B. Naegel. Using mathematical morphology for the anatomical labeling of vertebrae from 3D CT-scan images. *Computerized Medical Imaging and Graphics*, 31(3):141–156, 2007. → pages 16, 50

[40] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*, pages 807–814, 2010. → pages 36

[41] A. Neubert, J. Fripp, C. Engstrom, R. Schwarz, L. Lauer, O. Salvado, and S. Crozier. Automated detection, 3D segmentation and analysis of high resolution spine MR images using statistical shape models. *Physics in Medicine and Biology*, 57(24):8357–8376, 2012. → pages 6

[42] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng. Multimodal deep learning. In *Proceedings of the 28th International Conference on Machine Learning*, pages 689–696, 2011. → pages 33

[43] C. Nvidia. Cublas library. *NVIDIA Corporation, Santa Clara, California*, 15, 2008. → pages 70

[44] A. B. Oktay and Y. S. Akgul. Simultaneous localization of lumbar vertebrae and intervertebral discs with SVM-based MRF. *Biomedical Engineering, IEEE Transactions on*, 60(9):2375–2383, 2013. → pages 16, 50

[45] Y. Otake, M. Armand, R. S. Armiger, M. D. Kutzer, E. Basafa, P. Kazanzides, and R. H. Taylor. Intraoperative image-based multiview 2D/3D registration for image-guided orthopaedic surgery: incorporation of fiducial-based C-arm tracking and GPU-acceleration. *Medical Imaging, IEEE Transactions on*, 31(4):948–962, 2012. → pages 66

[46] N. Otsu. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975. → pages 52

[47] B. Park, S.-O. Jeong, M. Jones, and K.-H. Kang. Adaptive variable location kernel density estimators with good performance at boundaries. *Journal of Nonparametric Statistics*, 15(1):61–75, 2003. → pages 30

[48] Z. Peng, J. Zhong, W. Wee, and J.-h. Lee. Automated vertebra detection and segmentation from the whole spine MR images. In *Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the*, pages 2527–2530. IEEE, 2006. → pages 6, 16, 50

[49] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(7):629–639, 1990. → pages 8, 54

[50] A. Rasoulian, R. Rohling, and P. Abolmaesumi. Lumbar spine segmentation using a statistical multi-vertebrae anatomical shape+pose model. *Medical Imaging, IEEE Transactions on*, 32(10):1890–1900, 2013. → pages 6, 7, 9, 18, 44, 50, 54, 66

[51] A. Rasoulian, R. Rohling, and P. Abolmaesumi. A statistical multi-vertebrae shape+pose model for segmentation of CT images. In *SPIE Medical Imaging*, volume 8671, 2013. → pages 50

[52] A. Rasoulian, R. N. Rohling, and P. Abolmaesumi. A statistical multi-vertebrae shape+ pose model for segmentation of CT images. In *SPIE Medical Imaging*, pages 86710P–86710P. International Society for Optics and Photonics, 2013. → pages 7

[53] A. Rasoulian, R. N. Rohling, and P. Abolmaesumi. Automatic labeling and segmentation of vertebrae in ct images. In *SPIE Medical Imaging*, pages 903623–903623. International Society for Optics and Photonics, 2014. → pages 16, 17, 50

[54] L. Remonda, A. Lukes, and G. Schroth. [spinal stenosis: current aspects of imaging diagnosis and therapy]. *Schweizerische Medizinische Wochenschrift*, 126(6):220–229, 1996. → pages 2

[55] P. J. Richards, J. George, M. Metelko, and M. Brown. Spine computed tomography doses and cancer induction. *Spine*, 35(4):430–433, 2010. → pages 2

[56] T. D. Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 2(6):459–473, 1989. → pages 33

[57] J. Schmidhuber. Deep learning in neural networks: An overview. *arXiv preprint arXiv:1404.7828*, 2014. → pages 17

[58] S. Schmidt, J. Kappes, M. Bergtholdt, V. Pekar, S. Dries, D. Bystrov, and C. Schnörr. Spine detection and labeling using a parts-based graphical model. In *Information Processing in Medical Imaging*, pages 122–133. Springer, 2007. → pages 16, 50

[59] S. J. Sheather and M. C. Jones. A reliable data-based bandwidth selection method for kernel density estimation. *Journal of the Royal Statistical Society*, pages 683–690, 1991. → pages 30

[60] R. Shi, D. Sun, Z. Qiu, and K. L. Weiss. An efficient method for segmentation of MRI spine images. In *Complex Medical Engineering, IEEE/ICME International Conference on*, pages 713–717. IEEE, 2007. → pages 6, 9

[61] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *International Journal of Computer Vision*, 81(1):2–23, 2009. → pages 19

[62] P. Shwaluk. Clinical anatomy and management of low back pain. *Australasian Chiropractic & Osteopathy*, 6(1):24, 1997. → pages 1

[63] P. P. Smyth, C. J. Taylor, and J. E. Adams. Automatic measurement of vertebral shape using active shape models. *Image and Vision Computing*, 15 (8):575–581, 1997. → pages 16

[64] D. Štern, B. Likar, F. Pernuš, and T. Vrtovec. Parametric modelling and segmentation of vertebral bodies in 3D CT and MR spine images. *Physics in Medicine and Biology*, 56(23):7505, 2011. → pages 6

[65] D. Štern, T. Vrtovec, F. Pernuš, and B. Likar. Segmentation of vertebral bodies in CT and MR images based on 3D deterministic models. In *SPIE Medical Imaging*, pages 79620D–79620D. International Society for Optics and Photonics, 2011. → pages 6

[66] A. Suzani, A. Rasoulian, S. Fels, R. N. Rohling, and P. Abolmaesumi. Semi-automatic segmentation of vertebral bodies in volumetric MR images using a statistical shape+ pose model. In *SPIE Medical Imaging*, pages 90360P–90360P. International Society for Optics and Photonics, 2014. → pages 50

[67] C. Szegedy, A. Toshev, and D. Erhan. Deep neural networks for object detection. In *Advances in Neural Information Processing Systems*, pages 2553–2561, 2013. → pages 22

[68] G. R. Terrell and D. W. Scott. Variable kernel density estimation. *The Annals of Statistics*, pages 1236–1265, 1992. → pages 30

[69] P. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004. → pages 19, 52, 62

[70] P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990. → pages 26, 37

[71] P. L. Williams, M. Dyson, L. Bannister, P. Collins, M. Berry, M. Ferguson, and J. Dussek. Gray's anatomy: The anatomical basis of medicine and surgery. 1995. → pages 1

[72] J. Yao, J. E. Burns, H. Munoz, and R. M. Summers. Detection of vertebral body fractures based on cortical shell unwrapping. In *Medical Image Computing and Computer-Assisted Intervention*, pages 509–516. Springer, 2012. → pages 7

[73] K. Yu, W. Xu, and Y. Gong. Deep learning with kernel regularization for visual recognition. In *Advances in Neural Information Processing Systems*, pages 1889–1896, 2009. → pages 34

[74] P. A. Yushkevich, J. Piven, H. Cody Hazlett, R. Gimpel Smith, S. Ho, J. C. Gee, and G. Gerig. User-guided 3D active contour segmentation of anatomical structures: Significantly improved efficiency and reliability. *Neuroimage*, 31(3):1116–1128, 2006. → pages 7, 18, 50

[75] G. Zamora, H. Sari-Sarraf, and L. R. Long. Hierarchical segmentation of vertebrae from x-ray images. In *Medical Imaging 2003*, pages 631–642. International Society for Optics and Photonics, 2003. → pages 16

[76] M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, et al. On rectified linear units for speech processing. In *Acoustics, Speech and Signal Processing, IEEE International Conference on*, pages 3517–3521. IEEE, 2013. → pages 36

[77] J. Zhang, L. Lv, X. Shi, F. Guo, Y. Zhang, and H. Li. Hough transform-based approach for estimating 3D rotation angles of vertebrae from biplanar radiographs using GPU-acceleration. *International Journal of Imaging Systems and Technology*, 23(3):272–279, 2013. → pages 66

[78] D. Zukic, A. Vlasák, T. Dukatz, J. Egger, D. Horínek, C. Nimsky, and A. Kolb. Segmentation of vertebral bodies in MR images. In *Vision,*

Modeling & Visualization, pages 135–142. The Eurographics Association, 2012. → pages 6