Spring 2016

Deformable Contour Models for Digitizing a Printed Brainstem Atlas

Nirmal J. Patel
Old Dominion University

DEFORMABLE CONTOUR MODELS FOR DIGITIZING A PRINTED BRAINSTEM ATLAS

by

Nirmal J. Patel
B.S. May 2014, Old Dominion University

A Thesis Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

ELECTRICAL AND COMPUTER ENGINEERING

OLD DOMINION UNIVERSITY
May 2016

Approved by:

Dean J. Krusienski (Director)

Michel A. Audette (Co-Director)

Jiang Li (Member)

ABSTRACT

DEFORMABLE CONTOUR MODELS FOR DIGITIZING A PRINTED BRAINSTEM ATLAS

Nirmal J. Patel
Old Dominion University, 2016
Co-Directors: Dr. Dean J. Krusienski
Dr. Michel A. Audette

The brainstem is a part of the brain that is connected to the cerebrum and the spinal cord. Ten out of twelve pairs of cranial nerves emerge from the brainstem. The cranial nerves transmit information between the brain and various parts of the body. Due to its anatomical and physiological relevance, a descriptive digital brainstem is important for neurosurgery planning and simulation. For both of these neurosurgical applications, the complexity of the brainstem requires a digital atlas approach to segmentation that maps intensities to tissues rather than less descriptive voxel or surface-based approaches. However, a descriptive brainstem atlas with adequate details for neurosurgery planning and simulation has not been developed to date. Fortunately, various textbooks contain 2D representations of the brainstem at various longitudinal coordinates. The aim of this thesis is to describe a minimally supervised method to segment sketches coinciding with slices of the brainstem featuring labeled contours. This thesis also describes a deformable contour model approach, emphasizing a 1-simplex framework, to reconstruct a 3D volume from 2D slices.

ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Michel A. Audette who has been advising me since my undergraduate years. I would like to thank my co-advisor Dr. Dean J. Krusienski for his help and guidance. I am most grateful to both Dr. Audette and Dr. Krusienski for their moral support. I would like to thank Dr. Jiang Li for his participation in my thesis defense committee. I would like to also acknowledge Sharmin Sultana and Tahsin Khajah for their help. Finally, I am grateful to my brother and my parents.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

The brainstem is a part of the brain that is connected to the cerebrum and the spinal cord. The brainstem is structurally continuous with the spinal cord. Figure 1 shows the coarse anatomy of the brainstem. The brainstem is the pathway for fiber tracts passing up and down from peripheral nerves and the spinal cord to the upper areas of the brain. It controls various bodily functions such as swallowing, heart rate, and breathing and it transmits information between the brain and various parts of the head and neck using cranial nerves as shown in Figure 2. It is the point of attachment of ten of the twelve pairs of cranial nerves. It consists of three components: the midbrain, the pons, and the medulla oblongata.

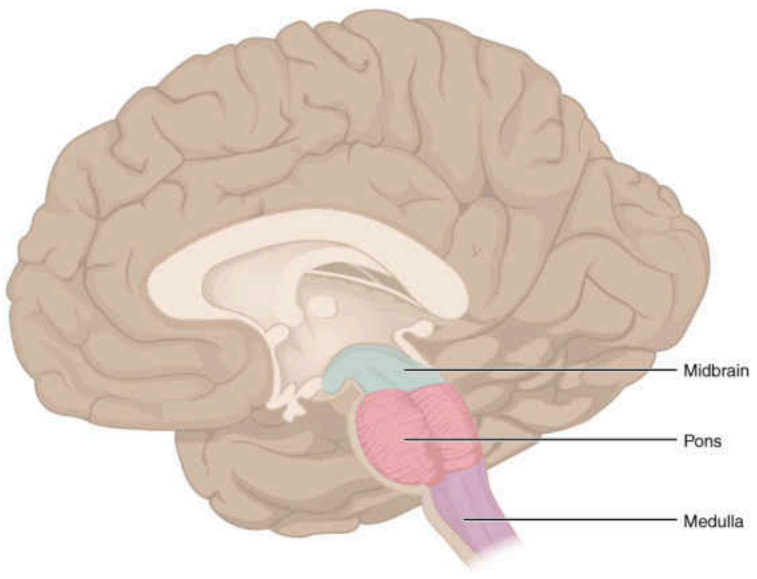


FIG. 1: The coarse anatomy of the brain showing the midbrain, the pons, and the medulla oblongata [7].

The midbrain is important for auditory and visual processing as well as ocular motor control. In the midbrain, there are crus cerebri, which are tracts made of neurons that connect the cerebral hemespheres to the cerebellum. The pons is involved in auditory and visual sensory analysis. For instance, the auditory information first arrives in the brain in pons. Furthermore, the pons transmits signals between the medulla oblongata and the higher cortical structures of the brain. The medulla oblongata is important for regulating autonomic activities such as respiration and heart rate. It also controls swallowing and vomiting. Since the brainstem is involved in many vital bodily functions, it is an important component of the neuroanatomy.

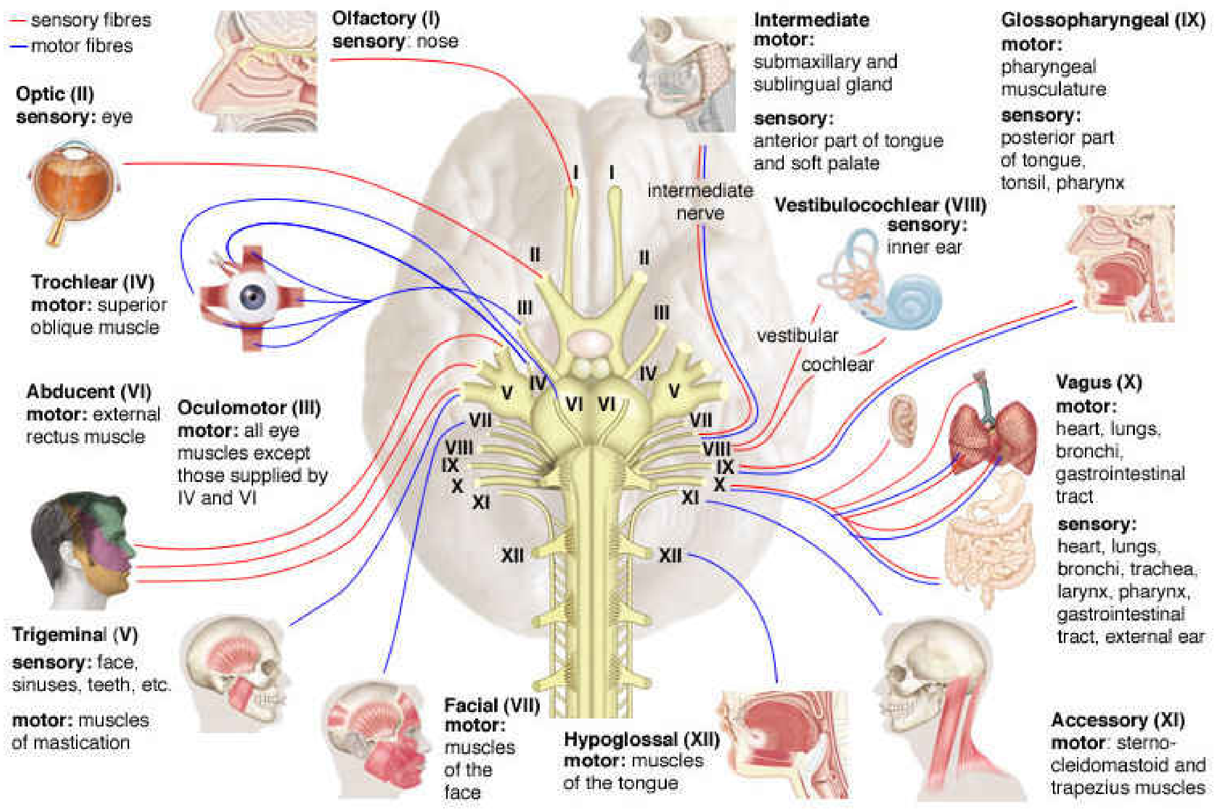

— sensory fibres
— motor fibres
Olfactory (I)
sensory: nose
Optic (II)
sensory: eye
Intermediate
motor: submaxillary and sublingual gland
sensory: anterior part of tongue and soft palate
Glossopharyngeal (IX)
motor: pharyngeal musculature
sensory: posterior part of tongue, tonsil, pharynx
intermediate nerve
Vestibulocochlear (VIII)
sensory: inner ear
vestibular
cochlear
Trochlear (IV)
motor: superior oblique muscle
Abducent (VI)
motor: external rectus muscle
Oculomotor (III)
motor: all eye muscles except those supplied by IV and VI
Vagus (X)
motor: heart, lungs, bronchi, gastrointestinal tract
sensory: heart, lungs, bronchi, trachea, larynx, pharynx, gastrointestinal tract, external ear
Trigeminal (V)
sensory: face, sinuses, teeth, etc.
motor: muscles of mastication
Facial (VII)
motor: muscles of the face
Hypoglossal (XII)
motor: muscles of the tongue
Accessory (XI)
motor: sterno-cleidomastoid and trapezius muscles
I I II II III III IV IV V V VI VI VII VII VIII VIII IX IX X X XI XI XII XII

FIG. 2: Cranial nerves and their attachment points. Cranial nerve III to XII emerge from the brainstem [8].

Although the brainstem is an invaluable neuroanatomical structure, there is a lack of digital 3D atlases of the brainstem featuring high resolution data, which can be used for patient-specific brainstem modeling. A deformable atlas is required for surgery planning and simulation. Although there are a number of digital atlases of the brain, these atlases neglect the brainstem, with little detail provided on it. The brain atlases include Harvard's Surgical Planning Laboratory (SPL) atlas [14], the atlas available with Oxford's FMRIB Software Library (FSL) [31], and the Freesurfer software package [11]. None of these atlases represents the brainstem. The only competing atlas is the Mai online atlas [20], but its brainstem is not as descriptive as the atlases found in textbooks such as the Duvernoy's atlas [22] and the Paxinos' atlas [21].

1.2 MOTIVATION

There have been several complications because of iatrogenic damage in the skull base and spine by surgery or radiotherapy [3, 33, 6, 27, 2]. Some of these complications could have been mitigated using surgery planning and simulation tools which, using patient-specific brainstem models, could be used in simulation to penalize gestures causing damage to cranial nerves as well as preventing intraoperative gestures deleterious to these structures. Thus, it can be strongly argued that improved surgery planning for experts and surgery simulation for residents would improve patient outcome in terms of morbidity and mortality statistics. Iatrogenic complications in neuro, head, and neck surgery include the following cases:

- The spinal accessory nerve (SAN) is susceptible to injury in head and neck surgery. Estimates of SAN injury incidence in diagnostic lymph node biopsies of the posterior triangle of the neck are 3-8 % [3, 33].
- The optical and oculomotor nerves are susceptible to injury in pituitary surgery. In a national survey [6] to which 1162 neurosurgeons responded, 939 reported having witnessed at least one complication. Of the 939 neurosurgeons, 179 reported post-operative visual loss in at least one patient.
- A case of the cranial nerve injury in an acoustic neuroma patient was documented [27]. Her bone-embedded tumor was removed using an ultrasonic surgical aspirator. The patient suffered right-sided palsies of the 5^{th}, 6^{th}, 7^{th}, and 12^{th} cranial nerves.
- Surgical complications during skull base surgery in the posterior fossa can introduce neurological dysfunction resulting in a swallowing disorder [2].

Thus, better surgery planning and simulation tools capable of producing patient-specific models are needed. In order to improve patient outcome, surgery planning and simulation must model cranial and cervical nerves explicitly. To explicitly model these nerves, it is important to exploit both shape priors and tractographic reconstruction of the nerves using diffusion tensor imaging. However, tractographic reconstruction of cranial and spinal nerves is still in its infancy compared to the tractography of the brain. Thus, it is vital to develop a digital atlas for the nerves' most prevalent topology in order to produce reliable shape priors.

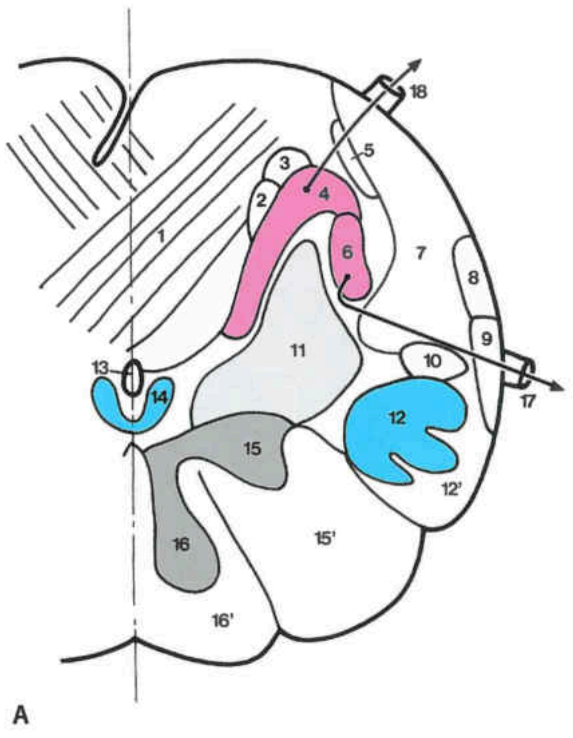
In order to generate patient-specific models for neurosurgery planning and simulation, medical data obtained using magnetic resonance imaging (MRI) or other means must be segmented. This segmentation of medical images entails determining correspondence between each voxel in the medical image and the functional regions. In particular, the segmentation involves assigning each voxel to an anatomical structure. There are three categories of image segmentation techniques, with some overlap possible [13]:

Voxel-based methods rely on the intensity information in medical images. In particular, they use voxel intensity and local information in order to determine to which structure the voxel belongs. The voxel is then assigned the label of the structure. Unfortunately, there are a lot of ambiguities in voxel-based methods because there is no one-to-one relationship between voxel intensities and tissue labels.

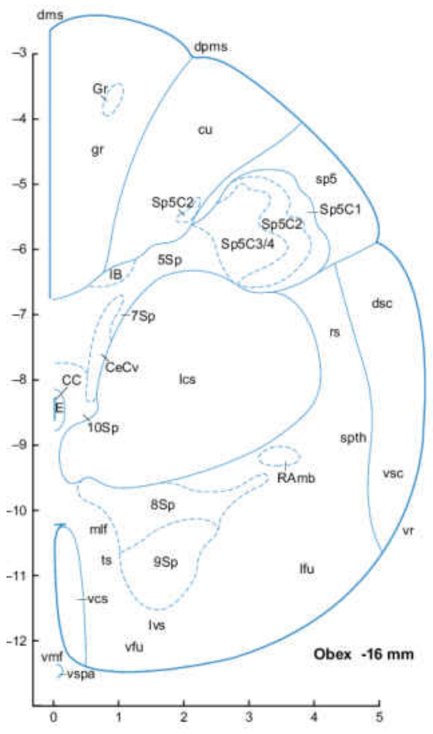
Boundary-based methods locate boundaries of tissue structures rather than assigning a label to each voxel. These methods typically evolve a contour in 2D or a surface in 3D, starting from a user-defined initialization, until it reaches a boundary. A boundary is typically considered a portion of the image with a high gradient in the direction coinciding with the interface between two tissues. The contour or surface evolution can also be constrained with a shape prior.

Atlas-based methods segment a medical image by deforming an already segmented high-resolution source image to a target image to find correspondence between the two images. This approach requires an expertly segmented source image.

Although the atlas-based approach requires a manually segmented and verified model, it is capable of providing descriptiveness far richer than the other two methods. If the deformation finishes successfully, everything in the target image such as tissue labels is essentially understood in terms of the manually segmented model. Furthermore, a region of interest in an image segmented using an atlas-based approach can be used as



(a)



(b)

FIG. 3: Examples of 2D sketches of the brainstem found (a) in Duvernoy's [22] textbook and (b) in Paxinos' [21] textbook.

a shape prior with another method for refinement. Therefore, the atlas-based approach is often clinically more relevant for complex anatomies. In contrast with whole-brain digital atlases, there is a lack of digital descriptive brainstem atlases required for atlas-based segmentation.

Despite the paucity of digital brainstem atlases, there are atlases in printed form, found in textbooks. These atlases are represented as slices of the brainstem along a longitudinal axis. Figure 3 shows examples of brainstem atlases found in two textbooks [21, 22]. These atlases are rigorous and descriptive. They can be scanned, digitized, stacked, and reconstructed to generate a digital 3D atlas of the brainstem.

1.3 OBJECTIVE AND CONTRIBUTIONS

The objective of this thesis is to create a software tool which, with minimal supervision, digitizes the atlases of the brainstem slices found in textbooks. Once these 2D atlases are digitized, a novel methodology to reconstruct a region in 3D from 2D slices is also presented. The overall goal of these methods is to facilitate the generation of a brainstem atlas that is suitable for the atlas-based registration. Once the brainstem model is complete, it will fulfill the requirement of an anatomically detailed brainstem atlas for neurological surgery planning and simulation.

An anatomist was consulted for feedback on the thesis research. He felt that the more important functional structures should receive priority. In particular, digitizing nerve centers is more important than focusing on the entire brainstem. Thus, the goal of the thesis is not the digital reconstruction of the entire brainstem.

The first contribution of this thesis is a methodology to semi-automatically digitize the brainstem atlases found in textbooks. The second contribution of this thesis is a novel approach to reconstructing a 3D volume using 2D slices. In particular, the reconstruction methodology uses a novel 2D adaptation of a previously published 3D contour model called 1-simplex mesh [30]. This thesis provides an approach to address the lack of a digital brainstem atlas. This digital atlas will be suitable for producing patient-specific brainstem atlases and modeling cranial nerves using probabilistic shape priors. Therefore, this thesis is an important step towards enhancing the surgery planning and simulation tools.

CHAPTER 2

A BACKGROUND IN SEGMENTATION

Segmentation is vital for many areas of medical image analysis. Segmentation, in medical image analysis, is a process where a medical image is divided into multiple parts coinciding with different tissue structures. It is used to delineate anatomical structures automatically or semi-automatically in images produced by MRI, computed tomography (CT), and other modalities. These imaging modalities are noninvasive, so they are important for studying anatomy and planning treatments. Due to its tissue identification capabilities, segmentation can aid life-saving procedures such as surgery planning, surgery simulation, and diagnosis. Segmentation techniques save clinicians time that can be spent on other areas.

The fundamental component of medical images is a pixel in 2D or a voxel in 3D. Each pixel or voxel has an intensity. As discussed earlier, the three broad categories of segmentation are voxel-based, boundary-based, and atlas-based segmentation. Voxel-based methods use intensity information to determine the tissue represented by a specific voxel. Each voxel is assigned a label corresponding to the functional structure to which it belongs. These methods often use local intensity information such as Hounsfield value in CT and T1-weighted signal in MRI [16]. Medical images may have unclear boundaries, noise, and low resolution. Thus, using only voxel intensities often cannot yield satisfactory results. Voxel-based methods typically use classifiers for segmentation. These classifiers are trained with training data that are manually identified. One of the simplest classifiers is the *k*-nearest-neighbors classifier where a voxel is classified by finding the *k* intensities in the training data that are closest to the intensity of the voxel. The voxel is assigned the most common label in the *k* closest intensities [25]. A number of other voxel-based methods are available as well, including Expectation-Maximization and Fuzzy C-Means [23].

Boundary-based methods delineate anatomical structures using tissue borders. In these methods, a curve in 2D or a surface in 3D is initialized manually. Ideally, the initialization is close to the region of interest. The model is then deformed until it reaches a boundary. A model deformation is generally described as a model with a force that regulates or constrains the shape of the model while the image provides a force that attracts the model. The force exerted by the image attracts the model towards a boundary. The deformation stops once both forces are in equilibrium, which usually occurs at boundaries. In medical images, the general morphological properties of a structure are known. These properties can be incorporated as priors in boundary-based segmentation. Using priors alleviates some of the problems caused by low-contrast boundaries.

There are two deformable contour-based models mentioned in this paper. These models are level set methods [29], which are continuous, and the 1 simplex-mesh method [10], which is discrete. The level set framework is a framework that evolves curves in 2D and surfaces in 3D. It is a continuous framework that is often used to track and model object boundaries. Let $\mathcal{C}$ be a curve on a two dimensional image. Each point in $\mathcal{C}$ moves according to a certain speed until it reaches a boundary in the image. This behavior is emulated by the level set framework using partial differential equations (PDE). One of the strengths of the level set methods is that it allows curves to split and join. Furthermore, it is an adaptable framework featuring multiple deformation schemes. The 1-simplex mesh is another contour based model. An n-simplex is an (n+1)-connected mesh. A 1-simplex mesh is a contour with each vertex connected to 2 edges. Similarly, a 2-simplex mesh is a surface with each vertex connected to 3 edges. As opposed to the level set methods, the 1-simplex model is discrete and is a physically based model. Furthermore, a 1-simplex mesh does not have to be a closed curve. For example, it can model cranial nerves' paths, which are not closed shapes. For this thesis research, contour models are used to detect region boundaries in a 2D scanned atlas. Therefore, surface models are not relevant. A more detailed explanation of these contour models is provided later.

The atlas-based approach requires a source image or model that has been segmented by an expert. In order to segment a target image, a one-to-one correspondence between the source and the target image is determined. The process of finding this transformation, on the basis of correspondences, is called registration. The atlas-based segmentation can produce a result with rich functional details that may not be possible with the other two methods. Furthermore, atlas-based segmentation is capable of segmenting images with ill-defined borders or functionally complex anatomy such as that of the brainstem. A limitation of the atlas-based approach is that a non-rigid registration can be complex and time-consuming for complex anatomical structures.

Image registration geometrically aligns two or more images by finding an appropriate transformation that maps a source image to a target image. Due to anatomical variations among subjects, a non-rigid registration technique must be used for the atlas-based approach. In a non-rigid registration, the transformation must allow local warping of the source image for alignment. Rigid registration, which only allows rotation or translation, does not have enough flexibility for medical segmentation. A registration algorithm has three main components [15]:

- the similarity criterion that quantifies the correspondences between the source and the target images;
- the transformation parameters that describe how the source image can be morphed to match the target image;
- the optimization process that maximizes the similarity criterion by searching for the transformation parameters that minimize an objective function based on the similarity criterion.

One of the commonly used similarity measures is the sum of squared difference (SSD) in intensity. The SSD assumes that the images are identical after transformation excepting noise. A similarity measure only based on intensity information such as SSD cannot use the morphological information if it is present in the source image. Contrarily, some non-rigid registration methods are based on geometric features such as anatomical surfaces. These registration methods can match surfaces in the source image with its counterpart in the target image. These methods are called surface-based registration algorithms [9]. The transformation describes the coordinate system mapping between the source and the target images. The optimization process optimizes the similarity criterion by adjusting the transformation parameters. A non-rigid transformation has a high degree of freedom because portions of the image can deform locally. Therefore, the optimization process for non-rigid segmentation requires changing more parameters. This increases the chance of the optimizer being trapped in the local minima, which does not yield the best transform [9]. Therefore, a good understanding of the registration problem is necessary to choose an appropriate optimizer.

CHAPTER 3

GENERAL IMAGE REPRESENTATIONS AND OPERATIONS

This thesis research entails digitizing a printed brainstem atlas. Scanning a brainstem atlas produces digital images. These images must be processed in order to produce a digital atlas. Processing images effectively requires understanding how images are represented and how necessary operations are performed on them. This chapter describes general image representations and relevant operations, which are mentioned in later chapters, on these images.

There are two kinds of distinct image types: vector images and raster images. Vector images are represented with mathematical shapes such as curves, ellipses, polygons, and glyphs. These images are continuous, so they can be scaled without any information loss. Unfortunately, it is impossible to capture real images without any error. Thus, the scanned images are discrete. These images with discrete values are known as raster images. Figure 4 shows an example of a how a line $y = x$ may be stored in both vector and raster images. The vector image stores the line information itself while the raster images reconstructs the line discretely.

The building blocks of raster images are called pixels in 2D images and voxels in 3D images. Each cell in Figure 4b is a pixel. The higher the pixels or voxels per physical unit area or volume, the more precise the image will be. Each pixel in the image has one value from a range of possible values. An image may have either scalar pixels or pixels with multiple components. For example, a grayscale picture has scalar pixels whereas a color picture typically has pixels with red, green, and blue components. The images with pixels containing red, green, and blue pixels are called RGB images. The most common images have pixels with intensities between 0 and 255. A 2D image can be treated as a mathematical function $I(x, y)$, so it can be manipulated in various ways.

3.1 COMMON OPERATIONS ON RASTER IMAGES

Scanning the brainstem atlases found in textbooks produces raster images. It is necessary to perform various operations on these raster images to extract relevant anatomical data. Some of the common pixel operations which include addition, multiplication, and comparison are not universally defined for RGB pixels. For example, while comparing

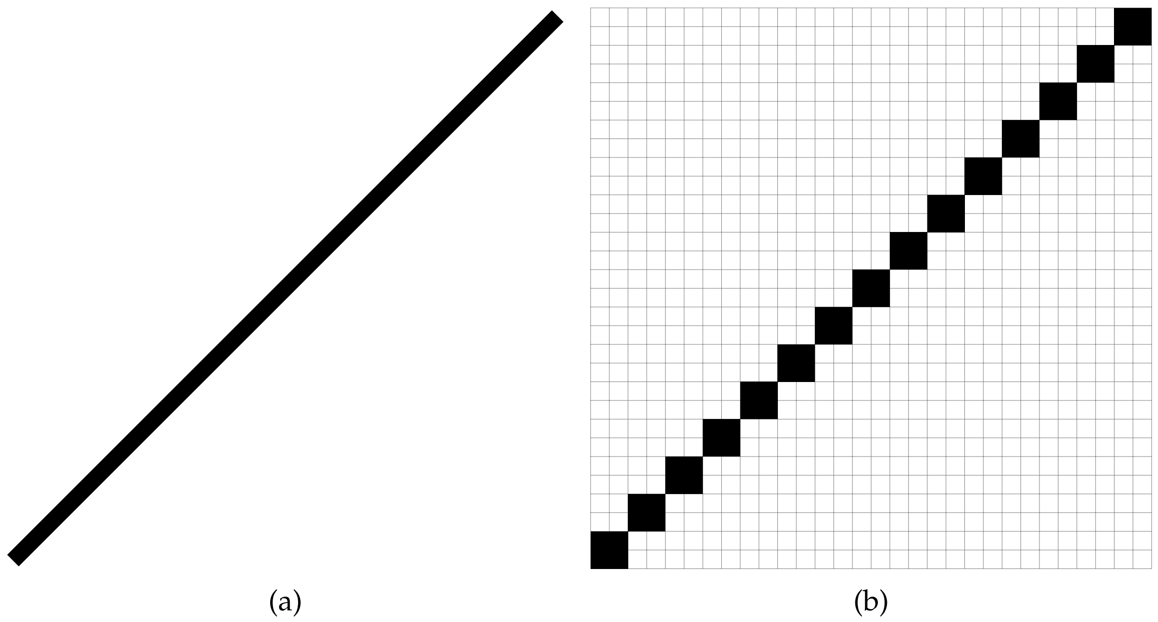


FIG. 4: A comparison of a line $y = x$ stored (a) in a vector image and (b) in a scalar image.

two RGB pixels, the red component in one pixel may be higher, the blue component may be equal, and the green component may be higher in the other pixel. One way to solve this problem is by manually defining fundamental operators such as addition and comparison for RGB pixels. Nonetheless, it is often preferred to convert the RGB images into grayscale images before the processing. Aside from the grayscale conversion algorithms, the other techniques are relevant for both regular and medical images.

3.1.1 GRAYSCALE CONVERSION

Sometimes the information loss resulting due to RGB to grayscale image conversion is acceptable. Various operations are much easier to perform on grayscale images compared to RGB images. There are several ways an RGB pixel can be converted into a grayscale pixel. One of the simplest ways is using the average of the red, green, and blue component of the pixel. Thus,

$$\text{Grayscale pixel} = \frac{\text{red} + \text{green} + \text{blue}}{3}. \tag{1}$$

However, humans do not perceive all colors equally strongly. For example, green is perceived more strongly than red and blue. A more sophisticated method, which weighs the colors according to how strongly they are perceived, uses the following formula [26] for grayscale conversion:

$$\text{Grayscale pixel} = 0.3 \times \text{red} + 0.59 \times \text{green} + 0.11 \times \text{blue}. \tag{2}$$

3.1.2 THRESHOLDING

Often, it is useful to determine which pixels in an image belong to an important structure depending on their intensities. Thresholding is used in these cases. For instance, the backgrounds in many images have a different intensity range compared to foreground objects. Removing the intensities of the background allows focusing on the objects. A simple global thresholding is described as the following:

$$f(x,y) = \begin{cases} 1 & \text{if } I(x,y) > \alpha \\ 0 & \text{if } I(x,y) \leq \alpha \end{cases} \tag{3}$$

where f is the thresholding function that is applied on each pixel, I is the image, and a is the thresholding bound. Although thresholding is often used to generate a binary image, it can generate images with more than two possible pixel values. For example,

$$f(x,y) = \begin{cases} a & \text{if } I(x,y) > \beta \\ b & \text{if } \alpha < I(x,y) \leq \beta \\ c & \text{if } \leq \alpha \end{cases} \tag{4}$$

where α and β are thresholding bounds.

3.1.3 CONVOLUTION

Convolution is used widely in image processing in algorithms such as Gaussian smoothing, Sobel edge detection, and Gabor filter. Typically, an image is convoluted with a small kernel such as a 3×3 or a 5×5 kernel. The convolution is performed on every pixel on an image. More specifically, a neighborhood with the size of the kernel is extracted around each pixel. Then, the pixel is replaced by the convolution of the neighborhood and the kernel. If N is the 3×3 neighborhood around a pixel and K is a 3×3

kernel, the convolution is calculated by

$$\begin{aligned} N * K &= \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} * \begin{bmatrix} j & k & l \\ m & n & o \\ p & q & e \end{bmatrix} \\ &= aj + bk + cl + dm + en + fo + gp + hq + ie. \end{aligned} \tag{5}$$

The sum of all elements of the kernel is usually unity. If the sum is higher than one, it is possible that the result of the convolution may exceed the allowable range of pixel intensities.

One use of convolution in image processing is Gaussian smoothing. The Gaussian distribution in 2D is described as:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right). \tag{6}$$

If the desired size of the kernel is 7×7, the values of x and y will be between -3 and 3 depending on their distances from the center. Then, a 7×7 kernel with the value of $G(x, y, \sigma)$ for each element is constructed. Consequently, the kernel is normalized by dividing the kernel by the sum of its elements. Convolving the image with this kernel will smooth the image.

3.1.4 GRADIENT CALCULATION

Image gradients are used in many different image processing algorithms. In 2D, the gradient is defined as

$$\nabla f(x, y) = \frac{\partial f}{\partial x}\hat{x} + \frac{\partial f}{\partial y}\hat{y} \tag{7}$$

where $\hat{x}$ and $\hat{y}$ are the unit vectors in the x and the y directions respectively. Derivatives are sensitive to noise, so images are often smoothed before calculating gradients. It is possible to generate the gradient of an image by combining Gaussian smoothing with the gradient calculation. Doing this leaves the original image unmodified while still having a smoothing step before gradient calculation. This is helpful because it may not be preferable to modify the original image by applying Gaussian smoothing before calculating gradients. For gradient calculation using this method, the derivatives of $G(x, y, \sigma)$ are generated with respect to x and y. Here,

$$\frac{\partial G}{\partial x} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \times -\frac{x}{\sigma^2} = -\frac{x}{\sigma^2} G(x,y,\sigma) \tag{8}$$

and

$$\frac{\partial G}{\partial y} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \times -\frac{y}{\sigma^2} = -\frac{y}{\sigma^2} G(x,y,\sigma). \tag{9}$$

A derivative of an image can be found by convolving the image with a kernel representing the derivative of a Gaussian. So, ∇I can be found by convolving $\frac{\partial G}{\partial x}$ and $\frac{\partial G}{\partial y}$ with the image. It should be noted that image gradient produces an output with two components. Often, it is the magnitude of the gradient that is required. The magnitude of the gradient at each point is

$$|\nabla I| = \sqrt{I_x^2 + I_y^2}. \tag{10}$$

The same process can be used to generate second order derivatives.

3.1.5 DISCRETIZATION OF DIFFERENTIAL EQUATIONS

Many equations are described using differential equations. Sometimes a differential equation may need to be solved for an unknown variable. For instance, a function is differentiated with respect to time, and its value at the current time is known. In order to solve for its value at the next time instance, the differentiation is discretized and solved for the unknown. There are various discretization schemes which can be used in these situations. The three common methods to approximate the differentiation are forward, backward, and central differencing schemes. Let $f(x,y)$ be a function to be differentiated at $f(i,j)$ and h be the step size. The differentiation schemes are:

Forward differencing:

$$\frac{\partial f}{\partial x} \approx D_{ij}^{+x} f = \frac{f(i+h,j) - f(i,j)}{h} \tag{11}$$

Backward differencing:

$$\frac{\partial f}{\partial x} \approx D_{ij}^{-x} f = \frac{f(i,j) - f(i-h,j)}{h} \tag{12}$$

Central differencing:

$$\frac{\partial f}{\partial x} \approx D_{ij}^{0x} f = \frac{f(x+h,j) - f(x-h,j)}{2h} \tag{13}$$

The central differencing scheme has a lower error but requires the value of two neighbors. As a result, central differencing is not possible on the edges of an image. On the other hand, forward and backward differencing have less accuracy but can be used on edges.

There is another finite differencing scheme called the upwind scheme. It uses forward or backward differencing depending on the flow of information. Depending on the direction of the flow, either backward or forward differencing may produce an unsatisfactory result, prompting the usage of an adaptive method such as the upwind scheme. The upwind differencing scheme for numerically solving $a\frac{\partial f}{\partial x}$ is

$$a\frac{\partial f}{\partial x} = \begin{cases} a \cdot D_{ij}^{-x} & \text{if } a > 0 \\ a \cdot D_{ij}^{+x} & \text{if } a < 0 \end{cases}, \tag{14}$$

which can be simplified to

$$a\frac{\partial f}{\partial x} = \max\left(a \cdot D_{ij}^{-x} f, 0\right) + \min\left(a \cdot D_{ij}^{+x} f, 0\right). \tag{15}$$

CHAPTER 4

DIGITIZING PRINTED BRAINSTEM ATLASES: A LEVEL SET APPROACH

4.1 INTRODUCTION

This chapter describes a methodology to semi-automatically process a series of brainstem sketches obtained from textbooks to generate continuously labeled digital images that are suitable for resampling and stacking to produce a digital volumetric atlas of the brainstem. This is an atlas-building process that lays the foundation for atlas-based segmentation in the future. In particular, this thesis research involves building a boundary-based atlas suitable for surface model-to image segmentation. A digitized 2D atlas is saved as an image where each region has its distinct label, which is a color. In other words, each region is filled with its distinct color. Region boundaries are recovered during 3D reconstruction, which is described in the next chapter.

4.2 METHODS

The 2D sketches of the brainstems are obtained from textbooks either digitally or by scanning. Figure 3 shows an example of such images. Before using them, they are converted to grayscale in order to simplify the process. It is difficult to automatically segment these images, so some operator supervision is necessary. The approach is analogous to paint-by-numbers toolkits where the user provides a rectangle covering each label. The rectangles then propagate to coincide with their respective regions. Each of these rectangles carries a user-supplied label. Figure 6c shows how a rectangle propagates its label. These labels are used to identify regions in the digitized image. Once all labels coincide with their perspective regions, the regions are reflected about the axis of symmetry. The axis of symmetry is shown in Figure 3 as a vertical line in each sketch. In order to simplify the objective, it is assumed that the axis of symmetry denotes perfect symmetry on both sides. A high-level overview of the process is shown in Figure 5. This methodology works for sketches with solid contours. Paxinos' [21] atlas features both solid and dashed

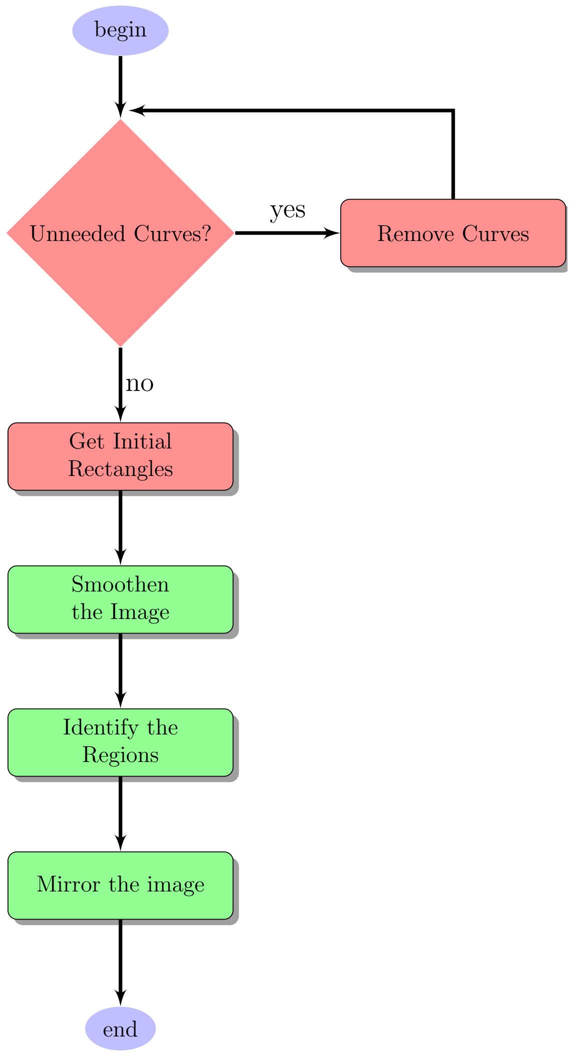


FIG. 5: A high-level flowchart showing the high-level overview of the 2D digitization process. The blocks in red require user input whereas the blocks in green do not.

contours. The level set model bled outside regions through openings in the boundaries. To solve this, dashed contours were converted to solid contours manually.

4.2.1 LEVEL SET FRAMEWORK FOR CURVE EVOLUTION

In order to propagate the user-supplied rectangles to coincide with the regions, a curve propagation method which can detect regions' boundaries in an image is needed. The level set framework provides a rich framework for curve evolution in 2D and surface evolution in 3D. Let $\varphi(x, y, t)$ be the level set function at a time t. The curve evolution occurs as t increases. A curve $\mathcal{C}$ at a time t is represented in this function using

$$\mathcal{C} = \{(x,y)/\varphi(x,y,t) = 0\}. \tag{16}$$

In other words, the curve at a time t is a set of points for which the level set function evaluates to zero. The curve $\mathcal{C}$ is also called the zero level set of φ. The sign of φ for points outside or inside the curve $\mathcal{C}$ is positive or negative depending on the particular level set method. The value of each point in φ is the distance of that point from the curve. Level set methods operate on φ to deform the curve $\mathcal{C}$. Let $I(x,y)$ be a two dimensional image. The properties of the image I will guide the evolution of the level set.

The level set methods can generally be divided into two categories: edge-based and region-based. Edge-based methods use edges to stop the curve propagation, so these algorithms require clearly noticeable edges. On the other hand, region-based methods do not require pronounced boundaries; they require that different regions in the same image have different characteristics. One of the region-based methods is the Chan-Vese method, which was tried with atlas images. It is described using the following formula [5]:

$$\frac{\partial\varphi}{\partial t} = \delta(\varphi)\left(\mu\nabla\cdot\frac{\nabla\varphi}{|\nabla\varphi|} - \nu - \lambda_1(I-c_1)^2 + \lambda_2(I-c_2)^2\right). \tag{17}$$

Equation 17 contains a curvature term κ which is

$$\kappa(\varphi) = \nabla\cdot\frac{\nabla\varphi}{|\nabla\varphi|} \tag{18}$$

where $\nabla\cdot$ is the divergence operator. Also, the Dirac delta δ is

$$\delta(x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

and $\mu > 0$, $\nu \geq 0$, $\lambda_1 > 0$, and $\lambda_2 > 0$ are constants. On the other hand, c_1 and c_2 are constants between each iteration that can be defined as

$$c_1 = \frac{\int I(x,y)\cdot H(\varphi(x,y))\,dx\,dy}{\int H(\varphi(x,y))\,dx\,dy} \tag{20}$$

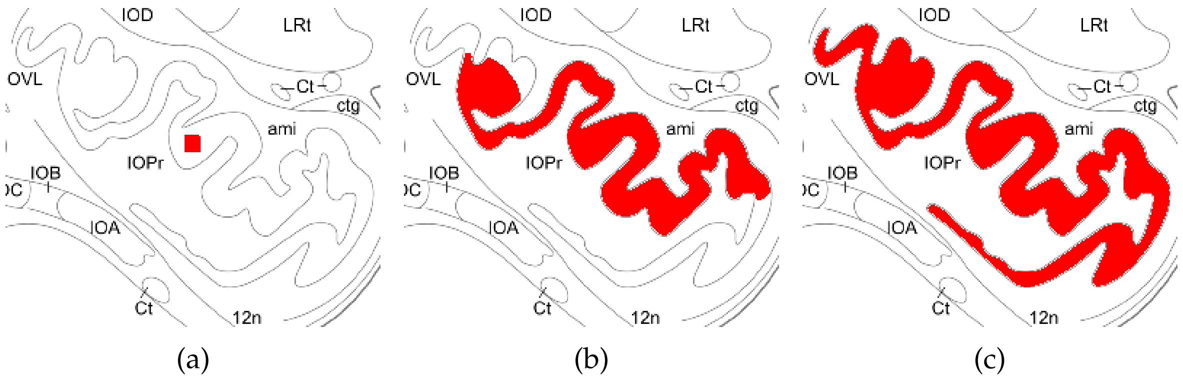


FIG. 6: Propagation of a curve: (a) A rectangle provided by the user as shown in red, (b) the curve during its evolution, and (c) the curve coinciding with the boundary of its region and stopping at the edges.

and

$$c_2 = \frac{\int I(x,y) \cdot (1 - H(\varphi(x,y)))\, dx\, dy}{\int (1 - H(\varphi(x,y)))\, dx\, dy} \tag{21}$$

where $H(x) = \int_{-\infty}^{+\infty} \delta(x) dx$ is the heaviside function. Here, c_1 is the average of the portion of I for which the corresponding level set function $\varphi \geq 0$, and c_2 is the average of the rest of the image.

As shown in Figure 3, a brainstem sketch may have multiple regions with the same characteristics, a white background in this case. Thus, Chan-Vese, a region-based method, bled through the edges and penetrated the neighboring regions when it was tested. A boundary-based method cannot yield acceptable results if it does not halt at a boundary, so the Chan-Vese method was not used further. Instead, an edge-based method called geodesic active contours was used [4].

Similar to the Chan-Vese method, geodesic active contours can be used in the level set framework. In this method, the sign of $\varphi(x,y,t)$ depends on the position of (x,y) at time t. The sign of φ is

$$\varphi(x,y) = \begin{cases} \text{negative} & \text{if } (x,y) \text{ is inside the curve} \\ 0 & \text{if } (x,y) \text{ is on the curve} \\ \text{positive} & \text{if } (x,y) \text{ is outside the curve} \end{cases} . \tag{22}$$

Initial rectangles covering each label are provided externally. These rectangles are evolved one-by-one. Let $\mathcal{C}$ be the curve to be propagated. This curve is initialized as a user-provided rectangle. As an edge-based method, geodesic active contours requires an image with noticeable edges. An edge in an image is typically recognized by detecting sudden changes in pixel intensities in small neighborhoods. One way to detect edges in an image is by examining the gradient of each pixel because the gradient magnitude describes the magnitude of change in pixel intensity. Thus, pixels on an edge have high gradient magnitudes. Geodesic active contours use this property of gradients to halt the contour deformation at a boundary.

Each point on the curve $\mathcal{C}$ moves in its normal direction until it reaches a pixel with a high gradient magnitude. Thus, a stopping function that negates the speed in the normal direction at pixels with high gradient magnitudes is used. The evolution of a curve using the geodesic active contours method is described by the following equation [4]:

$$\frac{\partial \mathcal{C}}{\partial t} = g(|\nabla I|)(c+\kappa)\vec{\mathcal{N}} - \left(\nabla g \cdot \vec{\mathcal{N}}\right)\vec{\mathcal{N}} \tag{23}$$

where I is the image, $g(|\nabla I|)$ is the image-based stopping function, κ is the curvature, $\vec{\mathcal{N}}$ is the normal, and c is a constant. The curvature κ can be calculated using Equation 18. The normal $\vec{\mathcal{N}}$ can be calculated using

$$\vec{\mathcal{N}} = \frac{\nabla \mathcal{C}}{|\nabla \mathcal{C}|}. \tag{24}$$

The stopping function, $g(|\nabla I|)$, is a function that is very small, ideally zero, at the edges. There are different functions that can be used as the stopping function. In this case, the sigmoid function was used as the stopping function. It is represented as

$$g(|\nabla I|) = \frac{1}{1+\exp\left(-\frac{|\nabla I|-\beta}{\alpha}\right)} \tag{25}$$

where α and β are constants weights. For finite real valued images and constants α and β, the result of the chosen stopping function is always a finite real value. Using a sigmoid filter, the effect of the value β with the range α can be pronounced. Since it is versatile, the sigmoid function is used as the stopping function.

During the evolution of the curve, if the curve is not close to an edge that has a high gradient the value of the stopping function will be close to one. Hence, the curve will move in the normal direction at each point. However, when a point is at an edge, the value of the stopping function will be close to zero, so the change in the point's position will be very small. However, the stopping function requires infinitely sharp edges to be zero at the edges. Since it is not possible to have them in real images, the second term in the equation is needed. The second term will make an already small first term very close to zero. Thus, the curve will stop at the edges provided that the edges are not ill-defined.

Equation 23 describes the evolution of a curve, but it does not use the level set framework. From Equation 23, the level set equation can be formulated [4] which is

$$\frac{\partial \varphi}{\partial t} = g(|\nabla I|)(c+\kappa)|\nabla \varphi| + \nabla \varphi \cdot \nabla g. \tag{26}$$

The level set formulation enables numerous benefits. For example, two curves are allowed to join or split. Furthermore, all pixels inside the curve have negative values, making it easier to determine which parts of the image belong to the interior or exterior of the region. Particularly, thresholding is used to filter the pixels inside the region. All the interior pixels are assigned the label provided by the user. On the other hand, each initial rectangular curve $\mathcal{C}$ needs to be embedded on the level set function $\varphi(x, y, 0)$. This is done by solving the following equation

$$|\nabla \varphi(x, y, 0)| = 1 \tag{27}$$

after setting all points from $\mathcal{C}$ in φ zero. A numerical way to determine φ is discussed in section 4.2.3. The interpretation of Equation 27 is that at each point, the magnitude of the gradient, which points in the normal direction, has the value of one. In other words, from the curve, the value of a pixel in the unit normal direction is one more than the current pixel's value. This produces the initial level set function φ, which is also a distance map from the curve.

4.2.2 IMAGE SMOOTHING

The 2D sketches of the brainstems are obtained from textbooks either digitally or by scanning. Scanning introduces noise in an image that may hinder the evolution of geodesic active contours. Noise is an unwanted component and needs to be minimized. In order to make the edges more immune to noise, smoothing is used to pre-process the images.

Typically, the change of pixel intensities in real images while traversing through the image pixel by pixel is small, i.e., sufficiently small regions in images are smooth. In other words, the neighboring pixels of a typical image have nearly the same intensity unless there is an edge in the neighborhood. A sudden noticeable change in pixel intensity respective to its neighbors is either noise or an edge. Smoothing algorithms try to reduce the noise by changing each pixel so that its deviation compared to its neighbors is reduced.

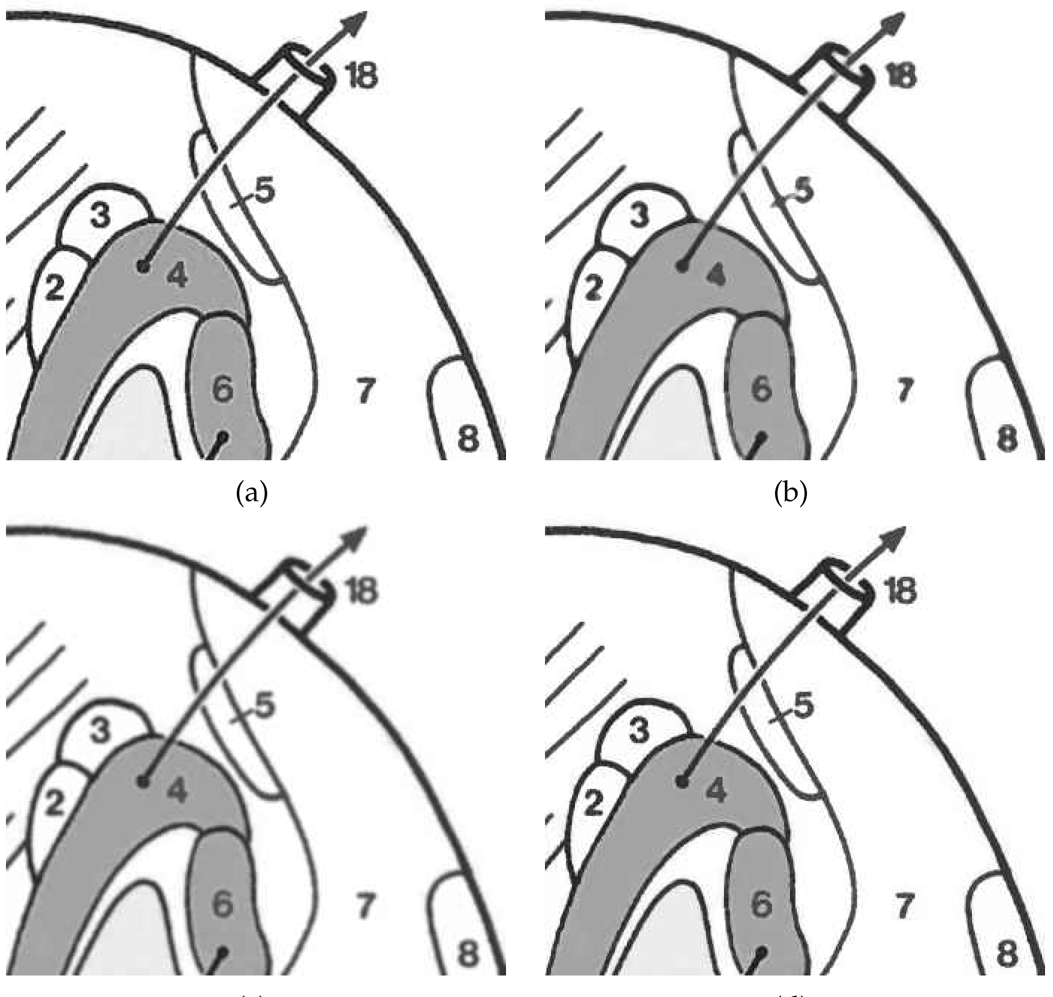


FIG. 7: Comparison of the smoothing algorithms side-by-side: (a) The input image, (b) the image smoothed using the median filter, (c) the image smoothed using the Gaussian filter, and (d) the image smoothed using the modified curvature diffusion equation.

A popular smoothing algorithm is median smoothing where each pixel is replaced by the median of all pixels in its neighborhood. The neighbors are enclosed by a window with an odd width and length. Typically, the bigger the window, the more pronounced the smoothing and the greater the reduction in noise. A median filter does not weigh the contributions of the pixels according to their distances from the center even though pixels that are nearer are more likely to be similar than the pixels which are farther. As a result, the quality of an image decreases rapidly if the median filter is applied with a big window or applied multiple times. On the other hand, a median filter window always has an odd number of elements. Thus, the median of the neighbors is guaranteed to be one of the elements. Therefore, the median filter does not introduce any new distinct value while smoothing an image.

Another popular smoothing algorithm is Gaussian smoothing, which does weigh the contributions of pixels depending on their distances. Gaussian smoothing can be described by the following equation:

$$G(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right). \tag{28}$$

Here, x and y are distances in from the center. As x and y increase, the effect of $G(x,y)$ decreases. However, Gaussian smoothing does not discriminate edges and blurs them too. This is an undesired behavior because geodesic active contours require clear edges. In order to preserve edges during smoothing, a function similar to the stopping function in geodesic active contours can be utilized. As mentioned above, the stopping function $g(|\nabla I|)$ is very small at the edges. A similar function $c(|\nabla I|)$, called the conductance function, can be introduced for smoothing. The Gaussian function can be formulated as the following PDE:

$$\frac{\partial I}{\partial t} = \nabla \cdot \nabla I \tag{29}$$

which is known as the heat equation. Like heat, the pixel intensities can be diffused in an image to simulate smoothing. Applying Equation 29 to an image would smooth all parts of the image equally. To reduce smoothing on edges, the conductance function $c(|\nabla I|)$ can be used in the following way:

$$\frac{\partial I}{\partial t} = \nabla \cdot (c(|\nabla I|)\nabla I). \tag{30}$$

Equation 30 is called the Perona-Malik diffusion, and it was introduced by Perona, Shiota, and Malik [24].

The smoothing function used in this thesis research was the modified curvature diffusion equation (MCDE) which is very similar to Equation 30. The MCDE is [32]

$$\frac{\partial I}{\partial t} = |\nabla I| \nabla \cdot \left(c\left(|\nabla I|\right) \frac{\nabla I}{|\nabla I|} \right). \tag{31}$$

Here, the curvature flow is $\kappa = \nabla \cdot \frac{\nabla I}{|\nabla I|}$. The conductance term adjusts the effect of the curvature flow. On edges, the conductance term largely negates the curvature flow, which minimizes the diffusion. The conductance function, which was chosen from [24], was

$$c(|\nabla I|) = \exp\left(- \left(\frac{|\nabla I|}{K} \right)^2 \right) \tag{32}$$

where K is the conductance parameter which is a constant used to control the algorithm's sensitivity to edges. Compared to the Perona-Malik diffusion, the MCDE is less sensitive to contrast.

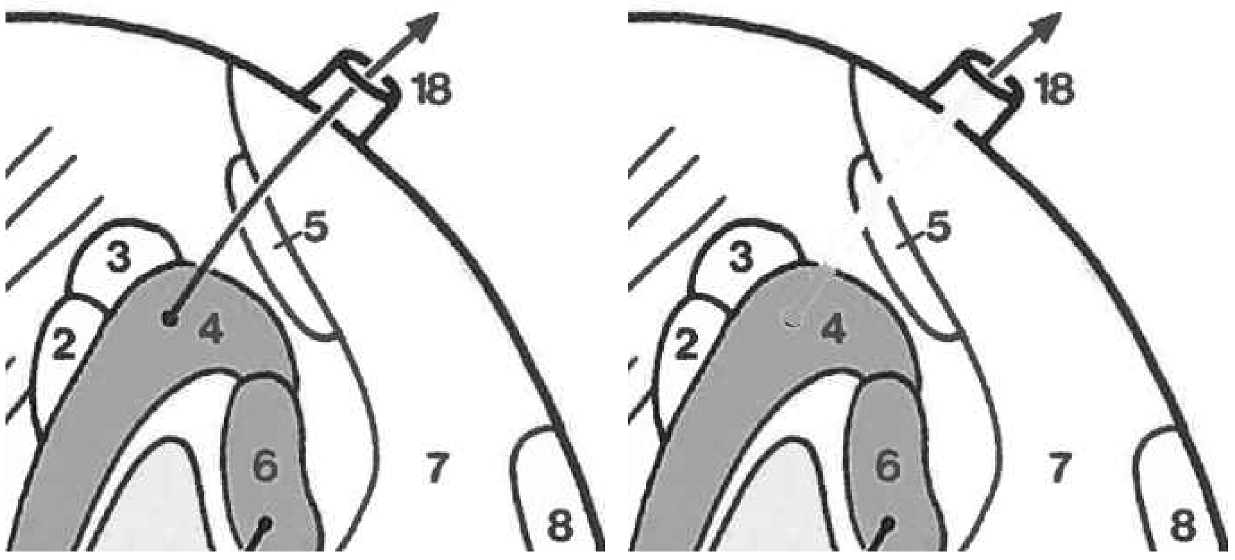
4.2.3 IDENTIFYING AND TREATING CURVED LINES

Some sketches of the brainstem may contain curved lines that may need to be treated before the digitization proceeds. For example, Figure 3a shows a brainstem sketch that has curved lines crossing multiple regions. Some of the curved lines may be cranial nerve paths or other important features, so identifying them as accurately as possible is crucial. Similar to detecting regions, it is difficult to automatically detect curved lines. Therefore, the users provides two points for each end of every curved line with optionally specifying midpoints. The Minimal Path [1], which does not escape the curved line, between two end points captures the general shape of the curved line. Once the curved line is detected, it can be removed. Figure 8 shows an example of the desired result of curved line removal.

Fast Marching for Minimal Path Detection

The Fast Marching method is used to evolve a front. In this method, a wave front is propagated from a chosen set of starting points. In the resulting image, the value of each pixel is the time T at which the front first arrives at that pixel. For the initial points, the value of T is zero. The arrival time T at a point $\mathbf{x}$ is represented as:

$$|\nabla T(\mathbf{x})| F(\mathbf{x}) = 1 \tag{33}$$



(a) (b)

FIG. 8: A side-by-side comparison of the image before and after the curve removal: (a) The input image with user provided end-points and (b) the output image with the erased curve.

where $F(\mathbf{x})$ is the speed of the wave at $\mathbf{x}$ [28]. This equation is known as the Eikonal equation. Note that the speed function F is known and the arrival time T for each pixel needs to be solved. Fast Marching is a numerical method to approximate Equation 33. Here, the speed function is always positive, so the front propagates only one way. If the speed function for each point $\mathbf{x}$ is $F(\mathbf{x}) = 1$, Equation 33 becomes $|\nabla T(\mathbf{x})| = 1$ which is Equation 27. Thus, the Fast Marching method can be used to generate $\varphi(x, y, 0)$ from an initial rectangle.

The property of a Fast Marching method's result can be exploited to calculate the shortest path between two points based on retracing a geodesic path along the gradient of the arrival time. This method is similar to Dijkstra's shortest path algorithm. In fact, it is considered the continuous version of Dijkstra's shortest path method. The arrival time can be treated as the length of the minimal path from a given point to the initial points. Since F is a speed function, it can be used to constrain the regions through which the Minimal Path must pass. In other words, the speed function can be generated such that it is close to one on boundaries and close to zero elsewhere. This results in an output image where the arrival time on points on the same curve are smaller than the points outside the curve. Thus, the Minimal Path between two end points will stay inside the line. The speed function is

$$F(\mathbf{x}) = \frac{\mathcal{V}(\mathbf{x}) - \min(\mathcal{V})}{\max(\mathcal{V}) - \min(\mathcal{V})} \tag{34}$$

where $\mathcal{V}(\mathbf{x})$ is the vesselness measure of the image at $\mathbf{x}$.

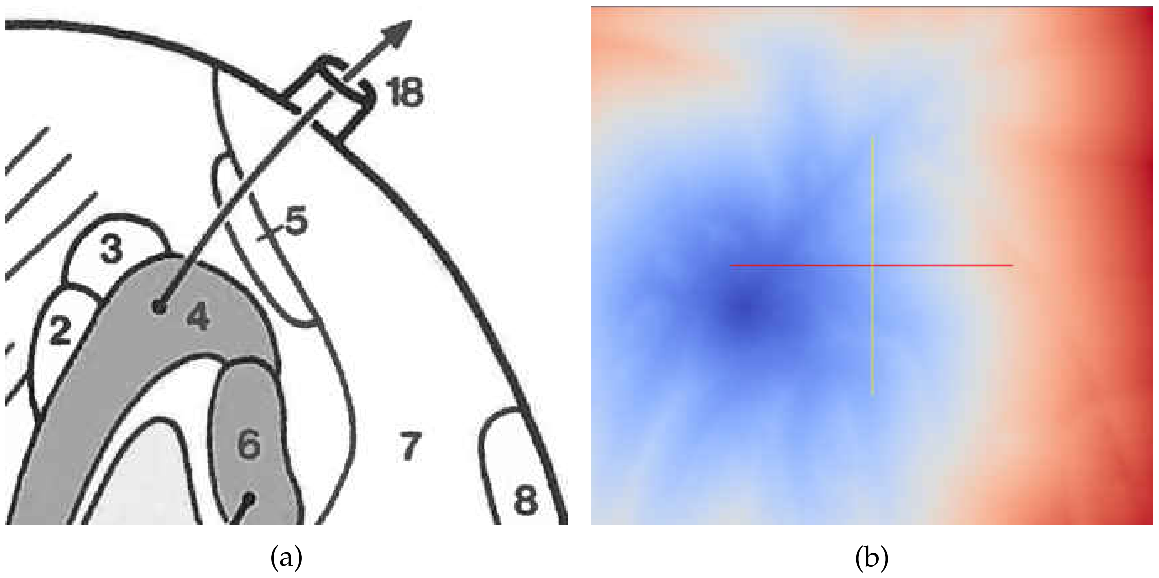


FIG. 9: A demonstration of the distance map generated using the Fast Marching method and the vesselness: (a) The input image with the initial point provided in the black filled circle in the region 4 and (b) the generated distance map showing faster arrival in blue and slower arrival in red.

Equation 33 can be simplified into

$$\left(\frac{\partial T}{\partial x}\right)^2 + \left(\frac{\partial T}{\partial y}\right)^2 = \frac{1}{F^2(x,y)}. \tag{35}$$

Since the Eikonal equation involves flow, a good scheme to numerically solve it is the upwind differencing scheme. Substituting Equation 15 into Equation 35 yields

$$\begin{aligned} &\left[\max\left(D_{ij}^{-x}T, 0\right) + \min\left(D_{ij}^{+x}T, 0\right)\right]^2 \\ &+ \left[\max\left(D_{ij}^{-y}T, 0\right) + \min\left(D_{ij}^{+y}T, 0\right)\right]^2 = \frac{1}{F_{ij}^2} \end{aligned} \tag{36}$$

at each point (i, j) [28]. Equation 36 can be solved for $T(x, y)$ to yield

$$T(x,y) = \frac{h}{F(x,y)} + \min(T(x-h,y), T(x+h,y), T(x,y-h), T(x,y+h)) \tag{37}$$

where h is the step size. Equation 37 is used to update $T(x, y)$.

The Fast Marching algorithm works as follows [28]:

1. For every point $\mathbf{x_i}$, set $T(\mathbf{x_i}) = +\infty$. These points are labeled the far away points.

2. For each initial point $\mathbf{x_{initial}}$, set $T(\mathbf{x_{initial}}) = 0$. These points are labeled the alive points.

3. For every alive point $\mathbf{x_i}$, calculate a tentative value $T_{new}(\mathbf{x_i})$. If $T_{new}(\mathbf{x_i}) < T(\mathbf{x_i})$, change $\mathbf{x_i}$ to be a trial point and set $T(\mathbf{x_i}) = T_{new}(\mathbf{x_i})$.

4. Find $\mathbf{x_j}$ which is a trial point with the smallest $T(\mathbf{x_j})$ and make $\mathbf{x_j}$ an alive point.

5. Calculate the $T_{tentative}$ value for every neighbor of $\mathbf{x_j}$ which is not currently in the alive set.

6. If, for any neighbor, $T_{tentative} < T(\mathbf{x_j})$, $T(\mathbf{x_j}) = T_{tentative}$. If $\mathbf{x_j}$ was a far away point, make it a trial point.

7. If a trial point exists, go to item 4.

An example of the output generated by the Fast Marching method is shown in Figure 9. These images show that it is much faster for the front to reach other parts inside the arrow compared to outside the arrow. In order to find the minimal path between a point $\mathbf{p}$ and the initial point $\mathbf{p_0}$, the gradient descent method is used by solving the following:

$$\frac{d}{dt}\gamma(t) = -\nabla T(\gamma(t)) \quad \text{and} \quad \gamma(0) = \mathbf{p} \tag{38}$$

until $\gamma(t_f) = \mathbf{p_0}$ for some t_f. Here, γ is the minimal path.

Vesselness Measure of an Image

The Fast Marching method requires a speed function. This speed function should have a high speed on curved lines and low speed elsewhere. A speed function with this desired property is generated by extracting the vesselness measure from the image. The vesselness filter is used to extract tubular structures resembling vessels from the image. Since edges and curves in the image are similar to tubular structures, they can be identified using vesselness filtering. To extract vesselness, a Hessian matrix is calculated for each pixel. The Hessian matrix is described by the following equation:

$$H = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{yx} & I_{yy} \end{bmatrix} \tag{39}$$

where I is the image. Two eigenvalues λ_1 and λ_2 of the Hessian matrix are calculated and then sorted such that $|\lambda_1| \leq |\lambda_2|$. It can be determined whether the pixel is in a tubular structure depending on the eigenvalues as shown in Table 1.

λ_1	λ_2	
L	H-	Tubular structure (bright)
L	H+	Tubular structure (dark)
H-	H-	Blob-like structure (bright)
H+	H+	Blob-like structure (dark)

TABLE 1: Classification of pixels depending on the eigenvalues of the Hessian matrix (H=high, L=low, +/-=sign of the eigenvalue) [12].

The underlying local structure in the image is captured by the combination of eigenvalues of the Hessian. For example, if two eigenvalues are both high values, the local structure is blob-like; if one eigenvalue is low and the other is absolutely high, the local structure is tubular. The most relevant to this situation is a combination of a high positive value and a low value that coincides with a dark vessel-like structure. After it has been determined which pixels are inside tubular structures, the vesselness can be extracted using the following function [12]:

$$\mathcal{V}_o(s) = \begin{cases} 0 & \text{if} \lambda_2 > 0 \\ \exp\left(-\frac{\mathcal{R}_\mathcal{B}}{2\beta^2}\right)\left(1 - \exp\left(-\frac{\mathcal{S}^2}{2c^2}\right)\right) & \text{otherwise} \end{cases} \tag{40}$$

where β and c are constants, $\mathcal{R}_\mathcal{B} = \frac{\lambda_1}{\lambda_2}$ is the blobness measure, and $\mathcal{S} = \sqrt{\lambda_1^2 + \lambda_2^2}$. Here, β and c are weighing parameters that control the sensitivity of $\mathcal{R}_\mathcal{B}$ and $\mathcal{S}$ respectively.

At every point, the image is twice differentiable, so $f_{xy} = f_{yx}$. Therefore, the Hessian matrix is symmetric. As such, it has two orthonormal eigenvectors. One eigenvector points to the direction of the greatest change while the other eigenvector points to the direction of the least change. If the pixel is inside a tubular structure, one eigenvector points along the tubular structure while the other eigenvector points across it. The eigenvalues associated with these eigenvectors are called the principal curvatures. If an $|\lambda_1| << |\lambda_2|$, it suggests that the change in one direction is much greater than the change in its orthogonal direction. Thus, it is possible to determine which pixels belong to tubular structures.

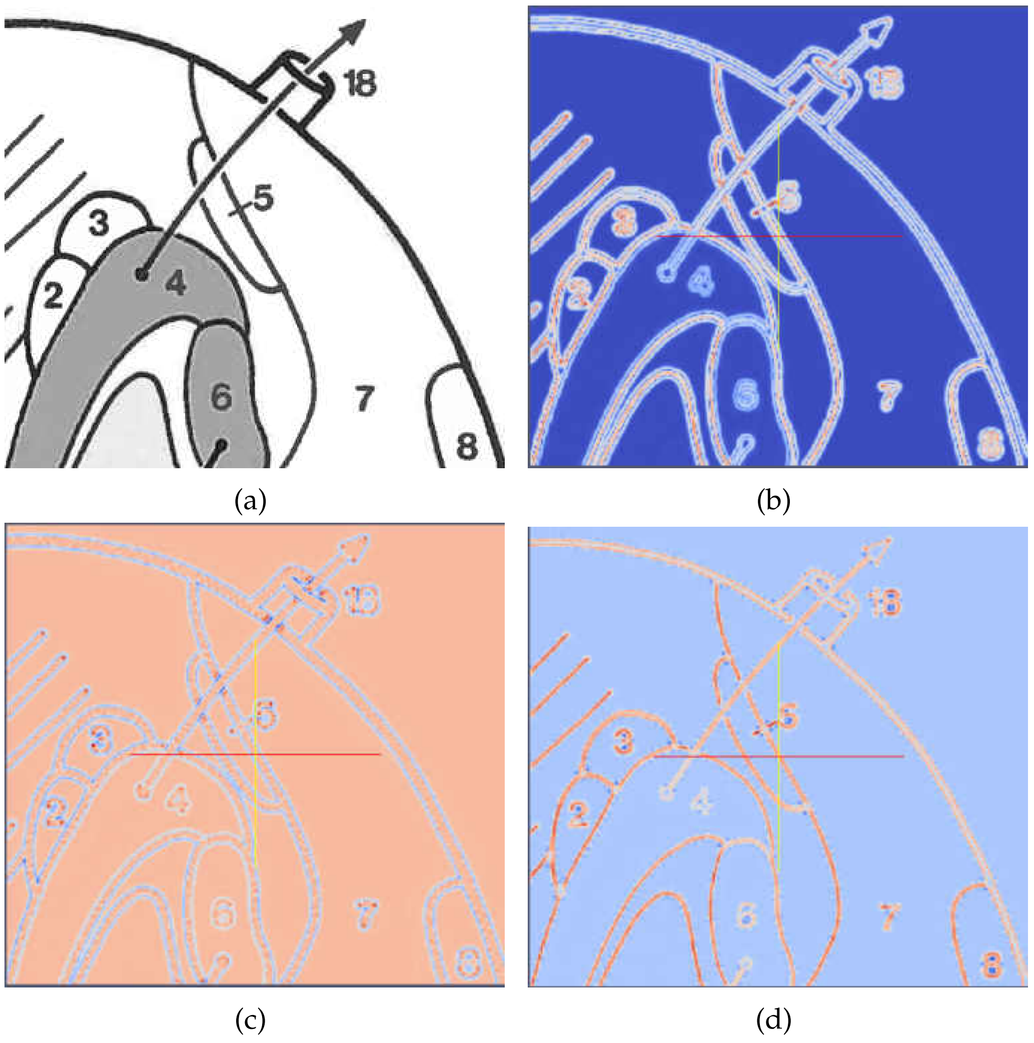


FIG. 10: Eigenvalues ($|\lambda_1| \leq |\lambda_2|$) of the Hessian matrix of an image: (a) The input image, (b) $\mathcal{S} = \sqrt{\lambda_1^2 + \lambda_2^2}$, (c) the smaller eigenvalue, and (d) the bigger eigenvalue.

Curve Removal

Once a curved line is detected, it needs to be removed to avoid impeding the evolution of geodesic active contours. However, a curved line cannot be naively erased by replacing the value of every pixel inside it with a predetermined constant. Since different regions may have different background intensities, one of the ways to remove a curved line is by replacing it in small portions with the local background intensity. For example, in Figure 8, an arrow originates in region 4 and passes region 5 and 7. The portion of the arrow in region 4 should be replaced by a different intensity compared to the rest of the arrow.

Although the Minimal Path, and consequently the general shape, of a curved line is known, the entire area of the line is unknown. Therefore, the Minimal Path is used as an initialization for geodesic active contours. The geodesic active contours propagate to cover the entire area of the curved line. Once the entire area is covered, the pixels belonging to the line are replaced by the median of the outside pixels in the neighborhood. This allows the curved line to be replaced by the local background intensity of its surrounding region. Therefore, if a curve passes multiple regions with different characteristics, different segments of the curves are replaced accordingly. Once the curves are removed, they may leave an opening on the boundaries of the regions they intersect. In this case, the user can input lines to fill the openings. Figure 8 shows an example of a curve before and after its removal.

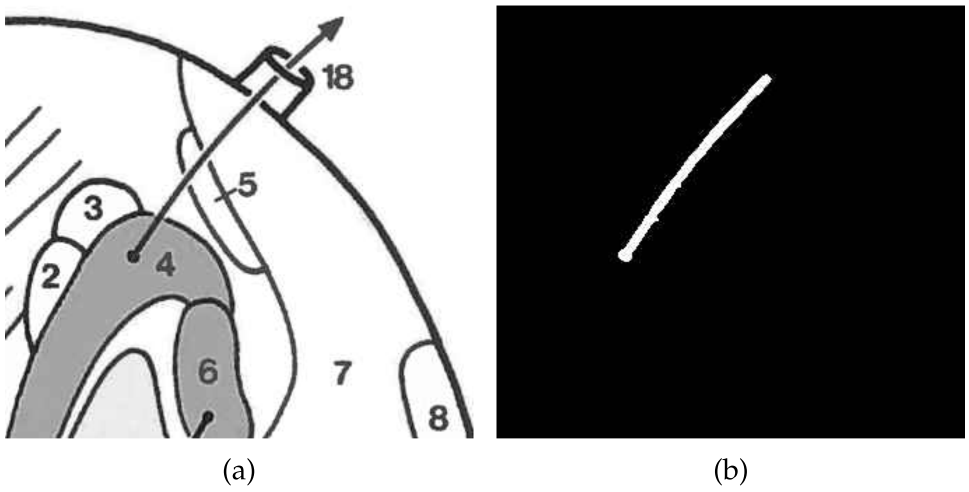


FIG. 11: An example of curve detection: (a) The input image and (b) the image showing the detected area of the curve.

4.2.4 POST-PROCESSING

Once all user-supplied rectangles have detected their perspective regions and propagated their values, these regions are overlaid on one another to produce a digitized image featuring each region containing pixels with a distinct value. However, since the active contours stop their evolution at edges and edges have a non-zero width, there is empty space between regions where there was a boundary in the sketch. The regions typically have boundaries with a constant width. Therefore, the regions are dilated by allowing the level sets to continue, very briefly, propagating outwardly after convergence at the boundary. This makes the resulting segmentation continuous which is necessary since the brainstem does not have empty spaces between regions.

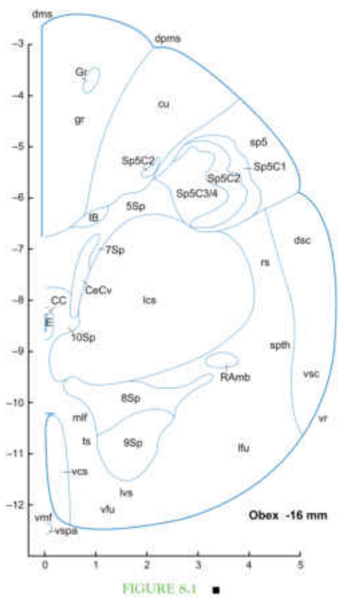
Each brainstem sketch contains an axis of symmetry about which the brainstem is symmetric. It is assumed that the axis of symmetry in each image to be digitized is vertical, or rather aligned with the y-axis. Furthermore, the methodology assumes that the axis of symmetry signifies an exact reflection of one side onto the other. The user provides a point on the axis of symmetry that determines the origin on the lateral axis, which coincides with the x-axis in the textbook. Then, the digitized image is cropped so that it begins at the origin on the lateral axis. Then, the image is reflected about the origin and padded to get a symmetric digitized atlas.

4.3 RESULTS AND DISCUSSION

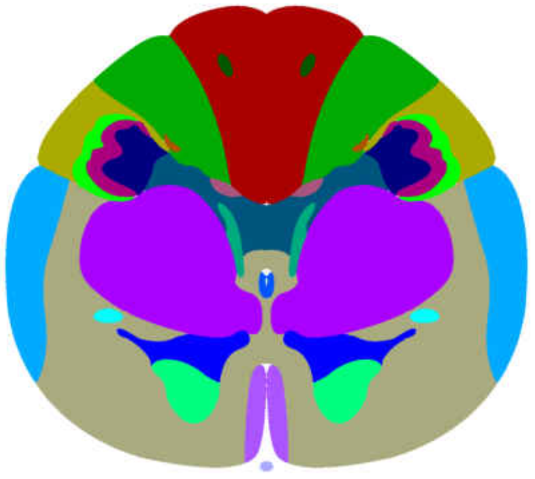
Figure 12 and Figure 13 show side-by-side the input sketches and the digitized outputs. These digitized slices are suitable for programs that require digitized brainstem slices. As shown in the aforementioned figures, each region has a distinct label, which is stored as a color. Therefore, it is easy to determine the number of regions and other information such as size and placement of each region. Since they are mostly continuous and suitable for resampling, in theory they can be used to reconstruct a 3D digital atlas of the brainstem.

The brainstem sketches are similar to the sketches found in textbooks for various other organs. On the other hand, different kinds of atlases may have different small features. As a result, it is hard to develop a methodology which is general enough to work on different kinds of atlases. Therefore, a limited user input allows segmentation of different kinds of atlases. The user input is required in the beginning of the process for each image, so the most time-consuming process, which is the level set evolution, does not require supervision. Therefore, the supervision in the beginning is a reasonable trade-off. For instance, Duvernoy's atlas was used in the beginning for segmentation. However, it was found to have insufficient resolution along the longitudinal axis, making a meaningful reconstruction impossible. As a result, the Paxinos brainstem atlas, which exhibits finer-grained longitudinal resolution, was used. Since it already required supervision, the methodology did not require significant changes. One of the differences between the two atlases is that in Duvernoy's atlases, the axis of symmetry is more prominent, whereas in Paxinos' atlas, the axis of symmetry is at 0 mm on the x-axis. Therefore, an automatic detection of the axis of symmetry may have produced erroneous results.

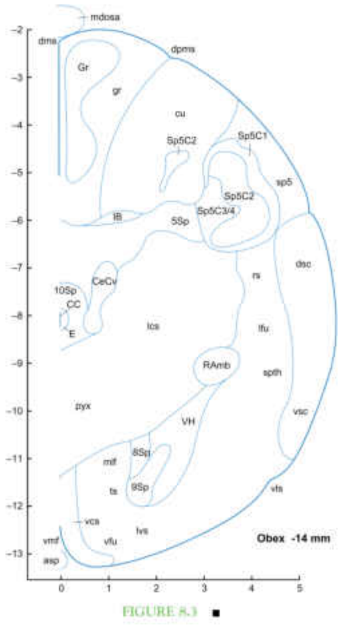
One limitation of this methodology is that it requires closed contours. Another limitation is that the level set often does not flow through a very narrow portion of the region. For example, there is a narrow region between region 4 and region 11 in Figure 13c. The digitized result shows that the level set does not completely propagate through the narrow trail. This problem can be mitigated by scanning a higher-resolution image, but it requires more processing time. On the other hand, removing curved lines often leaves gaps in the boundaries it crosses. If the gaps are left untreated, they cause the level sets to bleed outside their regions. In order to solve this problem, the software tool allows the operator to fill the gaps by manually drawing a line by scribbling.



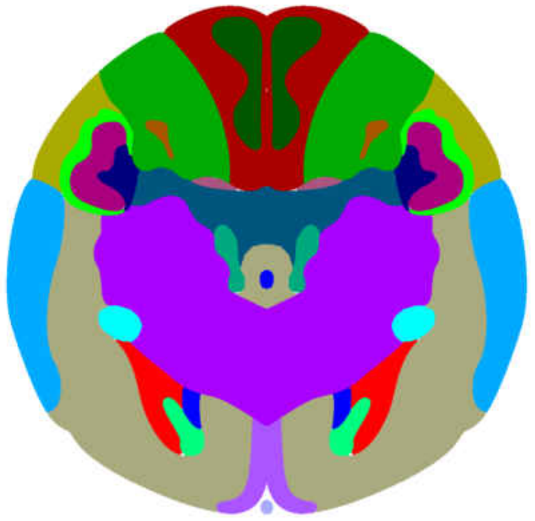
(a)



(b)

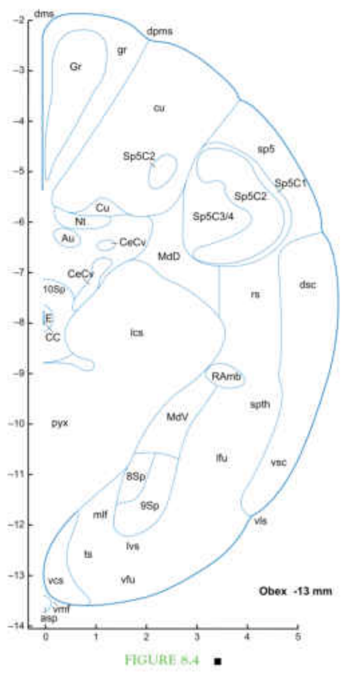


(c)

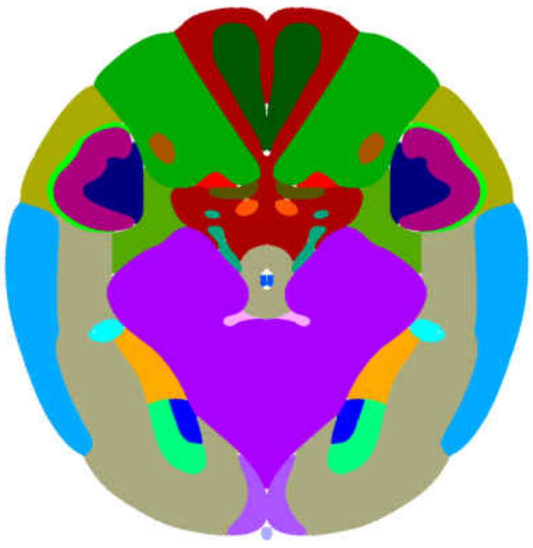


(d)

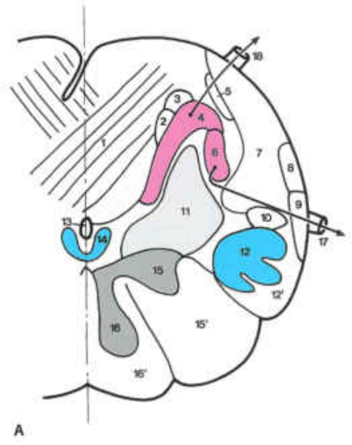
FIG. 12: Input and digitized 2D atlas of the brainstem: (a) and (c) are the input images, and (b) and (d) are digitized images.



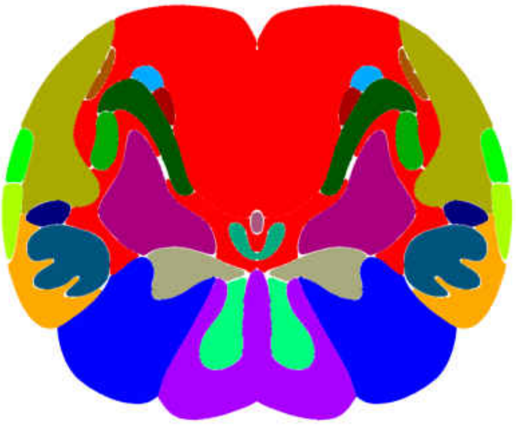
(a)



(b)



(c)



(d)

FIG. 13: Input and digitized 2D atlas of the brainstem: (a) and (c) are the input images, and (b) and (d) are digitized images.

CHAPTER 5

ROBUST 3D RECONSTRUCTION WITH DISCRETE CONTOUR MODELS

5.1 INTRODUCTION

The previous chapter described a level-set contour model, which performed well for a printed atlas with well defined boundaries but did not fare as well with the Paxinos atlas without modification. In addition, this model did not produce a result that is easily interpolated, which is needed to produce a label volume with isotropic or quasi-isotropic sampling.

This chapter presents a second contour model, based on a 2D adaptation of a 3D 1-simplex deformable contour model [30]. First of all, this contour model is resistant to bleeding effects through small contour openings. Second, this contour model is based on vertices linked by edges, which shows that the vertex motion between two slices is well defined, producing vectors of relative motion that can be interpolated. This model is used in conjunction with the output of the level set contour model as described in the previous chapter. In particular, the 1-simplex model operates on the digitized images, not the scanned images.

Once the 2D sketches of the brainstem are digitized, they need to be stacked to reconstruct a 3D brainstem atlas. In order to perform a smooth reconstruction, the volume between two consecutive slices needs to be interpolated. If the interpolation is not smooth, the resulting volume would be discontinuous at various points, which is not acceptable. Furthermore, a meaningful reconstruction requires an adequate number of slices. This chapter describes a method to reconstruct different regions of the brainstem one-by-one. Contrary to the 2D digitization portion, a user interface for 3D interpolation was not developed. Instead, this section presents a methodology which can be used to develop such an interface.

5.2 METHODS

In order to reconstruct a region from 2D slices, it is necessary to interpolate the region boundary between two consecutive slices. Then, the interpolated data need to be inserted between the slices. If the interpolation is done correctly, the resulting volume is smooth. Since the morphological variations between consecutive slices are not minimal, naively stacking them does not produce an acceptable volume. The 1-simplex meshes are used in order to interpolate regions between two slices. An n-simplex mesh is an (n+1)-connected mesh [10]. For example, a 1-simplex is a 2-connected contour and 2-simplex is a 3-connected surface. The level set contours in 2D are replaced by 1-simplex meshes because the discrete nature of the latter enables interpolation in between the slices. The simplex meshes need to be initialized with the boundary of a region in the initial, typically the first, slice. Because this is a discrete contour model, it benefits from an initialization based on a linearization of a contour, which is provided by the Marching Squares method.

5.2.1 MARCHING SQUARES

In order to create a contour using vertices and edges, the Marching Squares algorithm is used [19]. Naively extracting all points from a contour in an image yields points which are unordered. For instance, a naive algorithm scans for boundary pixels from left to right. These points are not necessarily connected to each other. In order to extract points on a boundary in the order they are connected, a more sophisticated algorithm is required.

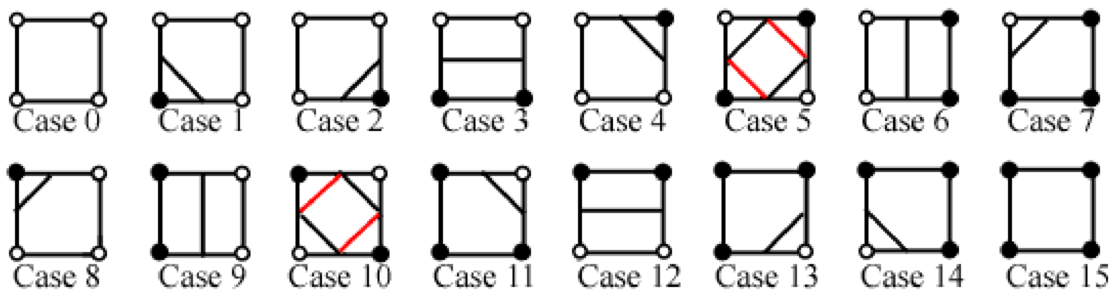


FIG. 14: All 16 possibilities of the Marching Square algorithm showing how a boundary may pass through the square depending on the value of the corners [19].

Marching Squares is a discretization algorithm, which polygonalizes the tissue boundary in a binary image. In Marching Squares, the image is divided in square cells of 2×2.

Each of the pixels in the window has one of two states depending on the pixels on the corners. There are 16 total possibilities. It can be inferred from one of those 16 possibilities whether the boundary is contained within this cell and the orientation of the boundary. For instance, if all four cells are true or false (white or black in Figure 14), the cell does not have a boundary inside it. However, if one cell is true, the boundary is adjacent to it. All 16 possibilities are shown in Figure 16. For each cell, depending on the states of the four pixels, it is possible to detect the direction of the boundary in that cell. After this process is done on all cells, the shape of the boundary is determined by stitching together the polygon segments. Once the cells containing the boundary are detected, the orientation of the boundary is known in each cell. Therefore, given a boundary cell, it is possible to predict which of its neighbor cell contains a boundary. Thus, it is possible to list all the points in order by starting at a boundary cell, going to the next boundary cell, and repeating the process until the first cell is found or the boundary has ended [19].

Once all boundary points are found in order, they are decimated since the simplex method works best with a relatively sparse collection of vertices and edges, which minimizes the number of local optima in the contour model-fitting. Although it is not necessary, it is generally better to construct a mesh with equally spaced vertices. This is achieved by using every n^{th} point from the list and discarding the rest because the Marching Squares method typically extracts points that are almost uniformly distanced. The output of this computational stage is a collection of vertices and edges at depth z= 0, which is used to initialize the discrete contour at other longitudinal coordinates.

5.2.2 1-SIMPLEX

A 1-simplex mesh is a deformable contour model for minimally supervised boundary-based segmentation, which previously has been published in 3D [30]. A 1-simplex mesh is initialized for a region in the initial slice at $z = n$ and is deformed to coincide with the same region in the consecutive slice at $z = n + 1$. Because the mesh deformation occurs when vertices move, the deformation is represented as a series of velocity vectors on each vertex. Estimating the contour at a z value entails interpolating this vector field defined along the contour. As long as the simplex mesh is stable and the change of morphology between two consecutive slices is not too big, a smooth and meaningful interpolation is possible. This process can be done separately for each separate contour at a given z coordinate.

Specifically, k-simplex is a $(k + 1)$-connected union of k-cells. A k-cell is a union of

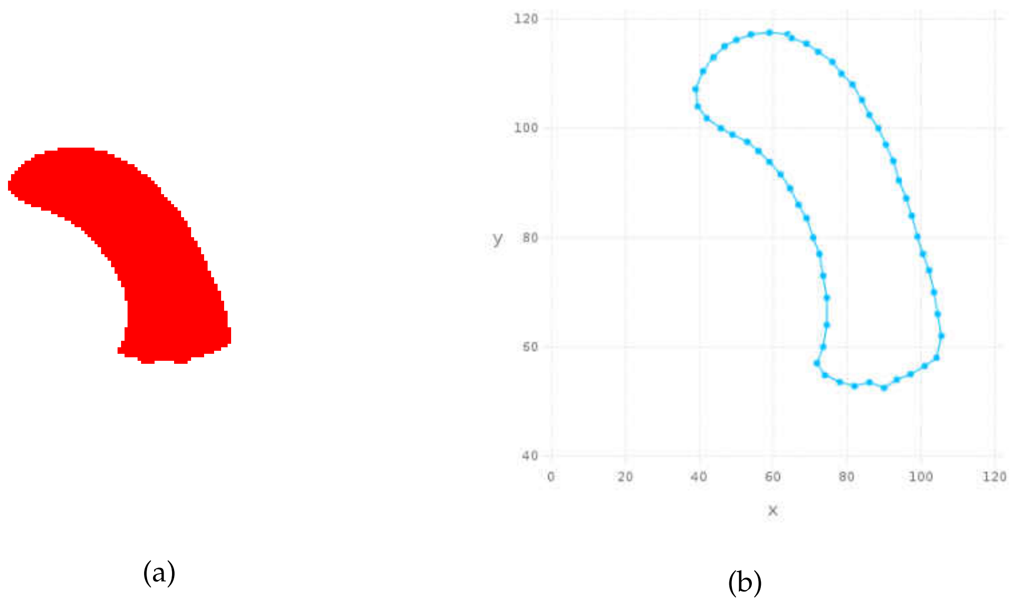


FIG. 15: A 1-simplex mesh corresponding to a region: (a) The input image with a region to be interpolated and (b) a 1-simplex mesh constructed to represent the boundary of the region in (a).

$(k - 1)$-cells. For example, a 0-cell is a vertex, a 1-cell is an edge, and a 2-cell is a face. Figure 15 shows an example of a 1-simplex mesh. This thesis research uses 1-simplex meshes for interpolation because the simplex meshes need to represent contours of regions. Each vertex in a 1-simplex mesh undergoes an internal and an external force. The internal force is originated due to the geometry of the simplex mesh and provides control to prevent strong discontinuities in the contour model. The external force is exerted on the simplex mesh by the image. The 1-simplex mesh model is adapted from the 2-simplex mesh model developed by Delingette [10]. Using the internal and external forces, the motion of each vertex in the 1-simplex can be described by the following [10]:

$$m\frac{d^2\mathbf{P_i}}{dt^2} = -\gamma\frac{d\mathbf{P_i}}{dt} + \vec{F}_{int} + \vec{F}_{ext}. \tag{41}$$

Here, m is the vertex mass, γ is the damping factor, $\mathbf{P_i}$ is the position of the i^{th} vertex, $\vec{F}_{int}$ is the internal force, and $\vec{F}_{ext}$ is the external force. It is assumed that all vertices have the same mass $m = 1$. The Equation 41 can be discretized as [10]:

$$\mathbf{P_t^{t+1}} = \mathbf{P_i^t} + (1-\gamma)(\mathbf{P_i^t} - \mathbf{P_i^{t-1}}) + \alpha_i\vec{F}_{int} + \beta_i\vec{F}_{ext}. \tag{42}$$

Here, α_i and β_i are weights for the i^{th} vertex, which are the same set of values for all vertices in this thesis project.

As shown in Figure 16b, the coordinates of $\mathbf{P_i}$ relative to its neighbors can be decomposed into two directions: the normal vector direction and the tangent vector direction. The tangential component is the orthogonal projection of $\mathbf{P_i}$, called $\mathbf{F_i}$, on the line $\mathbf{P_{i+1}P_{i-1}}$. The normal component L is the distance between $\mathbf{P_i}$ and $\mathbf{F_i}$. Given L and $\mathbf{F_i}$, the coordinates of $\mathbf{P_i}$ can be reconstructed using

$$\mathbf{P_i} = \mathbf{F_i} + L \cdot \vec{n}. \tag{43}$$

Here, $\mathbf{F_i}$ can be represented using $\mathbf{P_{i-1}}$ and $\mathbf{P_{i+1}}$ because it is on the line $\mathbf{P_{i+1}P_{i-1}}$. The barycentric coordinate system allows representing coordinates with respect to their neighbors. For a point on a line, two coefficients ϵ_{1i} and ϵ_{2i} as well as the two end-points form a barycentric coordinate system where

$$\mathbf{F_i} = \epsilon_{1i}\mathbf{P_{i-1}} + \epsilon_{2i}\mathbf{P_{i+1}}. \tag{44}$$

Here, $\epsilon_{1i} + \epsilon_{2i} = 1$. If $\epsilon_{1i} = 0$, $\mathbf{F_i} = \mathbf{P_{i+1}}$ and vice-verse. All three points are equidistant if $\epsilon_{1i} = \epsilon_{2i} = 0.5$. Essentially, ϵ_{1i} and ϵ_{2i} are normalized distances from the neighbors. According to Equation 43, finding L is necessary. As shown in Figure 16b, L can be written as the function of r_i, d_i, and ϕ_i. Here, r_i is the radius of the circle formed by $\mathbf{P_i}$, $\mathbf{P_{i-1}}$, and $\mathbf{P_{i+1}}$, d_i is the distance from $\mathbf{F_i}$ to the center $\mathbf{O}$ of the circle, and $\phi_i = \pi - \angle\mathbf{P_{i-1}P_iP_{i+1}}$. Thus,

$$L(r_i, d_i, \phi_i) = \frac{(r_i^2 - d_i^2)\tan(\phi_i)}{\xi\sqrt{r_i^2 + (r_i^2 - d_i^2)\tan^2(\phi_i)} + r_i} \tag{45}$$

where

$$\xi = \begin{cases} 1 & \text{if } |\phi_i| < \frac{\pi}{2} \\ -1 & \text{if } |\phi_i| > \frac{\pi}{2} \end{cases}. \tag{46}$$

Substituting Equation 44 and Equation 45 into Equation 43, $\mathbf{P_i}$ can be written as:

$$\mathbf{P_i} = \epsilon_{1i}\mathbf{P_{i-1}} + \epsilon_{2i}\mathbf{P_{i+1}} + L(r_i, d_i, \phi_i) \cdot \vec{n}_i \tag{47}$$

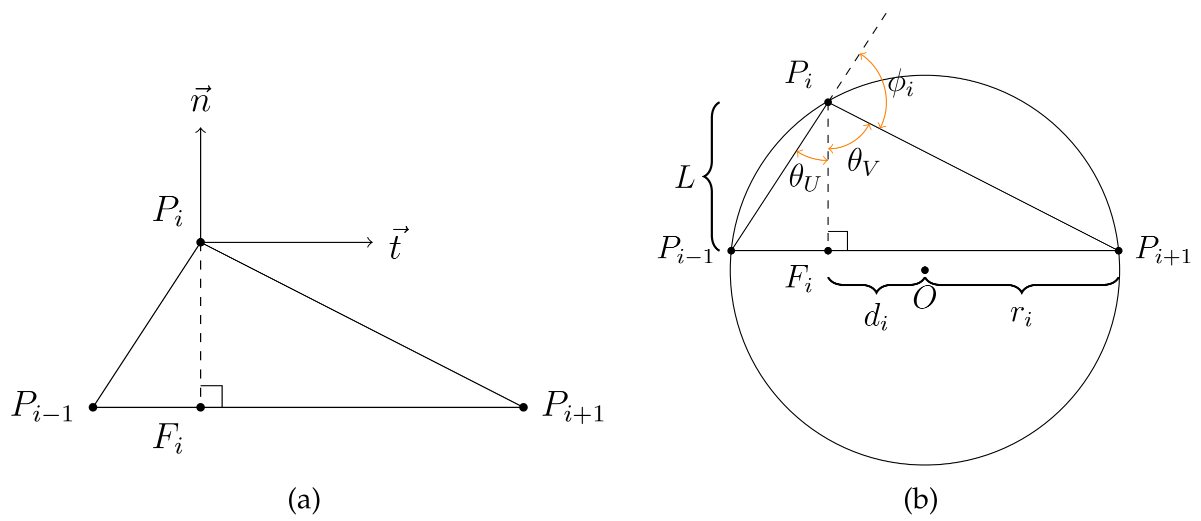


FIG. 16: The local neighborhood for each vertex $\mathbf{P_i}$ where (a) the position of $\mathbf{P_i}$ can be decomposed into normal and tangential directions and (b) the value of L depends on r_i, d_i, and ϕ_i.

Here, d_i can be simplified to $d_i = |2\epsilon_{1i} - 1| r_i$. If the desired next position of $\mathbf{P_i}$ is $\widetilde{\mathbf{P}}_\mathbf{i}$, $\vec{F}_{int} = \widetilde{\mathbf{P}}_\mathbf{i} - \mathbf{P_i}$. Here, $\widetilde{\mathbf{P}}_\mathbf{i}$ is defined with respect to the neighbors of $\mathbf{P_i}$, the simplex angle $\widetilde{\phi}_i$, and the parameters $(\widetilde{\epsilon}_{1i}, \widetilde{\epsilon}_{2i})$. Therefore [10],

$$\vec{F}_{int} = \widetilde{\epsilon}_{1i}\mathbf{P_{i-1}} + \widetilde{\epsilon}_{2i}\mathbf{P_{i+1}} + L(r_i, d_i, \widetilde{\phi}_i) \cdot \vec{n}_i - \mathbf{P_i} \tag{48}$$

where

$$\vec{t}_i = \frac{\mathbf{P_{i+1}} - \mathbf{P_{i-1}}}{|\mathbf{P_{i+1}} - \mathbf{P_{i-1}}|} \quad \text{and} \quad \vec{n}_i = \vec{t}_i^{\perp}. \tag{49}$$

The internal force depends on the continuity condition imposed. The 1-simplex mesh allows various continuity conditions which include:

C^0 Continuity allows vertices to freely bend. This will allow the simplex mesh to coincide with very rough contours at the cost of smoothness. For C^0 continuity, $\widetilde{\phi}_i = \phi_i$.

C^1 Continuity tries to force the simplex mesh to become smoother by setting $\widetilde{\phi}_i = 0$ and making the first derivatives continuous. This removes the normal force which reduces the relative elevation of a vertex relative to its neighbors. Simplifying F_{int}

leads to:

$$\vec{F}_{int} = \widetilde{\epsilon}_{1i}\mathbf{P_{i-1}} + \widetilde{\epsilon}_{2i}\mathbf{P_{i+1}} - \mathbf{P_i}. \tag{50}$$

C^2 Continuity is curvature continuity which tries to force the simplex mesh so that $\widetilde{\phi}$ is the average of ϕ_{i-1}, ϕ_i, and ϕ_{i+1}. Thus,

$$\vec{F}_{int} = \widetilde{\epsilon}_{1i}\mathbf{P_{i-1}} + \widetilde{\epsilon}_{2i}\mathbf{P_{i+1}} - \mathbf{P_i} + L(r_i, \widetilde{d}_i, \frac{\phi_{i-1} + \phi_i + \phi_{i+1}}{3}). \tag{51}$$

Shape Continuity tries to keep the shape similar to its initial shape. In order to enforce this, $\widetilde{\phi}_i = \phi_i^0$. Thus,

$$\vec{F}_{int} = \widetilde{\epsilon}_{1i}\mathbf{P_{i-1}} + \widetilde{\epsilon}_{2i}\mathbf{P_{i+1}} - \mathbf{P_i} + L(r_i, \widetilde{d}_i, \phi_i^0). \tag{52}$$

The external force can be described as the sum of gradient and edge force. The gradient force is

$$\vec{F}_{grad} = \beta_i^{grad}((\mathbf{G_i} - \mathbf{P_i}) \cdot \vec{n}_i)\vec{n}_i \tag{53}$$

where β_i^{grad} is a weighing parameter and $\mathbf{G_i}$ is the pixel with the highest gradient intensity in $m \times m$ window. On the other hand, the edge force is

$$\vec{F}_{edge} = \beta_i^{edge}(\mathbf{E_i} - \mathbf{P_i}) \tag{54}$$

where β_i^{edge} is the weighing parameter, $\mathbf{E_i}$ is the closest edge pixel along the normal line, and $\mathbf{E_i} - \mathbf{P_i}$ is collinear with $\vec{n}_i$ [10]. In this thesis research, the edge force is not adopted. Thus, $\beta^{edge} = 0$.

5.2.3 B-SPLINE-BASED LONGITUDINAL CONTOUR INTERPOLATION

Between two consecutive slices, a 1-simplex mesh is used to interpolate a region. During the evolution of the simplex mesh, each iteration represents that region at a different point on the z-axis. However, a 1-simplex is a discrete contour model, so a closed 1-simplex mesh is essentially a polygon. The boundary of a simplex mesh must be smooth in order for it to resemble an anatomically suitable border. In other words, the points between the vertices must be interpolated so that there are no visible angles at any point.

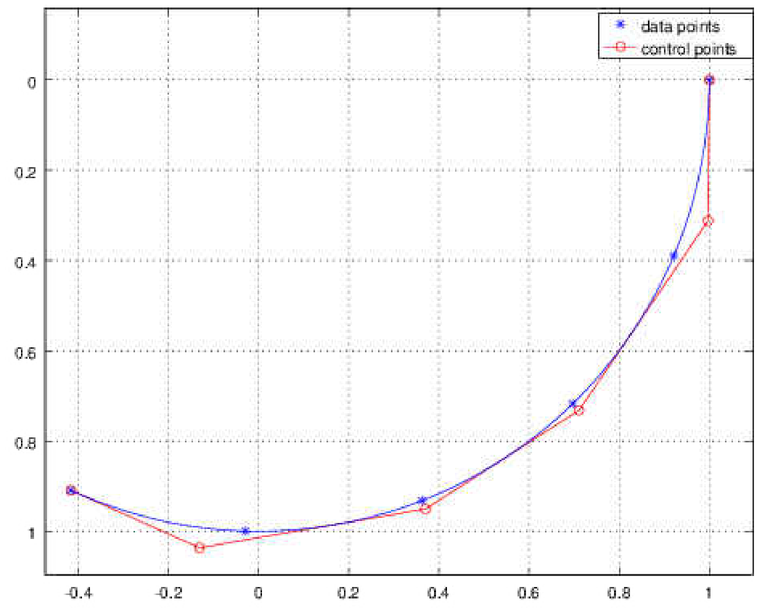


FIG. 17: An interpolated B-spline curve with control points. The control points are not the data points through which the curve must pass.

In order to achieve this, B-spline interpolation is used. B-spline interpolation produces a smooth curve that passes through all data points. Generating a B-spline using the vertices from 1-simplex mesh yields a smooth boundary from the polygon mesh. B-spline interpolation is dependent on the order of the spline. Figure 18 shows the difference between the results of different orders.

A B-spline requires control points as shown in Figure 17 and a knot vector. In order to create a B-spline that passes through all data points, control points and the knot vector have to be calculated first. Let the order of the spline be p, control points be $\mathbb{P} = \{\mathbf{P_0}, \mathbf{P_1}, ..., \mathbf{P_n}\}$, and the knot vector be $\mathbf{U} = (u_0, u_1, u_2, ..., u_{n+p+1})$. Let the data points be $\mathbb{X} = \{\mathbf{X_0}, \mathbf{X_1}, \mathbf{X_2}, ..., \mathbf{X_n}\}$. A parameter $\overline{u}_k \in \overline{\mathbf{U}}$ is assigned to each $\mathbf{X_k}$. This parameter $\overline{\mathbf{U}}$ can have equally spaced values, but this uniform parameter scheme often does not provide a satisfactory result. The scheme used to calculate $\overline{u}_k$ is called the centripetal model, which almost always creates better shapes. In this model [18],

$$\overline{u}_k = \begin{cases} 0 & \text{if } k = 0 \\ 1 & \text{if } k = n \\ \overline{u}_{k-1} + \frac{\sqrt{|\mathbf{X_k} - \mathbf{X_{k-1}}|}}{\sum_{j=1}^{n} \sqrt{|\mathbf{X_k} - \mathbf{X_{k-1}}|}} & \text{otherwise} \end{cases} . \tag{55}$$

In turn, the knot vector $\mathbf{U}$ is constructed using the following formulae [17]:

$$u_0 = u_1 = ... = u_p = 0, \tag{56}$$

$$u_{n+1} = u_{n+2} = ... = u_{n+p+1} = 1, \tag{57}$$

and

$$u_{j+p} = \frac{1}{p}\sum_{i=j}^{j+p-1} \overline{u}_i \quad j = 1, 2, 3, ..., n-p. \tag{58}$$

Here, the data points $\mathbb{X}$ and the knot vector $\mathbf{U}$ are known. The control points $\mathbb{P}$ are unknown.

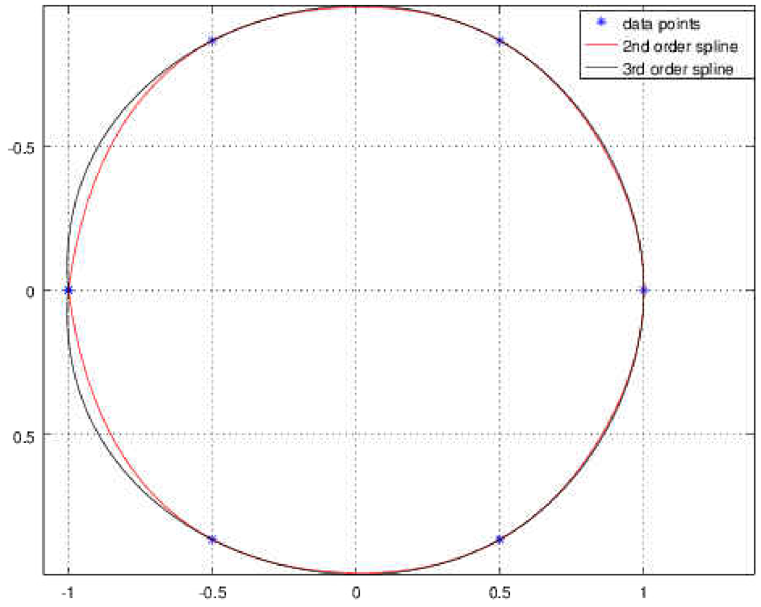


FIG. 18: Two splines constructed from evenly distributed six coordinates of a unit circle. The red spline is a second order spline and the black spline is the third order spline.

The B-spline is a linear combination of its basis functions. Thus, each data point can be represented as a linear combination of the basis functions N and control points. Therefore,

$$\mathbf{X_k} = \sum_{i=0}^{n} N_{i,p}(\overline{u}_k)\mathbf{P_i} \tag{59}$$

where the B-spline basis function N is a recursively defined function. The zeroth order basis function is [17]

$$N_{i,p=0}(\overline{u}) = \begin{cases} 1 & \text{if } u_i \leq \overline{u} \leq u_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{60}$$

and an arbitrary p^{th} order basis function for $p > 0$ is

$$N_{i,p}(\overline{u}) = \frac{\overline{u} - u_i}{u_{i+p} - u_i} N_{i,p-1}(\overline{u}) + \frac{u_{i+p+1} - \overline{u}}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(\overline{u}). \tag{61}$$

Here, the knot vector $\mathbf{U} = (u_0, u_1, ..., u_{n+k+1})$ and the parameter $\overline{u}_k$ for each data point $\mathbf{X_k}$ are known. Hence, the basis functions can be calculated. Only the control points are unknown. Equation 59 can be solved for the control points $\mathbb{P}$. Once the control points and basis functions are known, the spline can be constructed by choosing appropriate $\overline{\mathbf{V}}$ that is similar to $\overline{\mathbf{U}}$ and substituting $\overline{v}_k$ in Equation 59. Here, $\overline{\mathbf{V}}$ is chosen such that the difference between two consecutive elements is very small. Thus, the consecutive interpolated values will be very close to each other.

5.2.4 POST-PROCESSING

The velocity vector for each vertex during iterations is calculated by subtracting its previous position from its current position. This process is applied for each vertex after each iteration, so having n iterations means that each vertex has a series of $n - 1$ vectors. The contour at an arbitrary z position between the two slices is then interpolated by interpolating each vertex's position. A B-spline is then generated for these vertices, and the contour is inserted at the z position. Repeating this process between the slices in small increments of z yields a smooth 3D reconstruction of the region.

5.3 RESULTS AND DISCUSSION

Figure 19 and Figure 20 show the input regions and the reconstructed outputs. The resulting volume is smooth and appears to be a faithful approximation of the input. The results show that the longitudinal resolution of the Paxinos atlas is acceptable, i.e., the reconstructed volume does not display discontinuities. The 1-simplex mesh method is sensitive to initialization. If the mesh is initialized outside the target, the mesh may converge, but it may not yield acceptable interpolation. In order to mitigate this issue, regions in each slice may be aligned beforehand, and the resulting volume can be re-adjusted after its generation. The 1-simplex mesh algorithm is sensitive to the density of vertices and various parameters. Therefore, it is often necessary to reconfigure the parameters. On the other hand, the 1-simplex mesh is adaptable since it allows changing parameters during the deformation.

Selecting an inappropriate window size for the gradient force for a 1-simplex mesh can potentially prevent convergence because there may not be a pixel with a high gradient in any vertex's window. This eliminates the portion of the external force that attracts the mesh towards high gradient pixels. The computation time for the 1-simplex mesh algorithm depends on the number of vertices, the continuity condition, and the vertex-to-boundary distance. The overall size of the image does not affect the computation time. Since the internal and the external forces need to be calculated only for the vertices, the simplex mesh converges quickly.

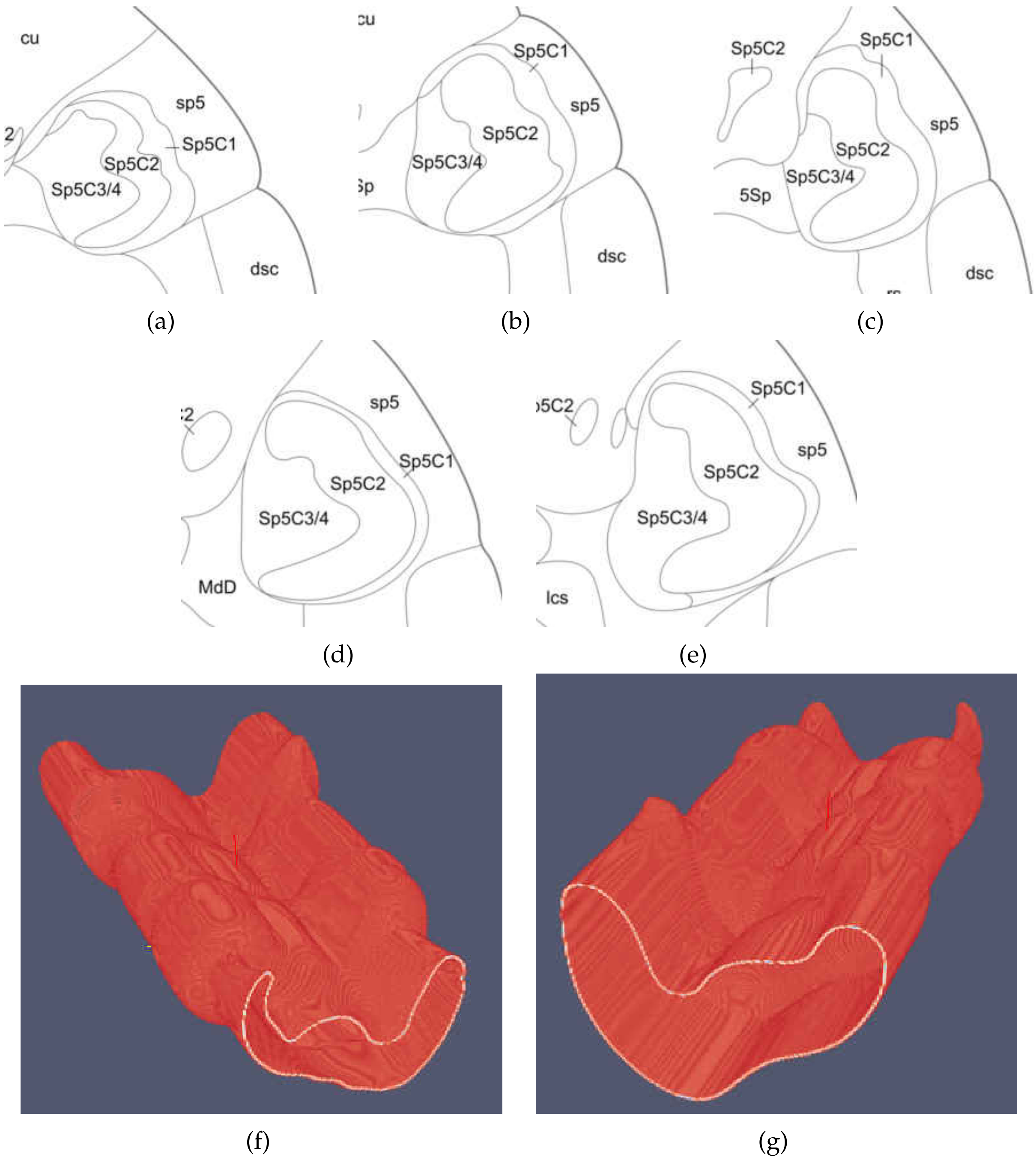
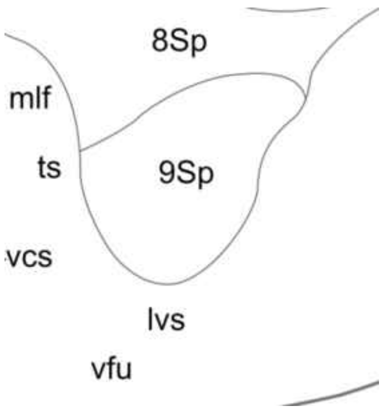
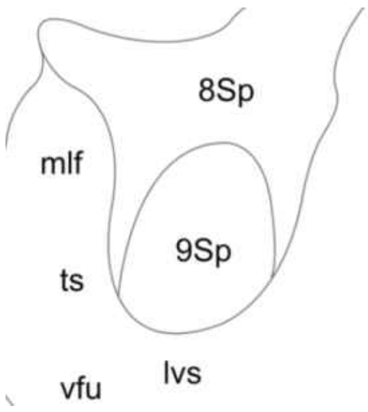


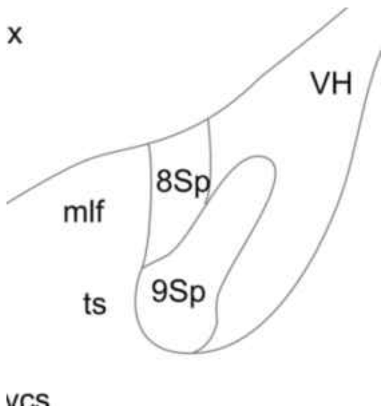
FIG. 19: A reconstructed volume from five slices. Images (a), (b), (c), (d), and (e) show a region labeled Sp5C2 which was interpolated. These images were digitized with the methodology shown in the previous chapter. Then, the methodology of this chapter was applied to produce a 3D reconstruction which is shown in two different perspectives in (f) and (g). The size of each input image is 604 $\times$ 540. The reconstructed volume contains 641 slices.



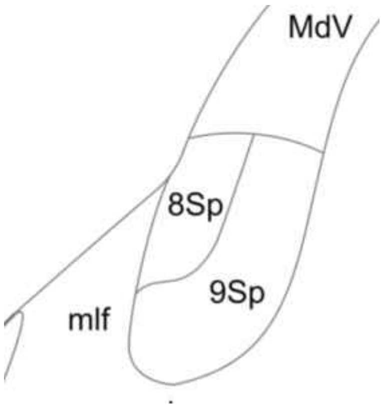
(a)



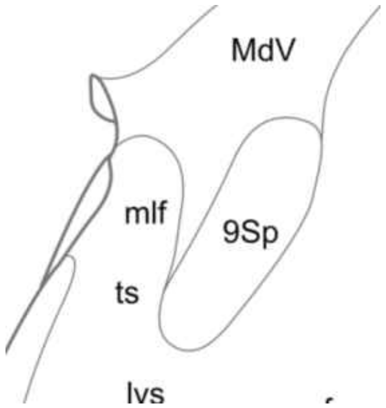
(b)



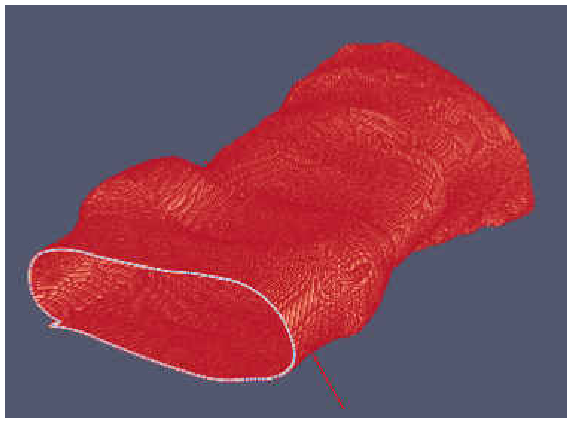
(c)



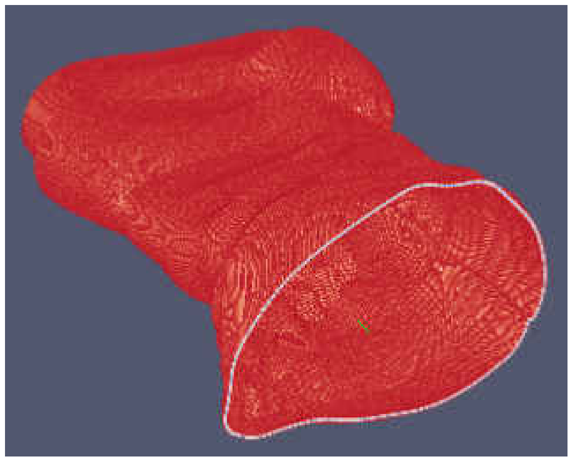
(d)



(e)



(f)



(g)

FIG. 20: A reconstructed volume from five slices. Images (a), (b), (c), (d), and (e) show a region labeled 9Sp which was interpolated. These images were digitized with the methodology shown in the previous chapter. Then, the methodology of this chapter was applied to produce a 3D reconstruction which is shown in two different perspectives in (f) and (g). The size of each input image is 405 × 430. The reconstructed volume contains 371 slices.

CHAPTER 6

CONCLUSION

The brainstem is an important anatomical structure, but there is an insufficiency of digital brainstem atlases that can be used for the atlas-based segmentation approach. In order to fulfill this dire need, the brainstem atlases found in textbooks can be digitized. This thesis research's aim was to create a software tool that semi-automatically digitizes a 2D atlas of the brainstem. Another goal of the thesis research was to devise a methodology that can be used to stack the digitized slices and reconstruct a volume. The 3D reconstruction approach was manually applied; a software tool for semi-automatic reconstruction is planned in the future.

The methodology presented in this thesis is reliable and adaptable. The main contribution of this thesis research is that it lays the foundation for the production of a descriptive 3D atlas of the brainstem, which can be used to generate descriptive patient-specific brainstem atlases. These patient-specific atlases can be used in various neurological applications such as neurosurgery planning and simulation to improve patient outcome. This thesis research has implemented the 2D adaptation of the 1-simplex mesh algorithm which is novel. Another implication of a digital brainstem atlas is that it will allow explicit modeling of the cranial nerves using probabilistic shape priors.

There are a few limitations of the methodology presented in this thesis. A brainstem atlas found in a textbook must have edges without openings for this methodology. The current approach cannot segment open regions. Regions that are open need to be closed before the digitization process. Another limitation is that the axis of symmetry implies perfect symmetry, which is not true. Moreover, experts are needed to correctly digitize the images since these images may have ambiguities. For example, a region may have a different label in different places. During 3D reconstruction, care must be taken for correct interpolation because it is affected by the numerous parameters of the 1-simplex mesh. The values of these parameters often need to be changed during deformation to fine-tune it. The current methodology allows the reconstruction of only one region at a time. Furthermore, the neighboring structures do not affect the shape of the region being interpolated. Finally, the contour interpolation between two slices is affected by only those two slices; it is not affected by slices that are farther away.

Despite its limitation, the presented methodology has several strengths. The digitization of 2D atlases from textbooks is robust as long as the scanned images are within the domain of the methodology. The geodesic active contours do not bleed through a closed boundary. The 1-simplex model is hard to configure correctly but allows fine-tuning of the behavior, which is beneficial for the interpolation. Another strength of this methodology is that it requires minimal supervision because it greatly reduces the complexity and increases the reliability of the methodology.

In the future, a contour deformation model that works on dashed boundaries or a method to treat them will be used. Also, a software tool that stacks and reconstructs the digitized regions semi-automatically will be created. Finally, an anatomist will be consulted to verify the fidelity of the reconstructed structures.

BIBLIOGRAPHY

[1] June Andrews and JA Sethian. "Fast marching methods for the continuous traveling salesman problem". In: *Proceedings of the National Academy of Sciences* 104.4 (2007), pp. 1118–1123.

[2] David W Buchholz. "Oropharyngeal dysphagia due to iatrogenic neurological dysfunction". In: *Dysphagia* 10.4 (1995), pp. 248–254.

[3] Johnny Cappiello, Cesare Piazza, and Piero Nicolai. "The spinal accessory nerve in head and neck surgery". In: *Current opinion in otolaryngology & head and neck surgery* 15.2 (2007), pp. 107–111.

[4] Vicent Caselles, Ron Kimmel, and Guillermo Sapiro. "Geodesic active contours". In: *International journal of computer vision* 22.1 (1997), pp. 61–79.

[5] Tony Chan and Luminita Vese. "An active contour model without edges". In: *Scale-Space Theories in Computer Vision*. Springer, 1999, pp. 141–151.

[6] Ivan Ciric et al. "Complications of transsphenoidal surgery: results of a national survey, review of the literature, and personal experience". In: *Neurosurgery* 40.2 (1997), pp. 225–237.

[7] OpenStax College. *Anatomy & Physiology*. OpenStax College, 2013. ISBN: 1938168135.

[8] *Cranial Nerves*. 2016. URL: `http://www.britannica.com/science/cranial-nerve`.

[9] William R Crum, Thomas Hartkens, and DLG Hill. "Non-rigid image registration: theory and practice". In: *The British Journal of Radiology* (2014).

[10] Herv Delingette. "General object reconstruction based on simplex meshes". In: *International Journal of Computer Vision* 32.2 (1999), pp. 111–146.

[11] Rahul S Desikan et al. "An automated labeling system for subdividing the human cerebral cortex on MRI scans into gyral based regions of interest". In: *Neuroimage* 31.3 (2006), pp. 968–980.

[12] Alejandro F Frangi et al. "Multiscale vessel enhancement filtering". In: *Medical Image Computing and Computer-Assisted InterventationMICCAI98*. Springer, 1998, pp. 130–137.

[13] Eric Grimson and Polina Golland. "Analyzing anatomical structures: Leveraging multiple sources of knowledge". In: *Computer Vision for Biomedical Image Applications*. Springer, 2005, pp. 3–12.

[14] Michael Halle et al. "Multi-modality MRI-based Atlas of the Brain". In: (Nov. 2015).

[15] Hrvoje Kalinic. "Atlas-based image segmentation: A Survey". In: *Department of Electronic Systems and Information Processing, Universiy of Zagreb* (2008).

[16] Mika Kapanen and Mikko Tenhunen. "T1/T2*-weighted MRI provides clinically relevant pseudo-CT density data for the pelvic bones in MRI-only based radiotherapy treatment planning". In: *Acta Oncologica* 52.3 (2013), pp. 612–618.

[17] Tahsin Khajah and Gene J.-W Hou. *Isogeometric analysis for electromagnetism.* Old Dominion University theses : Mechanical engineering: 2015. 2015.

[18] Eugene TY Lee. "Choosing nodes in parametric curve interpolation". In: *Computer-Aided Design* 21.6 (1989), pp. 363–370.

[19] Diane Lingrand. *The marching squares algorithm*. URL: `http://engineering.purdue.edu/~mark/puthesis` (visited on 03/14/2016).

[20] Juergen K Mai. *The Human Brain Atlas of the Human Brain*. URL: `http://www.thehumanbrain.info/brain/brain_slicer.php`.

[21] Juergen K. Mai, George Paxinos, and Thomas Voss. *Atlas of the Human Brain, Third Edition*. Academic Press, 2007. ISBN: 012373603X.

[22] Thomas P. Naidich et al. *Duvernoy's Atlas of the Human Brain Stem and Cerebellum: High-Field MRI, Surface Anatomy, Internal Structure, Vascularization and 3 D Sectional Anatomy*. Springer, 2008. ISBN: 321173970X.

[23] Sara Nasser, Rawan Alkhaldi, and Gregory Vert. "A modified fuzzy k-means clustering using expectation maximization". In: *Fuzzy Systems, 2006 IEEE International Conference on*. IEEE. 2006, pp. 231–235.

[24] Pietro Perona, Takahiro Shiota, and Jitendra Malik. "Anisotropic diffusion". In: *Geometry-driven diffusion in computer vision*. Springer, 1994, pp. 73–92.

[25] Dzung L Pham, Chenyang Xu, and Jerry L Prince. "Current methods in medical image segmentation 1". In: *Annual review of biomedical engineering* 2.1 (2000), pp. 315–337.

[26] William K. Pratt. *Digital Image Processing: PIKS Scientific Inside*. Wiley-Interscience, 2007. ISBN: 0471767778.

[27] P- Ridderheim, Cl von Essen, and B Zetterlund. "Indirect injury to cranial nerves after surgery with Cavitron ultrasonic surgical aspirator (CUSA). Case report". In: *Acta neurochirurgica* 89.1-2 (1987), pp. 84–86.

[28] James A Sethian. "A fast marching level set method for monotonically advancing fronts". In: *Proceedings of the National Academy of Sciences* 93.4 (1996), pp. 1591–1595.

[29] James Albert Sethian. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*. Vol. 3. Cambridge university press, 1999.

[30] Sharmin Sultana et al. "Patient-Specific Cranial Nerve Identification Using a Discrete Deformable Contour Model for Skull Base Neurosurgery Planning and Simulation". In: *4th Workshop on Clinical Image-based Procedures: Translational Research in Medical Imaging* (2015).

[31] *Templates and Atlases included with FSL*. URL: `http://fsl.fmrib.ox.ac.uk/fsl/fslwiki/Atlases` (visited on 03/07/2016).

[32] Ross T Whitaker and Xinwei Xue. "Variable-conductance, level-set curvature for image denoising". In: *Image Processing, 2001. Proceedings. 2001 International Conference on*. Vol. 3. IEEE. 2001, pp. 142–145.

[33] J Michael Wiater and Louis U Bigliani. "Spinal accessory nerve injury." In: *Clinical orthopaedics and related research* 368 (1999), pp. 5–16.

VITA

Nirmal J. Patel
Department of Electrical and Computer Engineering
Old Dominion University
Norfolk, VA 23529

EDUCATION

AUG. 2014 - MAY 2016	Master of Science in COMPUTER ENGINEERING Old Dominion University, Norfolk, Virginia GPA: 3.37/4.0 Thesis: Deformable Contour Models for Digitizing a Printed Brainstem Atlas
AUG. 2010 - MAY 2014	Bachelor of Science in COMPUTER ENGINEERING Old Dominion University, Norfolk, Virginia Minor: Computer Science GPA: 3.32/4.0

PUBLICATIONS

Nirmal Patel, Sharmin Sultana, and Michel A Audette. "Contour Models for Descriptive Patient-Specific Neuro-Anatomical Modeling: Towards a Digital Brainstem Atlas". In: *Recent Advances in Computational Methods and Clinical Applications for Spine Imaging*. Springer, 2015, pp. 199–211.

Nirmal Patel et al. "Application and histology-driven refinement of active contour models to functional region and nerve delineation: towards a digital brainstem atlas". In: *SPIE Medical Imaging*. International Society for Optics and Photonics. 2015, pp. 94150I–94150I.