Summer 8-2012

Zifazah: A Scientific Visualization Language for Tensor Field Visualizations

Haipeng Cai
University of Southern Mississippi

The University of Southern Mississippi

ZIFAZAH: A SCIENTIFIC VISUALIZATION LANGUAGE FOR TENSOR FIELD VISUALIZATIONS

by

Haipeng Cai

A Thesis
Submitted to the Graduate School
of The University of Southern Mississippi
in Partial Fulfillment of the Requirements
for the Degree of Master of Science

Approved:

[redacted]

Dean of the Graduate School

August 2012

ABSTRACT

ZIFAZAH: A SCIENTIFIC VISUALIZATION LANGUAGE FOR TENSOR FIELD VISUALIZATIONS

by Haipeng Cai

August 2012

This thesis presents the design and prototype implementation of a scientific visualization language called Zifazah for composing and exploring 3D visualizations of diffusion tensor magnetic resonance imaging (DT-MRI or DTI) data. Unlike existing tools allowing flexible customization of data visualizations that are programmer-oriented, Zifazah focuses on domain scientists as end users in order to enable them to freely compose visualizations of their scientific data set. Verbal descriptions of end users about how they would build and explore DTI visualizations are analyzed to collect syntax, semantics, and control structures of the language. Zifazah makes use of the initial set of lexical terms and semantical patterns to provide a declarative language in the spirit of intuitive syntax and usage. Along with sample scripts representative of the main language design features, some new DTI visualizations created by end users using the novel language have also been presented.

ACKNOWLEDGMENTS

I would like to first thank my advisor, Dr. Jian Chen, who offered me a comfortable place to learn doing research and valuable academic advice with constant patience and encouragement, without which this work would have not been practical. I am especially grateful to her research guidance that will greatly contribute to my long-term profession in the future.

I would also like to my other thesis committee members, Dr. Alexander P. Auchus, Dr. David H. Laidlaw, and Dr. Chaoyang Zhang, for their kind advice and generous support of resources that were essential for this work to be undertaken. Their invariable help in all possible aspects has been indispensable for the smooth progress throughout this work and even my whole graduate study period here at The University of Southern Mississippi.

Additionally, I would express my gratitude towards medical experts in DTI and non-expert participants for their participation in our experimental studies from which the language design features and principles are summarized. Their comments on the language when using it as end users are appreciated as well.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

Figure

CHAPTER I

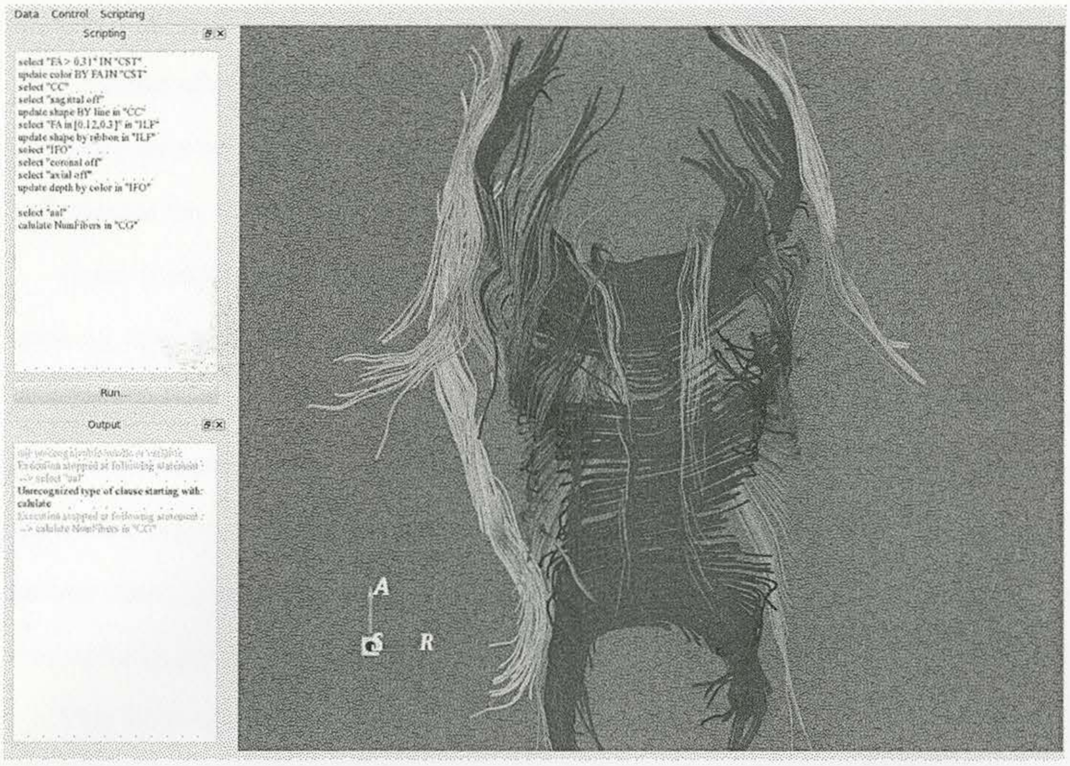
INTRODUCTION

Visualization tools often support user customization, which allows changes of the visualization so as to help users gain better understanding of the underlying data, thus facilitating knowledge discoveries about the data that would be hard to achieve otherwise. However, the support of user creativity is usually constrained by the limits of predefined options or functionalities for the customizations. An effective way to address these constraints is to offer users a programming environment in which they can freely compose towards desirable visualizations of their data through a visualization language. While such languages have been proposed and successful in the information visualization (InfoVis) community [17, 25, 12], there is a lack of end-user visualization language for 3D scientific visualization (SciVis). Based on our many discussions with domain users, we have recognized that domain scientists want a visualization of their own data to be designed and built by themselves. Now that the success of visualization languages for InfoVis is probably attributed to their capabilities of empowering users to design their own visualizations, what if domain scientists have a visualization tool that is powerful but easy to maneuver so that they can fully control the design elements and visual components to create whatever visualization they really have in mind?

A recently advanced MRI technique, diffusion tensor imaging (DTI), has proven advantageous over other imaging techniques in that it enables *in vivo* investigation of biological tissues and, through three-dimensional (3D) tractography [8], explorations of the distribution and connectivity of neural pathways in fibrous tissues like brain white matter and muscles. Further, as one way to visualize DTI data, 3D visualization of the streamline data model derived from the tractography can illustrate the connectivity of fiber tracts and structures of anatomy, and therefore provides a powerful means to assist neuroscientists in clinical diagnosis and neurosurgical planning.

I proposed a visualization language as the first tool of this kind for DTI visualizations because DTI is complex enough to stimulate a design that would be useful for simpler and similar visualization problems such as that of flow visualization. Although mainly driven by neurologists' need for conducting their clinical tasks with DTI visualizations, this language design would also be reusable in a broader range in 3D SciVis. Motivated by the needs of spatial explorations in 3D scientific visualizations because of the spatial constraints within the data, the present language is particularly useful in empowering domain scientists to build 3D visualizations that best meet their specific needs. Furthermore, the language can facilitate domain scientists' effective use and exploration of the visualizations as well because it allows them to customize essential elements of visualization with the maximal flexibility by applying their best understanding of the domain data to the visualization composition process. Illuminated by Bertin's *Semiology of Graphics* [9], I designed the language to allow users to compose symbols in 3D visualizations, including visual encoding methods and other causes that affect visualization task performance.

To capture the design elements of the language, I have conducted experimental studies with domain scientists in DTI who are expected users of this language, and I have summarized design principles for the language out of their descriptions of visualization making and exploration, from which basic lexical terms such as verbs, prepositions, and conjunctions were also reduced. With these principles and language elements, I have developed a language prototype, named Zifazah, as an initial implementation of the visualization language I am proposing. To target non-programmer users like neural medical doctors, Zifazah is designed to be a high-level declarative language. Also, for an easier usage for users without any programming skills and experience, Zifazah is currently developed as a procedural language that contains only an intuitive type of control structure, i.e. the sequential struc-



```
select "FA > 0.31" IN "CST"
update color BY FA IN "CST"
select "CC"
select "sagittal off"
update shape BY line in "CC"
select "FA in [0.12,0.3]" in "ILF"
update shape by ribbon in "ILF"
select "IFO"
select "coronal off"
select "axial off"
update depth by color in "IFO"

select "aal"
calulate NumFibers in "CG"
```

```
[illegible]
Execution stopped at following statement :
--> select "aal"
Unrecognized type of clause starting with:
calulate
Execution stopped at following statement :
--> calulate NumFibers in "CG"
```

Figure 1. A screenshot of the Zifazah programming interface, consisting of a programming text board (upper left), a simple debugging output window (bottom left) and the visualization view (right).

ture. As such, users can write Zifazah scripts as simply as if they verbally describe the process of authoring visualizations in sequence. Figure 1 gives an outlook of the Zifazah programming interface.

The following usage scenarios briefly show the utility of Zifazah. In the first scenario, an end-user first loads a whole DTI model and then programs to vary tube size in the default streamtube visualization by fractional anisotropy (FA) and tube color by fiber distance to the viewing point in a specific brain structure. In the end, the user can change the streamtube representation of another brain region to ribbons. In the second scenario, a user filters fibers according to an estimate of linear anisotropy (LA) threshold and then gradually adjusts the threshold until satisfied. The user then further cuts off the selected fibers outside a target brain region through spatial commands with precisely calculated movements and thus reaches the tubes of interest. As the final example, a user can get the size of a brain structure in terms of the number of fibers, average FA in a brain region, and other common DTI metrics after reaching the target fibers. In each of these scenarios, the user achieves each step by writing a declarative program statement in the script editor, and the results are reflected in the visualization view (see Chapter V).

Apart from a visualization language that helps domain scientists build DTI visualizations by themselves to exactly meet their specific needs with the visualizations, this work also contributes several design features to general DTI visualizations including: (1) visual symbolic mapping based on color, size and shape, as is new for scientific visualizations; (2) lexical representations of spatial relationships for 3D object visualization and manipulations; and (3) data encoding flexibility built upon the migration of Bertin's semiological principles to scientific visualizations.

This language will be the first of its kind. The following snippet gives a quick view of how a Zifazah program looks. This script describes an exploratory process of an end user with the streamtube model [33] of a human brain DTI data set, in which different fiber bundles are filtered according to threshold of DTI metrics and customized with various visual encoding methods.

```
LOAD "/tmp/allfb_tagged.data"
SELECT "CC"
SELECT "FA in [0.2,0.25]" IN "IFO"
UPDATE color BY FA IN "CC"
SELECT "LA > 0.35" IN "CST"
UPDATE shape BY line IN "CC"
UPDATE shape BY tube IN "IFO"
UPDATE size BY FA WITH 0.1,20 IN "IFO"
```

As shown in this example, a Zifazah program is essentially an intuitive sequence of steps, each carrying out a single visual transformation of data. Although the script is written in a textual form as in a traditional computer programming language, each of the statements is more like a high-level command. Also, there is no logic structure other than the sequential one in Zifazah, which makes this language fairly easy to learn and use for end users in medical field.

The rest of this thesis is organized as follows. I first give general background and discuss related work in Chapter II. In Chapter III I detail design principles and supporting language elements and then brief implementation issues in Chapter IV. Chapter V expands the details of the three usage scenarios introduced above and gives the corresponding Zifazah scripts and running results. I discuss other design considerations of the language and design features to be fully implemented that are integral to the overall language design in Chapter VI before finally concluding the thesis in Chapter VII.

CHAPTER II

BACKGROUND AND RELATED WORK

Visualization of DTI Models

In general, DTI data sets can be visualized using various approaches ranging from direct volume rendering of tensor field [21] to geometry rendering of the fiber model derived from tensor field. With geometry rendering, DTI fibers are usually depicted as streamlines [20], streamtubes and streamsurfaces [33]. In order to explore 3D visualizations of the fiber geometries, 2D embedding and multiple coordinated views [19], along with various interactive techniques [10, 29], have been employed.

Many other powerful tools have also been developed for exploring DTI visualizations [6, 19, 32, 7, 10]. However, due to the data complexity, domain users' needs for performing their various tasks in daily practice have not yet been fully satisfied by using those tools. To give users more flexibility, some of the visualization tools are made highly configurable by allowing a wide range of settings [28, 31, 29]. Nevertheless, it is still challenging to design a thoroughly effective visualization tool to meet all the needs of users. For instance, MedINRIA [31] provides rich predefined functionalities yet does not allow end-user composition of visualizations. For another example, Slicer [28] is made very powerful via the integration of various function models, working like a data processing and visualization tool suite, but it is not designed for users to create their own visualizations in order to satisfy their typical domain tasks. Also, although sometimes able to meet specific requirements, higher flexibility of a visualization tool may even make the tool more complex to use for domain users [22].

Composable Visualizations

Since pioneering the automatic generation of graphic representation [24], Mackinlay's work has been extended lately into a visual analysis system armed with a set of interface commands and defaults representing the best practices of graphical design [25], upon which a commercial software called Tableau was developed. In his work, the generation of visualizations was automated thanks to the application of a series of design rules and made adaptable to users with a wide range of design expertise via constrained flexibilities by those design rules. With Zifazah, I also intend to provide an environment in which end users can flexibly build their own visualizations like Tableau. However, instead of targeting visual analysis in the context of two-dimensional (2D) information visualization, Zifazah primarily aims at end-user visualization making and exploration with 3D scientific data such as DTI. Also, compared to the visual specifications in Tableau, like those in its predecessor Polaris [30], textual programming is the main means for end users to interact with visualizations of interest in Zifazah. Similar to Polaris in terms of using visual operations to build visualizations, dynamic queries [29] aims to support retrieving DTI fibers instead of querying relational database in Polaris.

As a toolkit, Protovis gives users high-level usage flexibility, even programmability, yet imposes constraints upon user programs through implicit rules to produce effective visualizations [12]. This tool has been evolved into its descendant named D3 [13] for a better support of animation and interaction. Zifazah shares some Protovis features like addressing non-programmer audiences and having concise and easy-to-learn grammar. However, different from Protovis that uses simple graphical primitives called marks to construct information visualizations and mainly targets web and interaction designers, Zifazah targets neuroscientists instead and enables them not only to flexibly construct, but also to effectively explore, in the context of scientific visualizations exemplified by that of DTI data.

Visualization Languages

Processing [17] is more a full-blown programming language and environment than a traditional visualization tool. Built with the full Java programming language facilities, Processing integrates the underlying visual design rules to help users build beautiful yet informative visualizations with the support of interaction design. Although developed to be accessible for new users and non-programmers, Processing is more oriented to users with a certain level of programming skills and might still be challenging for domain users like neuroscientists who are the primary audience I address. A sister visual programming language of Processing, Processing.js [2] also targets web developers. By contrast, Zifazah is distinct in that it empowers end users to explore scientific data through intuitive syntax within a sequential structure rather than offering a full set of programming features in a traditional computer language as Processing does. Like Zifazah, Impure [1] is also a programming language for data visualizations that targets non-programmers. Although supporting various data sources, this completely visual language is developed for information design rather than for scientific visualizations.

Although a natural language like WordsEye [11] for visualizations might be appealing to ordinary users without any programming knowledge, I do not attempt the entirely descriptive nature for Zifazah as WordsEye did at the current stage. In terms of lexical and syntax design, Zifazah is similar to Yahoo!'s Pig Latin [14], which is a new data processing language associated with the Yahoo! Pig data handling environment that balances between a declarative language and a low-level procedural one. The language supports data filtering and grouping with parallelism by its map-reduce programming capability. However, this language does not handle visualizations or any form of graphical representations but focusing on ad-hoc data analysis. Also, Zifazah is set apart from Pig Latin in the target audience again, since the latter mainly serves software engineers.

The Protovis specification language [18] is a declarative domain-specific language (DSL) that supports specification of interactive information visualizations with animated transitions, providing an approach to composing custom views of data using graphical primitives called marks that are able encode data by dynamic properties, which is similar to the mapping of object properties to graphical representations in another InfoVis language presented by Lucas and Shieber [23]. To some extent, both languages are comparable to Microsoft's ongoing project Vedea, aiming at a new visualization language [4] in terms of syntactic design and programming style, although its design goals are closer to that of Processing.

Also in the InfoVis domain, Trevil [3] is a programming language based on its predecessor Trevis [5], a framework used for context tree visualization and analysis. It supports composing visualizations but is dedicated to the visualization of unordered trees. Peterson et al. discuss another specific-purpose language [27] that serves the composition of visualizations of mathematical concepts like those in basic algebra and calculus.

Recently, Metoyer et al. [26] report from an exploratory study a set of design implications for the design of visualization languages and toolkits. More specifically, their findings inform visualization language design through the way end users describe visualizations and their inclination to use ambiguous and relative, instead of definite and absolute, terms that can be refined later via a feedback loop provided by the language. Their findings also disclose that end users tend to express in generally high-level semantics. During the design of the present language, I have benefited from these findings and actually have reflected them in the development of Zifazah.

CHAPTER III

LANGUAGE DESIGN

In this chapter, I first brief end-user descriptions on composing DTI visualizations from which design requirements and principles, as follow a summary of the language symbols and description of Zifazah data model, are extracted and motivated respectively. The development of Zifazah is driven by end-user requirements with DTI visualizations and the design principles are embodied in the language features of Zifazah. After each of the language features, Zifazah language elements that meet the feature are detailed, including related lexical terms and syntactic patterns. Instead of describing the implementation techniques, which are highlighted in Chapter IV, this chapter emphasizes how the design principles and language elements address the end-user requirements.

Design Motivations

The design of Zifazah is motivated by the needs of typical end users I target for composing DTI visualizations by themselves, which can be derived from their verbal descriptions about visualizations they would desire in my many interviews and discussions with them. I report just a few representative example comments from them on visualizations produced beforehand by computer scientists.

The participants include neurologists and neural physicians, both conducting clinical diagnosis with DTI data visualizations. In a typical interview, participants are presented visualizations of the same DTI brain data set composed differently by manipulating various visual elements and the compositions are done by computer scientists, who then revise the composing process according to the comments of participants. As a result, either the unsatisfactory visualizations are finally modified to meet participants' requirements or suggestions for achieving the desirable visualizations are received if the current tool is not capable of composing the desirable ones.

As an example, multiple visual mappings of depth values to size and color does not enhance the visualization of DTI model as expected. Surprisingly, "*...it is misleading to have the different size*" while color has already been used to discern depth, and "*...would rather have it stay the same size as I spin it around.*" However, visual mapping of depth to color is still preferable since "*...I like it with the color. That is what I need to look at.*" Nevertheless, the composed coloring scheme in which color is mapped by depth might also be useful "*...if determined by the principal eigen values.*" And "*...I think that color is a good idea but prefer color by orientation...*" etc.

There is also a call for doing analysis in the composing environment ("*...Also, one thing for fibers, I am looking at for analysis purposes*"). Emphatically, both classes of participants unanimously "*want to do the analysis over here on the same page, that will be good, too, rather than opening it up again and trying to do it... It will all come together. It will all be integrated into one....*"

These observations all suggest that domain users, exemplified by the typical end users of Zifazah above, potentially ask for a high-level tool, allowing them to define a self-controlled sequence of operations that works towards a visualization precisely meeting their own specific needs. By allowing users to compose with well-designed visual elements, a programming environment can provide the capabilities for neurologists to create their own visualizations, by which the present work is justified.

Furthermore, the work with Zifazah is substantially grounded upon the semiology of the graphic sign-system and especially the taxonomy about the properties and characteristics of retinal variables [9] in terms of the syntax and semantics design for the scientific visualization language. Zifazah incorporates a subset of the properties and characteristics that are most relevant, according to neurologists' verbal descriptions about DTI visualizations, to the language structure and content: size variation, color variation, and shape variation. For one thing, corresponding syntax terms are built into the language core as basic symbols. For another, semantics associated with these terms are designed to support composing DTI visualizations with respect to these retinal variables by allowing free manipulations of the attributes of related variables. While the semiology and taxonomy is originally formulated to guide the design of 2D graphical representations, I extend them into the 3D graphical environment and employ them in the case of DTI visualizations.

It is fairly noteworthy, and common as well, in participants' verbal descriptions that spatial terms are frequently used and most of the terms related to spatial locations are relative besides those measured in precise units. That Zifazah is designed to be a spatial language is exactly in response to the concerns of my target end users with the spatial relationship of data components in the scientific data model being visualized. The participants' descriptions are also in accordance with the fact that spatial constraint is a defining data characteristic of scientific visualization. Consequently, Zifazah includes a set of syntactic and semantic supports for spatial operations in order to meet end-user needs for composing 3D scientific visualizations like that of DTI models.

Intending to be an initiative of an end-user programming approach to scientific visualizations, Zifazah is designed to support an environment in which domain scientists as end users can compose highly customizable visualizations reflecting their thinking process with the graphical representations of their data set. Since Zifazah is designed for DTI visualizations, the language design primarily deals with DTI data. In this context, language elements of the present Zifazah are derived from experimental study with neuroscientists using diffusion MRI data models. As a matter of fact, the symbols and syntax of the current version of Zifazah are extracted from verbal descriptions by neurologists using DTI about how they would create and explore DTI visualizations. As often referred to as end users, neuroscientists, neurologists and other medical experts who conduct clinical practice with DTI data and its visualizations are the primary audience the Zifazah language targets.

Language Symbols

Table 1. Zifazah Language Symbols and Keywords

Verbs	LOAD, SELECT, LOCATE, UPDATE, CALCULATE
Prepositions	IN, OUT
Conjunctives	BY, WITH
Operators	[], <, <=, >, >=, ==, =, +, −
built-in routines	AvgFA, AvgLA, NumFiber
Constants	shape, color, size, depth, FA, LA, sagittal, axial, coronal, CC, CST, CG, IFO, ILF, DEFAULT, RESET

The core content of Zifazah itself is a simple set of language symbols and keywords. End-user actions intended with DTI visualizations are triggered through five key verbs that are all complete words in natural English. Prepositions are used for targeting scope of data of interest and conjunctives for connecting statement terms. All operators used in Zifazah are exactly the same as those used in elementary math. Specifically, [] serves as range operator here for giving a numerical bound that is used in conditional expressions and + and − are relative (increment and decrement) operators rather than serving arithmetical operations (addition and subtraction). Several built-in routines are provided in Zifazah for simple data statistics and analysis in DTI visualizations: *AvgFA* and *AvgLA* calculates the average FA and average LA of a scope of fibers respectively, and *NumFiber* gives the number of fibers in a fiber bundle. Among the reversed Zifazah constants, the aforementioned five major fiber bundles in human brain model are included.

In these language symbols, all verbs and prepositions are directly picked up from my neurologist collaborators' common descriptions of visualization composition and exploration in natural language. Fiber bundle constants(CC:corpus callosum, CST:corticospinal tracts, CG:cingulum, ILF:inferior longitudinal fasciculus, and IFO:inferior frontal occipital fasciculus) are also suggested by them and operators, built-in routines and other constants are reduced from our requirement analysis of their verbal descriptions. As shown in Table 1, current Zifazah implementation contains a small set of symbols. However, the language has been designed to be scalable to increase in each type of symbol listed in terms of implementation techniques.

Data Model and Input

For the design goals with target end users, Zifazah does not have any specific data types and does not deal with any low-level data processing either. Instead, Zifazah focuses on visual transformations in 3D visualization. As previous examples disclose, I have used a classified geometrical data model derived from DTI volumes, in which fibers are clustered in terms of brain anatomy. In the present data model as input to Zifazah, each fiber has been manually tagged with anatomical cluster identity as one of the five major bundles. Currently, Zifazah's ability to recognize the constants for the major anatomical bundles depends on these cluster tags in the structure of the data model input. However, the language design is not restricted to handling only clustered data. Actually, Zifazah is freely adaptable to an unclustered data model, although data target specification with the major bundle constants will be processed as the whole model then (see Figure 6 and the source script). Nevertheless, Zifazah's capability of spatial operations empowers users to explore regions of interest (ROIs) in the unclustered data models.

In a Zifazah program, the first step is to indicate the source of data model by giving the name of a data file. As an example, a Zifazah data input statement is written as:

```
normalBrain = LOAD "data/normalS1.dat"
```

where the LOAD command parses the input file and creates data structures that fully describe the data model, including identification of the cluster tags. This input specification statement can also update current data model at the beginning of the visualization pipeline if it is not the first step in a Zifazah script. The evaluation is optional and, when provided, saves the result to a variable (*normalBrain* here) for later reference. This is not used in the current version of Zifazah but is required for exploring multiple data sets concurrently (see Chapter VI).

Task-driven Language

The language design of Zifazah is originally driven by the visualization tasks that domain users need to perform in their ordinary clinical practices. Among others, some of their typical tasks are (1) checking integrity of neural structures of a brain as a whole; (2) examining fiber orientation in a ROI or fiber connectivity across ROIs; (3) comparing fiber bundle sizes between brain regions; (4) tracing the variation of DTI quantities such as FA along a group of fibers; and (5) picking particular fibers according to a quantitative threshold, etc.

When using DTI visualizations, not only looking for the whole data model, neurologists are also inclined to concentrate on regional details. In the case of brain DTI visualizations, they often narrow down the view scope toward a relatively large anatomical area in the first place and then dive into a specific ROI. In other words, they tend to pay more attention to ROIs than to the whole brain. More specifically, in the visualizations where neural pathways are depicted as streamtubes, the ROIs are usually clusters of fiber tracts called fiber bundles. For instance, at the beginning of a visualization exploration, one of the neurologist collaborators intends to look into frontal lobe fibers within the intersection of two fiber bundles, CST and CC, and ignores all other regions of the model. Further, suspicious of fibers with average FA under 0.5 for a cerebral disease with which the brain is probably afflicted, the user goes on to examine exactly the suspect fibers. Later on, the user focuses on the small fiber region to see how it differs from typical ones, in terms of orientation and DTI metrics, for instance.

Zifazah is designed as a task-driven language to support this requirement process through high-level primitives such as SELECT and common arithmetical conditional operators including a range operator *in*. Zifazah is mainly featured with facilities for step-by-step data filtering with these primitives. For example, suppose the user above is to explore the fibers of interest, he can write in Zifazah as follows:

```
SELECT "FA < 0.5" IN "CST"
SELECT "FA < 0.4" IN "CC"
```

As the result, fibers in both specific bundles with average FA under 0.5 will be highlighted to help users focus on the local data being explored. On top of this, the user can customize the visualization of the filtered fibers through various visual encoding methods using the UPDATE syntax. This is particularly useful when he wants to keep the data already reached in focus before moving to explore other relevant local data in order to add more fibers into his focus area, or when he simply seeks for a more legible visualization of the data first reached. The instance below, following the same example, illustrates how a better depth perception achieved by a type of depth encoding, together with a differentiating shape encoding, are added up to the two selected fiber bundles respectively.

```
SELECT "FA < 0.5" IN "CST"
SELECT "FA < 0.4" IN "CC"
UPDATE depth BY color IN "CST"
UPDATE shape BY ribbon IN "CC"
```

This simple sequence of commands help users locate desirable fiber tracts with high accuracy while allowing flexible customization upon current visualizations. With this language, users compose intuitive steps to finish tasks that are difficult to achieve by visual interactions. In this case, tracts of interest (TOIs) are first focused and then further differentiated for more effective exploration through improved legibility. In general, Zifazah's design emphasizes this task-driven process of visualization exploration, which fits the thinking process of end users with the present visualizations. Figure 2 shows the resulting visualization.

Filtering data in order to reach an ROI is an operation frequently used during our neurologist collaborators' explorations in DTI visualizations. Zifazah offers two commands

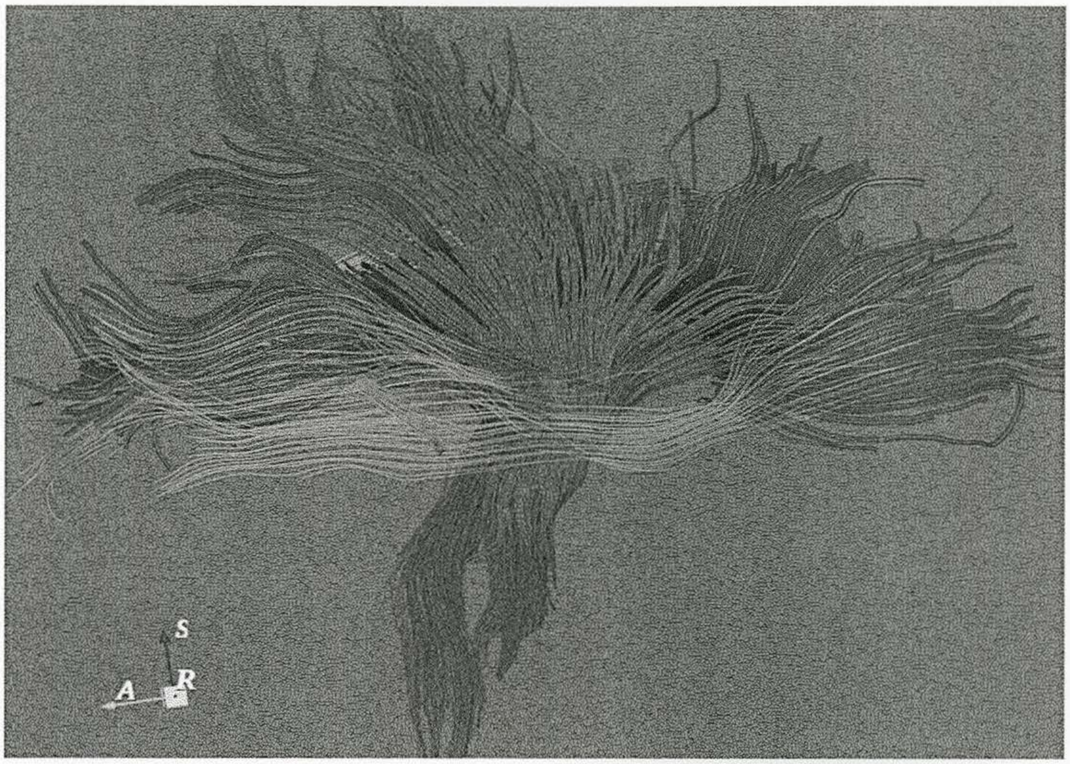


Figure 2. Illustration of the task-driven design of Zifazah.

for data filtering: SELECT and LOCATE. The data filtering syntax pattern in Zifazah is as follows:

```
SELECT condition|spatialOperation IN|OUT target
result = LOCATE condition IN|OUT target
```

With similar functionality, these two commands have different semantics: SELECT executes filtering in an immediate mode by highlighting target fibers while LOCATE performs an offline filtering operation, retrieving target fibers and sending the result to a variable without causing any change in the present visualization. Also, SELECT provides relative spatial operations through moving anatomical cutting planes. In fact, it is tempting to combine these two commands into one while differentiating the two semantics (by recognizing the presence of variable evaluation and taking spatial operations as an alternative to the *condition* term). However, I still keep these two commands separate based on end-user comments asking for a more straightforward understanding of the semantics and easier memory of language usage. For example,

```
SELECT "LA <= 0.72" IN "ALL"

partialILF = LOCATE "FA in [0.5,0.55]" OUT "ILF"
```

The SELECT statement will filter fibers in the whole DTI model with average anisotropy greater than 0.72 (by putting them in the contextual background) and highlight all other fibers. In comparison, the LOCATE statement will not update the visualization but pick up fibers outside the ILF bundle having average FA value in the specified range. Note that when no specific data encoding applied, different colors will be applied to ROI fibers in different major bundles in Zifazah for discerning one ROI from another when there is more than one highlighted. Also, filtered fibers will still be in semi-transparency as the contextual background rather than being removed from the visualization.

Spatial Exploration

One of my main design goals with Zifazah is to provide a language with which users are able to operate spatial structures. I found that my neurologist collaborators tend to frequently use spatial terms such as ”para-sagittal”, ”in”, ”out”, ”mid-axial” and ”near coronal”, etc. in their descriptions about DTI visualizations in the 3D space. They also use a set of other general spatial terms including ”above”, ”under”, ”on top of”, ”across” and ”between”, etc. like what Metoyer et al. found [26] and more domain-specific ones such as ”frontal”, ”posterior” and ”dorsal”, etc. At present, Zifazah contains only a subset rather than all of these spatial terms.

In such a 3D data model as that from DTI, spatial relationships between data components are one of the essential characters, which are actually typical of 3D scientific data in general. Accordingly, composing a DTI visualization necessitates the capability of using spatial operators with domain conventional terms in order to describe the process of visualization authoring. In response, Zifazah supports spatial operations through two approaches combined. First of all, three visible cutting planes that help guide in the three conventional anatomical views, namely the axial, coronal and sagittal view respectively, are integrated in the visualization view (see Figure 1). Then, flexible manipulating operations upon the three planes are built into the Zifazah spatial syntax definitions. This enables end users to navigate in the dense 3D data model with a highly precise filtering capability exactly as they examine a brain model in clinical practice.

For instance, suppose the streamtube representation of a DTI model being programmed is derived using unit seeding resolution from DTI volumes with a size of $256 \times 256 \times 31$ captured at a voxel resolution of $0.9375mm \times 0.9375mm \times 4.52mm$, and suppose both the axial and coronal planes are located at their initial position so that nothing is cut along these two views. In order to examine suspect anomaly in the brain region of the occipital lobe, a medical doctor attempts to filter the data model so that approximately only this region will be kept. For this task, the corresponding Zifazah script can be written as:

```
SELECT "coronal +159.25"
SELECT "axial -27.5"
SELECT "sagittal +183.2"
```

Similarly, relative movements can be imposed on the sagittal plane as well. These simple relative operators included in Zifazah in support of spatial exploration are also informed by the design implications found before [26], although they mainly come from user requirements of performing DTI visualization tasks pertaining to spatial operations. Figure 3 shows the resulting visualization.

Data Encoding Flexibility

According to Bertin's semiotic taxonomy [9], graphically encoding data with key visual elements such as color, size, and shape plays a critical role in the legibility of 2D graphical representations. In 3D visualizations, occlusion effect as an import factor in depth percep-

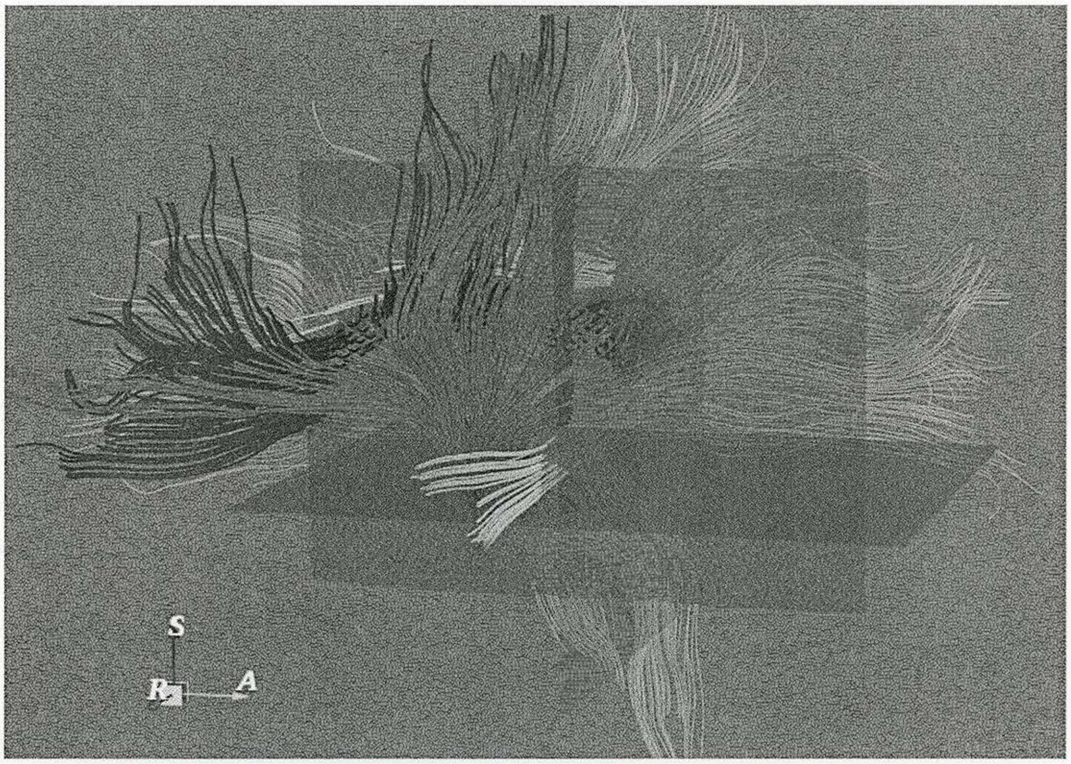


Figure 3. Illustration of the design of Zifazah as a spatial language.

tion has a detrimental impact on the overall legibility, and depth cues (DC) are an ordinal dimension in the design space of 3D occlusion management for visualization [16].

We have combined both aspects in our Zifazah language design: symbolic mapping of color, size, and shape for 2D graphical legibility enhancement and depth encoding, also via common visual elements such as color, size, value (amount of ink), and transparency, as depth cues for occlusion reduction in the 3D environment. As already shown in the previous example scripts, Zifazah allows end users to freely customize DTI visualizations using either a single data encoding scheme alone or compound encoding scheme by flexibly combining multiple encoding methods. The latter leads to a mixed visualization as illustrated in Figure 1.

In their composing or exploratory process with DTI visualizations, users often attempt to examine more than one data focus simultaneously and would like to differentiate one focused ROI from others so that they will not get lost themselves within the multiple ROIs.

There are also other occasions under which the users have difficulty in navigating along the depth dimension even in a single ROI. The data encoding flexibility in Zifazah is driven by both of the two user attempts. For an example, suppose a user has composed the streamtube visualization of a brain DTI data set with default data encoding (uniform size, color, and shape without depth cues) and now wants the overall encoding scheme to be different across fiber bundles. In order to achieve this effect, an example Zifazah snippet can be written as follows:

```
SELECT "ALL"
UPDATE shape BY LINE IN "CST"
UPDATE size BY FA IN "CG"
UPDATE color BY FA IN "IFO"
UPDATE depth BY transparency IN "CG"
UPDATE depth BY value IN "CC" WITH 0.2,0.8
UPDATE depth BY color IN "ILF"
```

Then, in the resulting visualization, each of the five major bundles will be visually disparate from others since all these bundles are encoded differently. Figure 4 shows the resulting visualization. Oftentimes, once one ROI or more is filtered out, it is also necessary to

Table 2. Combinatorial Rules of Constants in UPDATE Statement of Current Zifazah Implementation

*var*1	*var*2	*parameters*
shape	line, tube, ribbon	N/A
color	FA, LA	N/A
size	FA, LA	minimal,scale
depth	size,color,value, transparency	lower,upper
DEFAULT	N/A	N/A
RESET	N/A	N/A

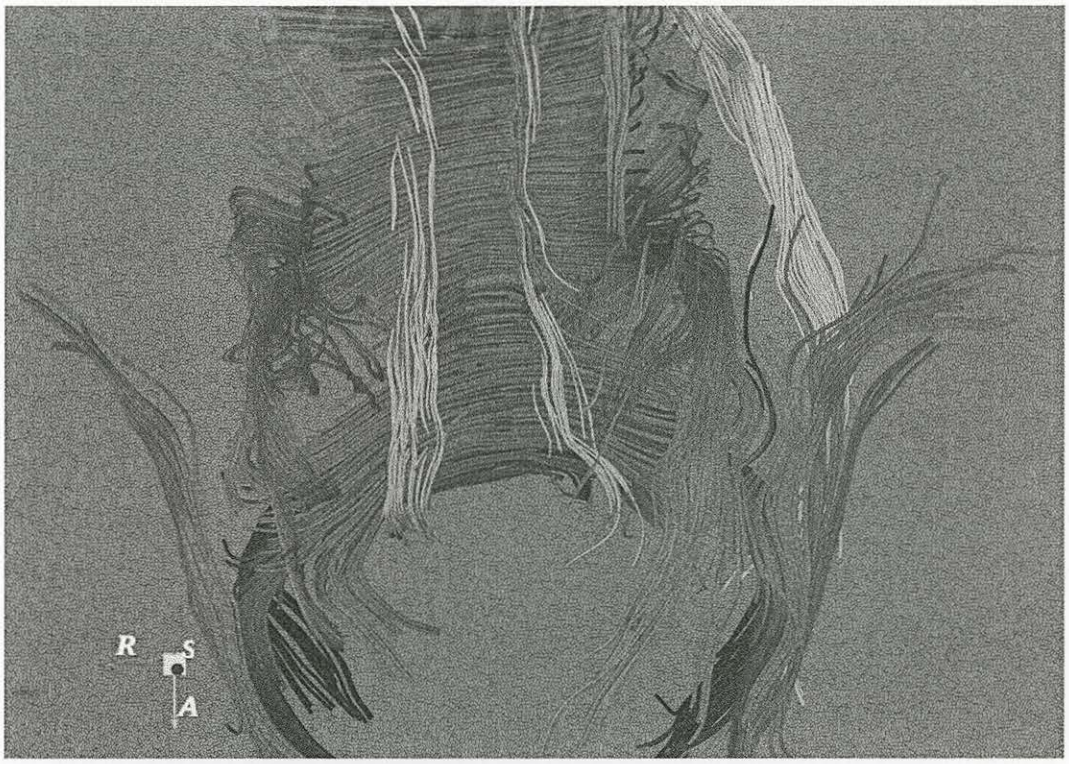


Figure 4. Flexible data encoding built in the design of Zifazah.

examine the selected fibers more carefully. For this purpose, Zifazah allows users to impose various data encoding schemes upon data targets. Such visualization customization is done by the UPDATE command, which always works in an immediate mode and updates the current visualization after execution. The general UPDATE syntax pattern is:

```
UPDATE var1 BY var2 WITH para1,...,paraN IN|OUT target
```

where *var*1 indicates an attribute, such as shape, color, size, depth, etc., of current visualization to be modified, and *var*2 gives how the actual updating operation is to be performed in terms of its relation to *var*1. The parameter list ending the statement presents extra information that the updating requires, as is specific to a particular data encoding operation. Like the target specification (optional with all commands as stated before), the BY clause and WITH clause are both optional. Table 2 lists all possible combinations of *var*1, *var*2 and associated parameter list already developed in present Zifazah. In the table, "lower,upper" gives the bound of depth mapping and "minimal,scale" indicates the minimum and the scale of variation in size encoding. DEFAULT and RESET, when going with the verb UPDATE, act as a command for revoking all data filtering and data encoding operations respectively. The following script shows how to inspect the change of FA along fibers in a ROI by mapping FA value to tube size, which results in a more intuitive perception of the FA variation in that ROI.

```
UPDATE RESET

partialILF = LOCATE "FA in [0.5,0.55]" OUT "ILF"

UPDATE size BY FA IN "partialILF"
```

Flat Control Structure

Another main design goal with Zifazah is to provide a declarative language environment for domain end users who have neither programming skills and experience, nor basic understanding of computer program structures. Consequently, we purposely eliminate the conditional and iteration structures from the language design of Zifazah and only keep the most intuitive one, i.e. the sequential structure, since this simple structure is much more intuitive than the other two. This features Zifazah with a flat control structure that is essential for achieving the design goal. Meanwhile, Zifazah uses high-level semantics to overcome its weakness in expressing user task requirements for lack of these two missed control structures through two approaches addressing the requirements for them.

First, requirement for an iteration structure usually stems from the needs to operate on multiple targets. In Zifazah, operation target is a common term in all syntax patterns to indicate the scope of data to focus on. I address this requirement through enumeration and target term defaults in Zifazah syntax patterns. On the one hand, with enumeration, end users simply list all targets in the target term to avoid iteration. For example, suppose a user intends to select three bundles and then to change size encoding for two of them; his Zifazah script can include:

```
SELECT "CST,CC,CG"
UPDATE size BY FA IN "CST,CG"
```

As such, no iteration structure for looping through the multiple targets is needed. On the other hand, with term default, when missing a target term in a single statement, ”ALL” will be assumed as a default scope, meaning the whole data model will be the target. This rule is applicable for all types of Zifazah statements, which means that target term is optional in all Zifazah syntax patterns.

Second, requirement for a conditional structure comes from users' requests for a means to express conditional processing. For example, they often filter fibers according to FA thresholds. In Zifazah, conditional expression can be flexibly embedded in a statement to avoid this structure. It has been shown in previous examples how to embed conditional expressions in SELECT statements. For syntactic simplicity, condition is expressed in UPDATE statements indirectly through variable reference, as the following another example snippet shows. Therein LOCATE is an alternative to SELECT but it results in a storage of the fibers filtered into a variable for later reference instead of highlighting those fibers immediately as SELECT does (see Section III for detailed language elements).

```
suspfibers = LOCATE "FA in [0.2,0.25]" IN "CST,ILF"
UPDATE size BY FA IN "suspfibers"
```

Figure 5 shows the resulting visualization.

Fully Declarative Language

Since the end users of Zifazah are medical experts who prefer natural descriptions over a programming style of thinking according to my talks with them, elements even slightly close to those in a computer programming language have been changed to be as declarative as

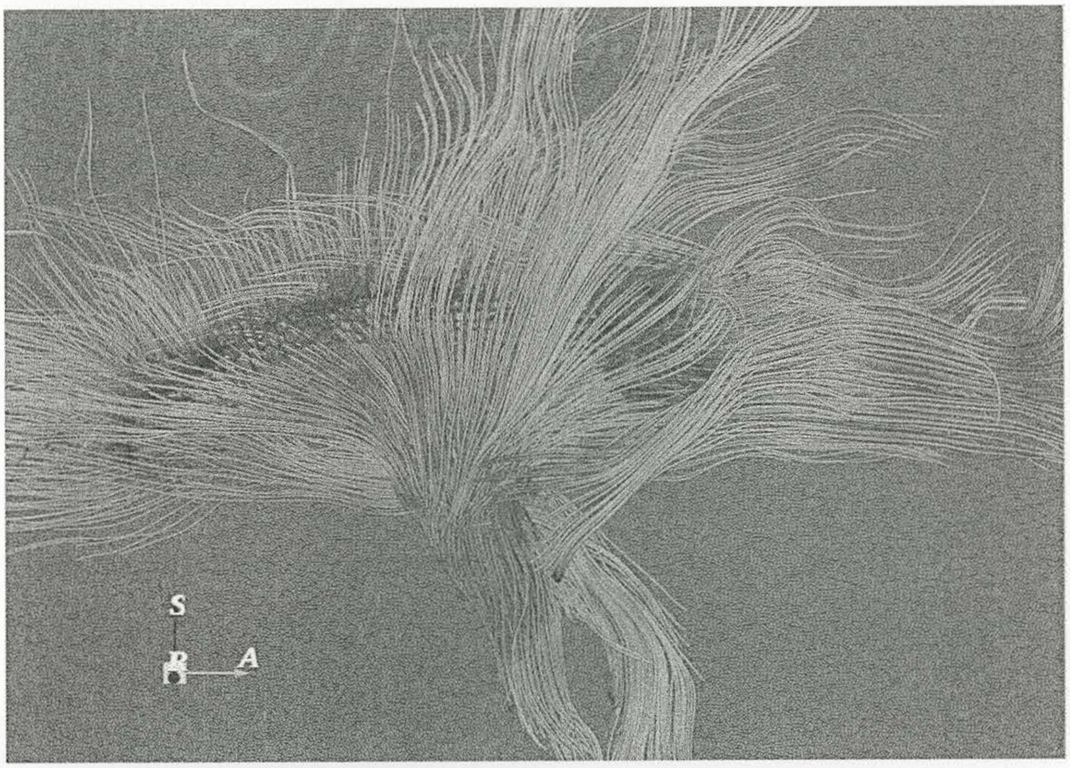


Figure 5. Illustration of the flat control structure of Zifazah program.

possible. In Zifazah, all types of statements are designed to be in a consistent pattern: started with a verb, followed by operations and, optionally, ended by data target specification, with optional evaluation of statement result to a variable for later reference if provided. This syntax consistency has been applied even to the data measurement statement where invocation of built-in numerical routines is involved. To measure the number of fibers in a selected bundle, for instance, instead of writing as:

```
CALCULATE  NumFibers("CST")
```

users with Zifazah write

```
CALCULATE  NumFibers IN "CST"
```

In addition, all keywords in Zifazah are case insensitive in order to reduce typing errors. Neuroscientists comment that these features make the language easy to learn and intuitive to use.

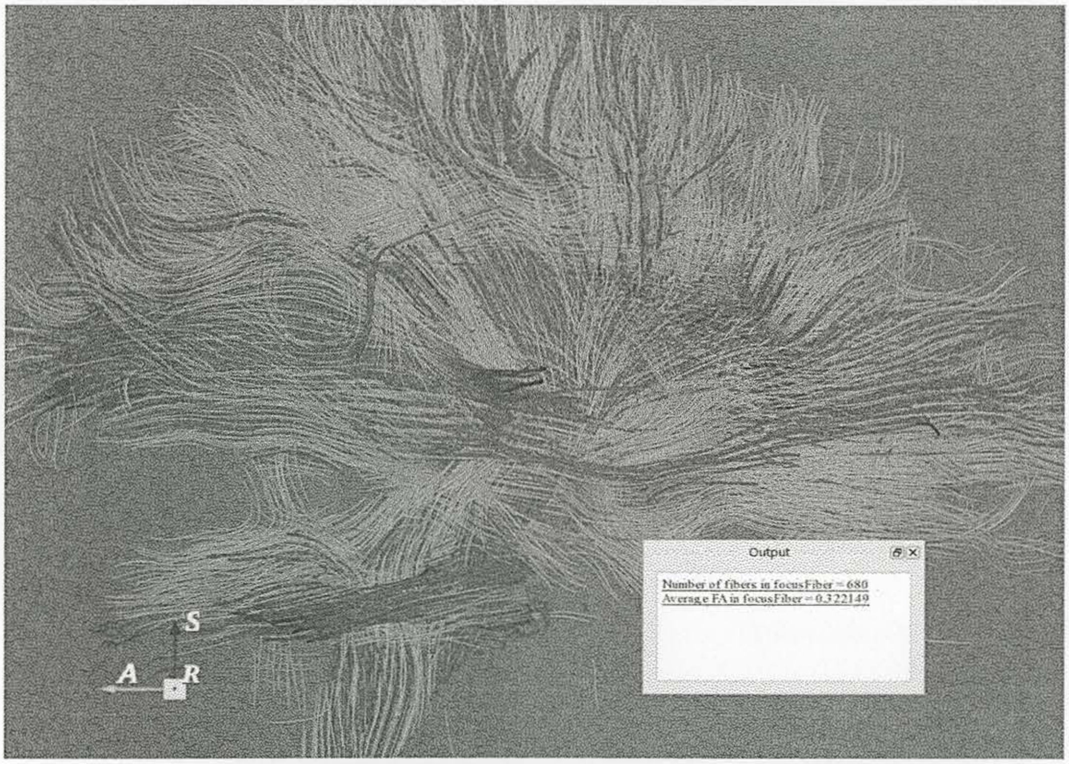
As exemplified above, besides visually examining the graphical representations, medical experts often need to investigate the DTI data itself in a quantitative manner. In clinical practice of neuroscientists using DTI, quantities such as average FA and number of fibers are important DTI tractography-based metrics for assessing cerebral white matter integrity [15]. In fact, these metrics are usually used in our end-user description of DTI visualizations as well. Accordingly, Zifazah provides capabilities to calculate some DTI metrics most frequently used in end users' practice of diagnosis through built-in numerical routines. The following pattern shows the Zifazah data analysis syntax.

```
val = CALCULATE metricRoutine IN|OUT target
```

At the current stage of Zifazah development, *metricRoutine* can be one of *AvgFA*, *AvgLA* and *NumFibers*, whose functions have been described before. In this syntax pattern, keeping the resulting value by evaluation is optional and sometimes useful when being referred to afterwards (see usage scenario 3 described in Chapter V). For example, in order to sum up fibers with average FA falling within a particular range and then figure out average LA of the target fibers, an end user can write the following script in Zifazah:

```
focusFiber = LOCATE "FA >= 0.285"
CALCULATE NumFibers in "focusFiber"
CALCULATE AvgFA in "focusFiber"
```

After running, the script above will dump results in the output window in the Zifazah programming environment as shown in Figure 6.



```
Number of fibers in focusFiber = 680
Average FA in focusFiber = 0.322149
```

Figure 6. Result of an example script showing Zifazah as a fully declarative language.

CHAPTER IV

IMPLEMENTATION

Zifazah is declarative in its general form with support of certain programming language features, such as variable referencing and arithmetical and logical operations. At this early stage, the language scripts are not executed via a fully-featured interpreter or compiler but a string-parsing-based translator of descriptive text to visualization pipeline components and manipulations upon them. The core of Zifazah is implemented on top of the Visualization Toolkit (VTK) using C++. The rendering engine is driven by the visualization pipeline and legacy VTK components ranging from various geometry filters to data mappers. However, in order to support language features such as mixed data encoding, a group of new pipeline components like those for view-dependent per-vertex depth value ordering has been extended on top of related VTK classes, and many legacy VTK components have been tailored for specific needs of visualizations in Zifazah.

In particular, the Zifazah script interpreter has also been implemented primarily as data filters in the VTK visualization pipeline. For instance, filtering according to thresholds of DTI metrics is developed as a set of separate VTK filters each serving a specific metric. As such, interpreting a Zifazah script is to translate the text, according to defined syntax and semantics, to data transformations in the VTK pipeline. For achieving the data encoding flexibility, multiple VTK data transformation pipelines have been employed.

Additionally, the overall programming interface is implemented using Qt for C++. Interactions like triggering the execution of a Zifazah program, serializing and deserializing the text script, etc. are all developed with Qt widgets, although the interactions with the visualization itself are handled using legacy VTK facilities with necessary extensions. Figure 1 illustrates the look of the current Zifazah programming interface. Both the code editor and "debugging" information window are dockable widgets, which facilitates the script programming by allowing free positioning and resizing as opposed to the visualization view.

Since our language is non-programmer oriented, program debugging skills are not expected of users. Consequently, instead of building a full-blown debugging environment as seen in almost all integrated development environments (IDEs), we simply use a dockable output window to prompt users with all error messages caused by invalid syntax or unrecognized language symbols. We have made use of GUI utilities of Qt for C++ to dump, after running a script, those messages to tell what is wrong and where in natural language descriptions with different levels of errors (fatal, warning, and notice, etc.) differentiated by different combinations of font size, type, and color of the text. Resulting values out of running data analyzing statements are also displayed in this output window. I do not set a separate window for displaying those numerical results in order to simplify the programming interface and, alternatively, we use remarkably disparate text background and underscore to highlight them among other messages. Also, natural language description has been used to present those numerical results so that they are easy to read and understand for end users.

CHAPTER V

USAGE SCENARIOS

In this chapter, we describe several sample tasks done by neurologists with visualizations of a brain DTI model using the Zifazah language. The usage scenarios associated with the sample tasks are representative of some typical real-world visualization tasks of neuroscientists and neurological physicians with expertise in DTI in their clinical practices. The usages range from visualization customization and exploration to DTI data analysis, covering the main language features and functionalities of the current Zifazah implementation.

In the following scenarios, Dr. Josh M. Anderson, a vascular neurologist and an end user of Zifazah, has a geometrical model derived from a brain DT-MRI data set and wants to compose and explore visualizations of the data for diagnosis purposes. For each of the scenarios, Josh fulfills his task by programming a Zifazah script that describes his thinking process for that task and then clicks the "Run" button to execute the script. Josh programs with Zifazah syntax references showing on a help window and corrects any term that is typed incorrectly with the assistance of error messages displayed in the output window. Once the script is interpreted correctly, either the visualization is changed or numerical values appear in the output window, as the results of script execution. Scripts and running results are presented at the end of the description of each usage scenario.

Scenario 1: Composing Visualizations

To start with, Josh specifies a data file that contains the geometries of the brain DTI model using the LOAD command. By default, running this single statement gives a streamtube visualization of the model with uniform visual encoding across all major bundles and without depth encoding. Suspicious of the association of a known disease named Corpus-Callosum-Agenesis (CCA) with the distribution of neural pathways at the intersection of the CC and CST bundles, Josh continues to customize the streamtube representation by mapping FA to tube radius along each CST fiber since he is interested in the FA changes of CST at the intersection, and encoding depth values of CC fibers to colors so that he can easily discern the genu and splenium fibers in the CC bundle along the depth dimension in the coronal view. Finally, Josh also wants to highlight the IFO fibers preferably represented with ribbons. Since the IFO bundle is roughly perpendicular to the CST bundle, he likes to take it as a reference as well. To achieve this task, Josh writes the final script after error corrections as:

```
LOAD "/home/josh/braindti.data"
Select "CC,CST,IFO"
Update size BY FA IN "CST"
Update depth BY color IN "CC"
Update shape BY ribbon IN "IFO"
```

The result in the visualization view is shown in Figure 7.

Scenario 2: Examining ROIs

It is quite common for neurologists to examine particular regions of interest (ROIs) rather than the whole brain when using DTI visualizations. In this task, Josh is only interested in all fibers within the temporal lobe area that belong to the CG bundle and CST fibers in the parietal lobe area that have average LA value no larger than a threshold to be determined. The SELECT command with relative spatial operations using the anatomical planes enables Josh to precisely reach the ROIs he desires. He firstly aims to filter fiber tracts outside the temporal and parietal area by adjusting the three cutting planes with relative movements and then starts trying to reach the exact target fiber tracts using both fiber

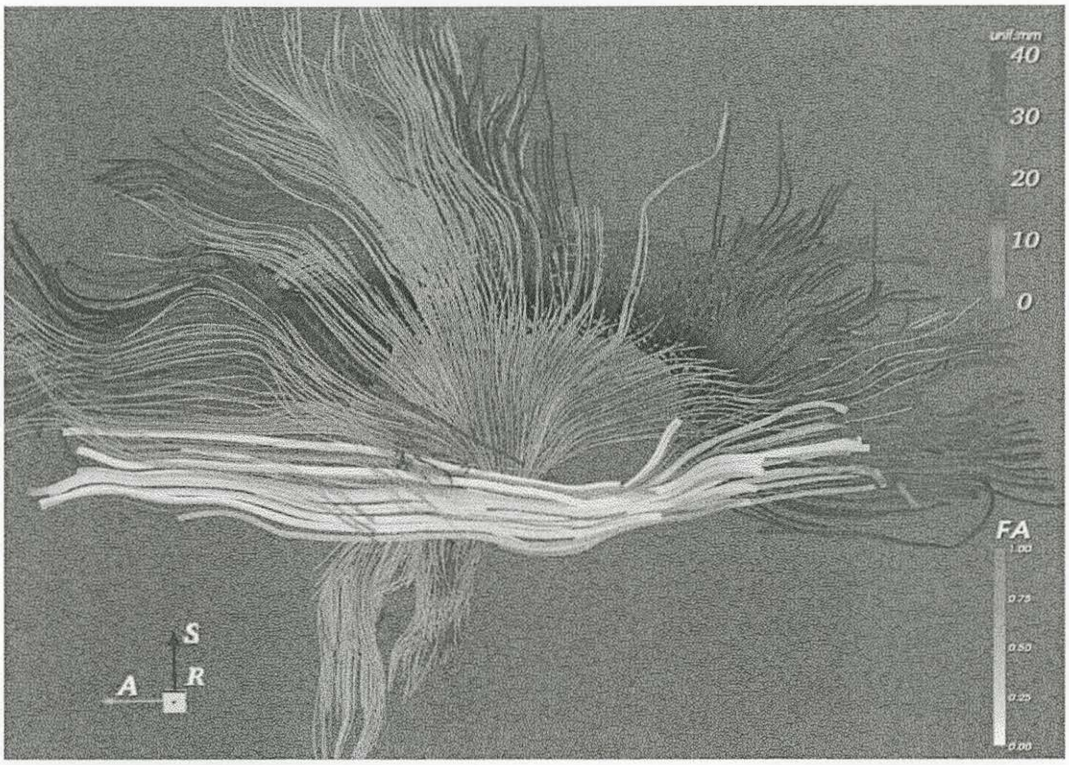


Figure 7. Screenshot of the visualization resulted from running the Zifazah program written in scenario 1.

bundle filters and conditional expression related to LA. With respect to the LA threshold undecided, Josh initially begins with an estimate and then keeps refining until he gets the accurate selection of target fibers. In the end, he has a runnable script written in Zifazah as:

```
LOAD "/home/josh/braindti.data"
Select "axial +63.35"
Select "sagittal +71"
Select "coronal -48.5"
Select "sagittal -0.25"
Select "axial +7.2"
Select "CG"
Select "LA <= 0.275" IN "CST"
```

As the result, Figure 8 shows the ROIs that Josh programs for.

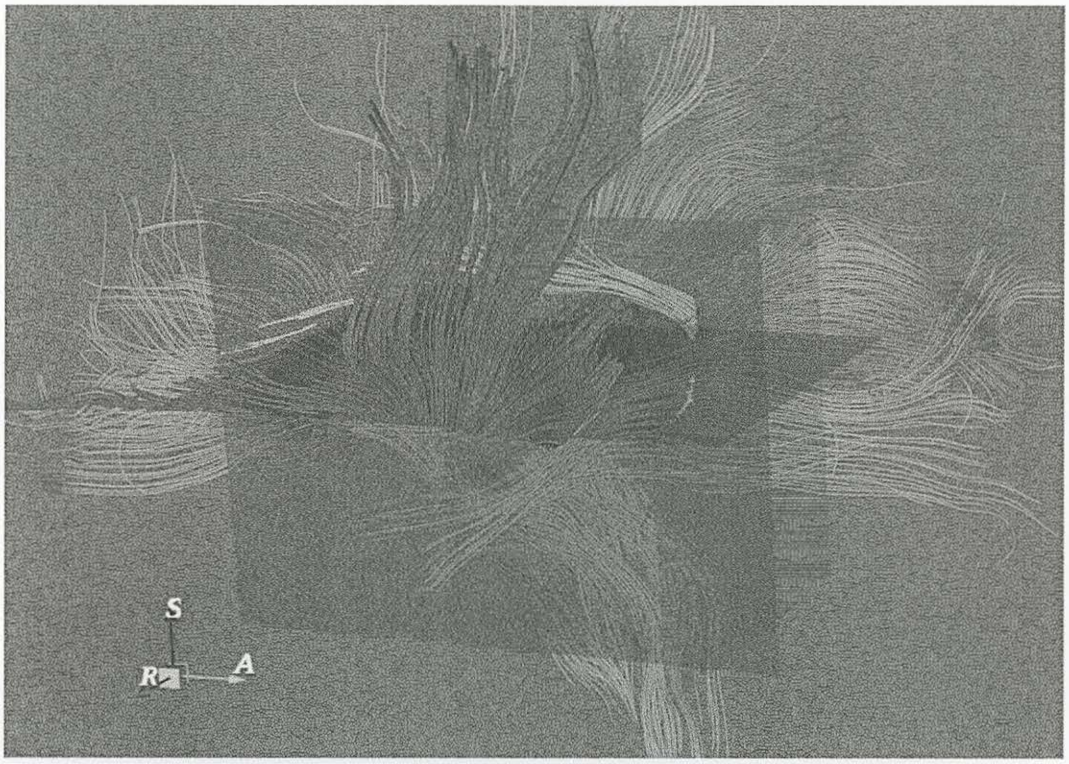


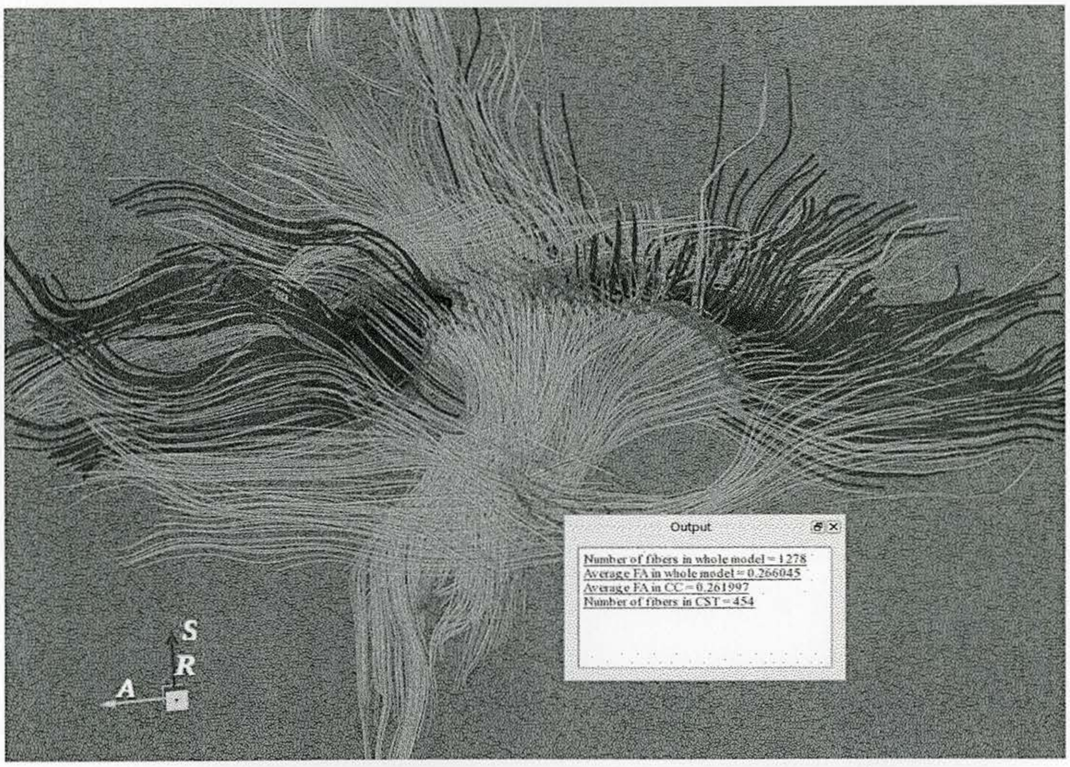
Figure 8. Screenshot of the result after running the Zifazah program that examines ROIs in scenario 2.

Scenario 3: Calculating Metrics

Beyond visual examinations, neurologists often request quantitative investigations of their DTI models as well. In this scenario, Josh attempts to check the white matter integrity in his brain model due to the limited reliability of DTI tractography. For a rough estimation of the integrity, he uses the CALCULATE command to retrieve the size, in terms of the number of fibers, and average FA of both the whole brain and representative bundles. With the average FA he has requested before, Josh goes further to make use of it to kick out CST fibers with average FA below the bundle-wise average. Josh writes the following script and obtains what he needs:

```
LOAD "/home/josh/braindti.data"
Select "ALL"
Calculate NumFibers
Calculate AvgFA

cstFAavg = Calculate AvgFA in "CC"

Calculate NumFibers in "CST"

Update reset IN "ALL"

Select "FA >= cstFAavg" IN "CC"
```

Figure 9 shows both the numerical values computed and the updated visualization using one of the values through variable reference.



```
Number of fibers in whole model = 1278
Average FA in whole model = 0.266045
Average FA in CC = 0.261997
Number of fibers in CST = 454
```

Figure 9. Screenshot of the running result of Zifazah script written for an end-user task in scenario 3.

CHAPTER VI

DISCUSSION

Since our language addresses scientific visualizations and targets non-programmer users, it is designed to be fully declarative with flat control structure. While these two design features make the language easy to use for domain users, it can cause difficulties in debugging the script since many low-level computations and control logics behind the high-level syntax are hidden for the users. In order to minimize such drawbacks of the current Zifazah design, the script interpreter has been developed to strictly check each current statement and stop further executions of the script once current return signals abnormal behaviors, such as importing invalid data input and referring to unknown variables.

In addition, regarding the execution mode, the current implementation of Zifazah does not follow a real-time interactive running mode by which the visualization is updated once the script changes. Instead, the programming interface requires a separate user interaction, such as clicking a button or pressing a key, for running the present script. This design is for interface simplicity and lower computational performance requirement, although a programming environment with otherwise real-time update is easy to implement.

While at the prototype stage, Zifazah is still under active development with an intention to add more useful features to this visualization language for the purpose of better user experience and more powerful language expressiveness from end-user perspectives.

Concurrent multiple-model exploration: While exploring more than one DTI model in order, i.e. switching data input from one to another using the LOAD command, has been supported, concurrent exploration of multiple models has not yet been implemented. However, requirements for doing so do exist among our end users. As an example, one typical case is to examine two brain models in which one is known as normal and another suspicious of a brain disease. This is often seen in clinical practice since the side-by-side comparison is helpful for efficient recognition of cerebral anomalies or simply finding structural differences. Corresponding Zifazah commands and related other types of symbols can be extended for such concurrent explorations. Among other changes, the evaluation of LOAD statements results to an identifier (a handle, for instance) can be utilized to identify a specific model out of multiple ones simultaneously explored.

Improved usability: Although Zifazah has been designed to be fully declarative and many features have been developed expressly for maximum usability, such as flat control structure and consistent syntax pattern, the usability of the overall programming environment can be further improved in two aspects. First of all, apart from a help window showing all symbols and syntactical details, which has already been implemented, context-aware automatic word completion can be built into the script editor so that users would not need to remember language keywords. Also, statement templates can be provided in the interface so that users can program a statement simply by filling blanks followed by clicking a button to confirm (then the statement will be added into the editor). Secondly, instead of only displaying error messages after execution, highlighting error-prone words when they are being typed is an additional editor feature.

CHAPTER VII

CONCLUSION

I have presented a visualization language for exploring 3D DTI visualizations and described the design principles and language features of it derived from end-user descriptions about how to customize and explore such visualizations. I have already developed some functions and features carefully selected for the proposed language, Zifazah, and described the elements of the language. A primary design goal with Zifazah is to initiate a scientific visualization language that is non-programmer oriented, especially for domain scientists who have no programming experience and skill to create and explore in their own visualizations. For this purpose, I have emphasized design features of Zifazah that particularly support the design goals.

I have also described representative usage scenarios of Zifazah apart from many example scripts written in the language before presenting its main content. These scenarios show that the new language is appealing to domain users, and it is promising to further develop the prototype towards a more capable and usable language for exploring more scientific visualizations.

While the development of the language as a whole is still at its early stage, the language core has already been implemented and more features are being extended on top of the current design. Among many possible directions to follow, I briefly discussed two main prospective features to follow up. With Zifazah I have presented a new approach, i.e. the end-user programming approach, to exploring DTI visualizations in 3D environment.

REFERENCES

[1] Impure: A new visualization programming language for non-programmers. `http://www.impure.com/`.

[2] Processing.js. `http://processingjs.org/`.

[3] Trevil - tree visualization language. `http://sape.inf.usi.ch/tools/trevil`.

[4] The vedea project. `http://blogs.msdn.com/b/martinca/archive/2009/12/03/introducing-the-microsoft-visualization-language.aspx`.

[5] Andrea Adamoli and Matthias Hauswirth. Trevis: A context tree visualization & analysis framework and its use for classifying performance failure reports. In *ACM Symposium on Software Visualization*, pages 73–82, October 2010.

[6] David Akers. Cinch: a cooperatively designed marking interface for 3d pathway selection. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 33–42, New York, NY, USA, 2006.

[7] David Akers, N. Gluhbegovich, and J. Jew. Wizard of oz for participatory design: Inventing a gestural interface for 3d selection of neural pathway estimates. In *SIGCHI Works in Progress*. ACM Press, 2006.

[8] Peter J. Basser, Sinisa Pajevic, Carlo Pierpaoli, Jeffrey Duda, and Akram Aldroubi. In vivo fiber tractography using DT-MRI data. *Magnetic Resonance in Medicine*, 44:625–632, 2000.

[9] J. Bertin. *Semiology of graphics: diagrams, networks, maps*. University of Wisconsin Press, 1983.

[10] J. Blaas, C. P. Botha, B. Peters, F. M. Vos, and F. H. Post. Fast and reproducible fiber bundle selection in DTI visualization. In *IEEE Visualization*, pages 59–64, 2005.

[11] Coyne Bob and Sproat Richard. Wordseye: an automatic text-to-scene conversion system. In *Proceedings of ACM SIGGRAPH*, pages 487–496, New York, NY, USA, 2001.

[12] M. Bostock and J. Heer. Protovis: A graphical toolkit for visualization. *IEEE Transactions on Visualization and Computer Graphics*, pages 1121–1128, 2009.

[13] Mike Bostock. Data-driven documents. `http://mbostock.github.com/d3/`.

[14] Olston Christopher, Reed Benjamin, Srivastava Utkarsh, Kumar Ravi, and Tomkins Andrew. Pig latin: a not-so-foreign language for data processing. In *Proceedings of the ACM SIGMOD international conference on Management of data*, pages 1099–1110, New York, NY, USA, 2008.

[15] S. Correia, S. Y. Lee, T. Voorn, D. F. Tate, R. H. Paul, S. Zhang, S. P. Salloway, P. F. Malloy, and D. H. Laidlaw. Quantitative tractography metrics of white matter integrity in diffusion-tensor mri. *Neuroimage*, 42(2):568–581, 2008.

[16] N. Elmqvist and P. Tsigas. A taxonomy of 3d occlusion management for visualization. *IEEE Transactions on Visualization and Computer Graphics*, pages 1095–1109, 2008.

[17] B. J. Fry. *Computational information design*. Ph.D. dissertation, Massachusetts Institute of Technology, 2004.

[18] J. Heer and M. Bostock. Declarative language design for interactive visualization. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1149–1156, 2010.

[19] Radu Jianu, Cagatay Demiralp, and David H. Laidlaw. Exploring 3D DTI fiber-tracts with linked 2D representations. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1449–1456, 2009.

[20] Gordon Kindlmann. *Visualization and analysis of diffusion tensor fields*. Ph.D. dissertation, The University of Utah, 2004.

[21] Gordon Kindlmann, David Weinstein, and David Hart. Strategies for direct volume rendering of diffusion tensor fields. *IEEE Transactions on Visualization and Computer Graphics*, pages 124–138, 2000.

[22] Y. Li, C. W. Fu, and A. Hanson. Scalable wim: Effective exploration in large-scale astrophysical environments. *IEEE Transactions on Visualization and Computer Graphics*, pages 1005–1012, 2006.

[23] W. Lucas and S. M. Shieber. A simple language for novel visualizations of information. *Software and Data Technologies*, pages 33–45, 2009.

[24] J. Mackinlay. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, 5(2):110–141, 1986.

[25] J. Mackinlay, P. Hanrahan, and C. Stolte. Show me: Automatic presentation for visual analysis. *IEEE Transactions on Visualization and Computer Graphics*, pages 1137–1144, 2007.

[26] R. Metoyer, B. Lee, N. H. Riche, and M. Czerwinski. Understanding the verbal language and structure of end-user descriptions of data visualizations. *ACM CHI*, 2012.

[27] John Peterson. A language for mathematical visualization. In *Proceedings of Functional and Declarative Languages in Education*, October 2002.

[28] Steve Pieper and Ron Kikinis. 3d slicer. `http://www.slicer.org/`.

[29] Anthony Sherbondy, David Akers, Rachel Mackenzie, Robert Dougherty, and Brian Wandell. Exploring connectivity of the brain's white matter with dynamic queries. *IEEE Transactions on Visualization and Computer Graphics*, 11:419–430, July 2005.

[30] Chris Stolte and Pat Hanrahan. Polaris: A system for query, analysis and visualization of multi-dimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics*, 8:52–65, 2002.

[31] N. Toussaint, J. C. Souplet, and P. Fillard. MedINRIA: Medical image navigation and research tool by INRIA. In *Proceedings of MICCAI*, volume 7, pages 1–8, 2007.

[32] Chen Wei, Zi'ang Ding, Song Zhang, A. MacKay-Brandt, S. Correia, H. Qu, J. A. Crow, D. F. Tate, Z. Yan, and Q. Peng. A novel interface for interactive exploration of dti fibers. *IEEE Transactions on Visualization and Computer Graphics*, pages 1433–1440, 2009.

[33] Song Zhang, Charlie Curry, Daniel S. Morris, and David H. Laidlaw. Visualizing diffusion tensor mr images using streamtubes and streamsurfaces. *IEEE Transactions on Visualization and Computer Graphics*, pages 454–462, 2003.