Gryphon: A 'Little' Domain-Specific Programming Language for Diffusion MRI Visualizations

Jian Chen, Haipeng Cai, Alexander P. Auchus, and David H. Laidlaw

Abstract We present Gryphon, a 'little' domain-specific programming language (DSL) for visualizing diffusion magnetic resonance imaging (DMRI). A key contribution is its compositional approach to customizing visualizations for evolving analytical tasks. The language is designed for non-programmer, here brain scientists for exploratory studies. The semantics of Gryphon includes a simple set of keywords derived from brain scientists vocabulary while performing imaging tasks of mapping data to graphic marks such as color, shape, value, and size. A pilot study with two neuroscientists suggested that Gryphon was easy to learn, though some additional functions and interface components are needed to empower brain scientists.

J. Chen (✉)
Assistant professor, Computer Science and Electrical Engineering, University of Maryland Baltimore County, Baltimore, MD, USA
e-mail: jichen@umbc.edu

H. Cai
Graduate student, School of Computing, University of Southern Mississippi, Hattiesburg, MS, USA
e-mail: haipeng.cai@eagles.usm.edu

A.P. Auchus
Professor, Department of Neurology, University of Mississippi Medical Center, Jackson, MS, USA
e-mail: aauchus@umbc.edu

D.H. Laidlaw
Professor, Computer Science Department, Brown University, Providence, RI, USA
e-mail: dhl@cs.brown.edu

W. Huang (ed.), *Handbook of Human Centric Visualization*,
DOI 10.1007/978-1-4614-7485-2_2, © Springer Science+Business Media New York 2014

1 Introduction

Brain researchers have long used diffusion magnetic resonance imaging (DMRI) techniques to study connectivity of brain structures. DMRI is a MRI technique that measures motion of water molecules in tissue [21]. Experimental evidence has shown that water diffusion is anisotropic in organized tissues, such as white matter and muscle, and that reconstructing the orientation and curvature of white matter can provide detailed information about neural pathways. The curves (or fibers) are portrayed graphically using streamline tracing algorithms or glyphs such as hyperstreamlines initialized at seed points to reveal fiber tracts. Tracts following similar directions form fiber bundles [22]. Numerous studies have shown that tract-based analysis of the brain white matter can offer a deeper insight into brain structures.

Understanding the structures of fiber tracts, however, has proven difficult for several reasons that lead to the needs for effective visualization and interaction techniques. First, the advances in image capturing and processing techniques permit the display of human brain features at millimeter scales, generating highly dense visualizations. A whole-brain tractography can have about 10,000 tubes within the volume of a human head. Without effective visualizations, brain scientists are left with occluded images that impede their view of the data. Second, grouping, trimming, and labeling of the numerically computed fibers are needed for the tractography to provide information about the connections between brain regions. Fiber tracking reliability can further vary with imaging resolution, noise, and patient orientation as well as decreased anisotropy that occur with disease. Therefore, interaction techniques are also crucial for performing some operations to reach certain regions of interest (ROI) in the brain for the tasks at hand [1].

One approach to addressing the sheer data complexity in DMRI is to provide a large number of geometric processing, registration, segmentation, and clustering algorithms to improve data analysis [6]. Further coupling these algorithms with graphical user interfaces (GUIs) made possible more efficient data queries and analysis workflow [32]. This paper takes a complementary approach to focus on a programming language where the brain scientists can customize visualization that has good visual encodings to convey more precise information. In general, relying the same visualization for all needed tasks is impossible. For example, streamtubes are not always good visual representations to reduce clutter, though they improve orientation perception [12]. Tensor shapes, sizes, and their spatial locations also influence legibility of the visual displays. If visual encoding is not carefully constructed and applied to data, the reader may become misled by the visualization [28].

We describe Gryphon, a domain-specific programming language that lets brain scientists quickly compose a visualization tailored to their brain DMRI data. We report a prototypical implementation in this work exemplified in Fig. 1. A Gryphon program has a sequence of steps, each of which carries out a single high-level visualization operation, e.g., grouping, aggregation, and other modification on

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

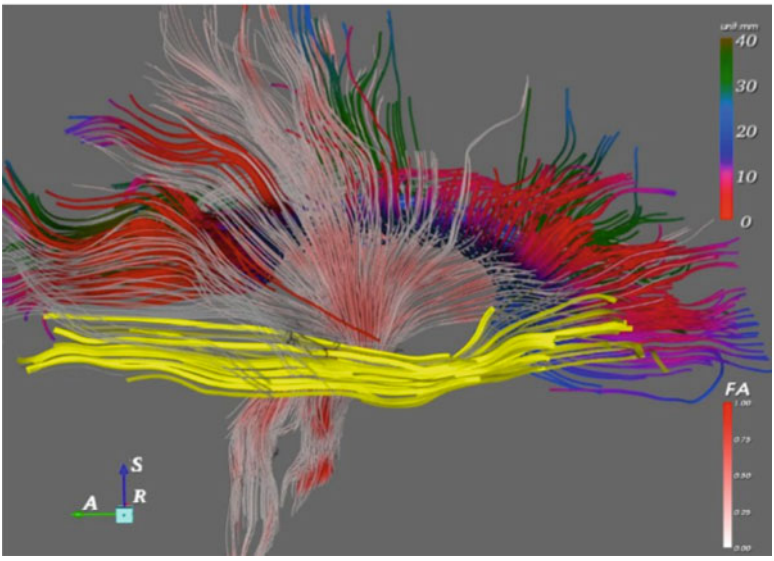

Fig. 1 Mixed color encoding of various fiber bundles

graphical elements such as color, size, value, shape, etc. Our language design makes three contributions: (1) a compositional approach to understanding visual encoding in scientific visualizations; (2) the design process to capture the domain-specific semantics to design this (we hope) easy-to-use "little" language; and (3) the language, Gryphon, itself to exemplify our design methods.

2 Background and Related Work

2.1 DMRI Visualization Toolkits

The first approach to addressing the sheer data complexity in DMRI is to design data processing, registration, segmentation, and clustering algorithms to improve data analysis. Some algorithms and toolkits are generic. SCIRun, MeVisLab, and Amira are general-purpose tools that provide comprehensive workflows in which users can directly manipulate the data [16, 17, 30]. Others are domain specific for DMRI data analysis. OpenWalnut supports diverse data processing and analyses of DMRI, computed tomography (CT), MRI, and functional MRI (fMRI), that on the one hand

emphasizes speed and extensibility, similar to Amira and, on the other hand, handles fluid data processing workflow, similar to MeVisLab and SCIRun [13]. Diderot is perhaps the first language tool that provides a rather complete set of algorithms for tensor field processing. It also introduces parallel algorithm for more efficient computing [6].

MedINRIA and DTIStudio also include data-processing pipelines for DMRI visualization for tract generation, processing of tracking data, and visualizations [15, 32]. A set of tools on the neuroimaging tools and resources (NITRC) website are uploaded regularly to help problem solving in specific neurological diseases (http://www.nitrc.org). TractVis, 3D Slicer, and ExplorDMRI have distinct functionality for measuring uncertainty [18, 25, 33]. vIST/e (http://bmia.bmt.tue.nl/software/viste/) is useful for high-angular-resolution diffusion imaging analysis (HARDI) for glyphs, tube visualizations and seeding controls and fusing DMRI/HARDI visualizations. Users can interactively explore and observe both local tensor field information and global details about brain structures.

Our goal for Gryphon is to focus on visualization design driven by human visual perceptions, to provide a DMRI visualization tool similar to ColorBrewer for color design [14]. We wish to either make use of known good design practice or run a set of experiments to collect design principles that will guide our visualization tool design for brain scientists detecting information from imaging. The data processing methods can be preprocessing steps before data are visualized.

2.2 Visualization Languages

There are two ways to study visualization language: using language as a framework to map data to visual features to perception [3], and using language as a programming tool for computers to draw visualizations on the screen [19].

These two language aspects were formally linked since APT, which is the first automated graphic presentation tool to characterize graphic presentations, provide a set of criteria for deciding the role of each visible sign or symbol placed in a graphic by studying effectiveness and efficacy, and finally implement the programming language to 'recommend' good visualizations [19]. Wilkinson also describes a set of grammatical rules for defining graphics that mainly focus on the data analysis process [34], similar to Bertins semiology for graphics mark drawings [3].

Many graphics presentation tools have used APTs style of analysis for formal graphical languages. For example, SAGE expanded APT's data-centric visualization to a task-centric presentation system that responds to user queries for information and generates explanations of the changes occurring in quantitative molding systems such as project modeling and financial spreadsheets [29]. The aim of Processing, D^3, and ProtoVis is information visualization to design multivariate data visualizations or information graphs [4, 5, 26].

Our Gryphon programming language is similar to APT [19], D^3 [5], or Processing [26] in being built from the semiotic perspective: color, shape, value,

texture work as sign vehicles for users to represent some parameter measurements. We expand the concept to scientific uses in DMRI visualizations to display spatial, physically-based, and spatial data versus their abstract and chosen representations [31]. Additionally, depth and occlusions in 3D that make it difficult to see all the data necessitate certain graphical tricks [35]. Another difficulty with adding 3D is that orientation becomes an issues: it makes easy to get lost providing the ability to move around in space. Thus, validating, measuring, and constructing effective 3D visualizations in general are still unsolved problems for dense dataset visualizations.

From the language design perspective, our Gryphon is also a 'little' language [2] that is simple for brain scientists to build, document, learn, and use. All instructions are captured following a human-centered computing design process to capture domain semantics. All programming instructions operate on small chunks of data of DMRI geometries.

Recently, Metoyer et al. report from an exploratory study a set of design implications for creating visualization languages and toolkits [20]. Their findings inform visualization language design through the way participants 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. We have obtained similar findings and have designed Gryphon to reflect high-level semantics and spatial data expression.

3 Gryphon: A DMRI Visualization Language

3.1 Language Design

3.1.1 Linguistic Analysis

The first contribution of Gryphon lies in its simple syntax and semantics, carefully selected by two methods: experts's reviews while performing some tasks and experts' comments on the use of shapes, color, size, value, and texture. These methods assume that the brain scientists programming effort would be reduced if the computer language enables them to express semantics in natural expressions explicitly similar to the way they think about them, and share some design goals with the natural programming environment [23].

We recorded about 5 h of audio with four brain scientists who were asked to query brain DMRI tractography visualizations. Think-aloud protocol was enforced when they spoke about their intent. Their speech semantics were coded, tagged, and ranked using a speech tagger and a statistical parser. The output of this process was a parser tree that represent the structure of the sentence. Note that since the parser was statistical, it attempted to resolve ambiguities, such as prepositional phrase attachments. The parse tree was then converted into a representation that was simply

Table 1 Gryphon language symbols and keywords: these keywords are not case-sensitive

Verbs	load, selection locate, update, calculate
Prepositions	in, out
Conjunctives	by, with
Selection rules	[], <, <=, >, >=, ==, =, +, −
Build-in routines	AvgFA, AvgLA, NumFiber
Constants	shape, color, size, depth, FA, LA, sagittal, axial, coronal
	CC, CST, CG, IFO, ILF, DEFAULT, RESET

a split of the words in the sentence, showing the words that they depended on and the words that were dependent on them. For example, say that we wish to select the structure corpus callosum (CC) which was inferior to the cortical spinal tracts. A neurologist would say, "*Select the inferior structure of CC next to the CST*". Such spatial reference was frequently used when the brain scientists talked about embedded structures.

The next phase of the analysis involved converting the dependency structure into a semantic representation. The semantic representation was a description of the top-ranked verbs and nouns, and relative spatial relationships that converged to a relatively small semantic space. We further queried the neuroscientists subjective preferences for colors, shape, and texture values for them to understand DMRI streamtube visualizations, keeping in mind the visual analysis of how structure affected human understanding.

As an example, we asked them about spatial relationships when using multiple visual mappings of depth values to size (the further away, the smaller) and color (the deeper, the redder in a blue-to-red color scale). Their comments included *"it is misleading to have the different sizes while colors are present to discern depth"*, and *"I would rather have it (the size) stay the same as I spin it (the model) around"*, *"Visual mapping of depth to color is preferable since I like it (the model) with colors. That is what I need to look at, and I think that color is a good idea but prefer color by orientation (of the fiber segments)"*.

We also hoped that each sentence in the language could record a segment of meaningful actions in the neuroscientists' workflow notation, that neuroscientists could revisit and reuse for new analyses. From our interview, brain scientists felt that current measurement matrices were sufficient, such as those reported in Correia et al. [7], yet the interaction and visualization were somewhat limited for allowing them to query effectively.

The analyses above produce the core content of Gryphon, the simple set of language symbols and keywords shown in Table 1. Five key verbs describe key actions brain scientists would like for DMRI visualizations. Prepositions are used to define the scope of the actions and conjunctives to connect statement terms. Selection rules in Gryphon are exactly the same as those in elementary math. Specifically, "[ ]" is the range operator to give numerical bounds in conditional expressions, and "+" and "−" are relative increment and decrement for brain

scientists to shift their previous operations spatially rather than giving absolute coordinates, e.g., moving a cutting plane by 5 mm. Some built-in routines or keywords are chosen following the tensor field measurement matrix in Correia et al. [7], e.g., AvgGA and AvgLA calculate the average FA (fractional anisotropy) and average LA (linear anisotropy) of fibers accordingly and NumFiber counts the number of fibers in a fiber bundle. Gryphon also recognizes some clustered fiber bundles, such as corpus callosum (CC), corticospinal tracts (CST), cingulum (CG), inferior longitudinal fasciculus (ILF), and inferior frontal occipital fasciculus (IFO).

3.1.2 Data Model and Input

Gryphon focuses on visual transformations in 3D visualizations. In our current implementation, fibers are clustered in terms of brain anatomy; each fiber is manually tagged with an anatomical cluster identity as either one of the five major bundles of CC, CST, CG, IFO, or ILF or not. In practice, Gryphons ability to recognize the constants for the major anatomical bundles depends on these cluster tags in the structure of the data model input. However, our language design is not restricted to visualizing clustered data: Gryphon can be expanded to be adaptable to an unclustered data model.

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

```
normalBrain = LOAD "/home/lucy/normalS1.dat",
```

where the LOAD command parses the input file and creates data structures that fully describe the data model, including identifying the clustering tags. This input specification statement can also update the current data model at the beginning of the visualization pipeline if it is not the first step in a Gryphon script. The evaluation is optional and, when provided, saves the result to a variable, here normalBrain, for later reference.

3.1.3 Encoding Composition

According to Bertin's semiotic theory [3], graphically encoding data with key retinal elements of color, size, orientation, texture, value, and shape is critical in the legibility of two-dimensional (2D) graphical representations. In 3D visualizations, occlusion, an important factor in depth perception, has a detrimental impact on overall legibility, and depth cues are an ordinal dimension in the design space of 3D occlusion management for visualization. Numerous good visualization design adopted certain occlusion management [10] mechanisms, such as halo effects [11].

We have contributed to symbolic mapping of color, size, and shape for 2D graphical legibility enhancement and depth encoding, also via common visual

```
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"
```

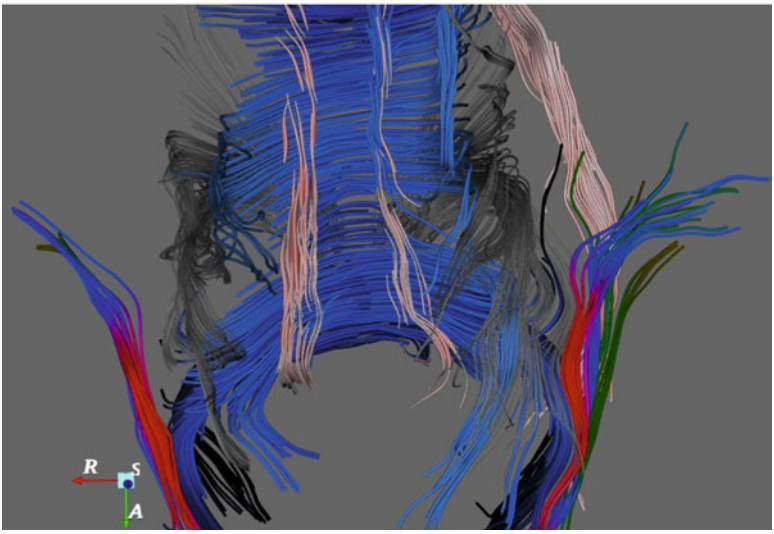

Fig. 2 Another example showing the mixed encoding by using the separation principles to depict fiber bundles

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, Gryphon allows brain scientistis to freely customize DMRI visualizations using either a single data-encoding scheme alone or compound encoding scheme by flexibly combining multiple encoding methods.

In composing or exploratory process with DMRI visualizations, brain scientists often attempt to compare a ROI across datasets captured at a different time or from another patient and would like to differentiate them from each other, thus a depth-enhanced visualization could help generate legible displays for the multiple ROIs. In addition, on other occasions users have difficulty in perceiving information along the depth direction.

Gryphon's data-encoding flexibility is also driven by the perceptual principle of shape choices, which states that it is easy for human eyes to perceive categorical data in different shapes. Figure 2 demonstrated some encoding approaches supported by the language, though the picture itself might not represent a good visualization. Suppose a brain scientist has composed the streamtube visualization of a brain DMRI data set with default data encoding (uniform size, color, and shape without depth cues) and now wants the overall encoding scheme to differ across fiber

Table 2 Combinational rules of constants in the UPDATE statement

var1	var2	var3
shape	line, tube, ribbon	–
color	FA, LA	–
size	FA, LA	minimal, scale
depth	size, color, value, transparency	lower, upper
DEFAULT	–	–
RESET	–	–

bundles. In order to achieve this effect, an example Gryphon snippet can be written as follows:

In the resulting visualization in Fig. 2, each of the five major bundles will differ visually from others since all these bundles are encoded differently (these mixed encoding approach is subject to experimental verification). Often times, once a ROI is filtered out, it is also necessary to examine the selected fibers more carefully. For this purpose, Gryphon allows brain scientists to impose various data-encoding schemes upon data of interests. Such visualization customization is performed by the UPDATE command, which works in an immediate mode and updates the view after execution. The general UPDATE syntax pattern is:

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

where *var1* is an attribute, such as shape, color, size, depth, of the current visualization to be modified, and *var2* gives how the actual updating operation is to be performed relative to *var1*. The parameter list, *para1...N* statement allows specific *UPDATE* 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 *var1*, *var2* and associated parameter list developed currently. 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 accompanying the verb *UPDATE*, act as a command to revoke all data-filtering and data-encoding operations respectively. The following script shows how to inspect the change in FA along fibers in a ROI by mapping FA value to tube size; this yields an alternative representation of the FA variation in that ROI, compared to some conventional coloring approach [9].

```
UPDATE RESET
partialILF = LOCATE "FA in [0.5,0.55]" OUT "ILF"
UPDATE size BY FA IN "partialILF"
```

3.1.4 Considering Interactivity

The third design contribution lies in the support of some typical interaction tasks: (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 DMRI quantities such as FA along a group of fibers; and (5) picking particular fibers according to a quantitative threshold.

When using DMRI visualizations, neurologists look at not only the whole data model, but also some regional details, for example the location of pathological conditions. To reach ROIs in brain DMRI visualizations, brain scientists often narrow down the view scope to a relatively large anatomical area in the first place and then dive into a specific ROI. In visualizations in which neural pathways are depicted as streamtubes, the ROIs are usually clusters of fiber bundles.

For instance, at the beginning of a visualization exploration, one of our brain scientist collaborators wants to look into frontal lobe fibers within the intersection of two fiber bundles, CST and CC, and will ignore 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 brain scientist continues examining the suspect fibers. Later on, the scientist focuses on the small fiber region to see how it differs from typical ones, for instance in terms of orientation and DMRI parameter measures in the evaluation metrics. Gryphon supports this process through high-level primitives, such as SELECT and common arithmetical conditional operators, including the range operator. Gryphon mainly contains facilities for step-by-step data filtering with these primitives. For example, supposed the user above wants to explore the fibers of interest, the Gryphon program will look like:

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

As a result, fibers in both specific bundles with average FA under 0.5 will be highlighted to help brain scientists focus on the local data being explored. In addition to 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 the brain scientist wants to keep the data already found in focus before moving to explore other relevant local data so as to add more fibers into the focus region, or when the scientist simply seeks a more legible visualization of the data. The instance below, following the same example, illustrates how 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:

```
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 brief sequence of commands can help the brain scientists locate desirable fiber tracts with high accuracy while allowing flexible customization upon current visualizations, similar to some slider dragging selection in interfaces [15]. In this case, tracts of interest (TOIs) are first focused and then further differentiated for more effective exploration. In general, Gryphons design emphasizes this task-driven process of visualization exploration, which fits the thinking process of end users of the present visualizations. Our neurologist collaborators frequently filter data in order to reach an ROI in their DMRI visualizations.. Gryphon offers two commands for data filtering: *SELECT* and *LOCATE*. The data-filtering syntax pattern in Gryphon is as follows:

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

These two commands have similar functionality but 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 combines 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, we have kept these two commands separate based on brain scientists' input asking for a more straightforward understanding of the semantics and more easily learned language usage, for example.

```
SELECT "LA <= 0.72" IN "ALL"
partialILF = LOCATE "FA in [0.5,0.55]" OUT "ILF"
```

The *SELECT* statement filters fibers in the entire DMRI model with average anisotropy greater than 0.72 (by putting them in the contextual background) and highlights all other fibers. In comparison, the *LOCATE* statement does not update the visualization but picks up fibers outside the ILF bundle having average FA value in the specified range. Note that when no specific data encoding is applied, different colors are assigned to ROI fibers in different major bundles in Gryphon so that one ROI can be distinguished from another when more than one is highlighted. Also, filtered fibers are still semitransparent as the contextual background rather than being removed from the visualization.

3.1.5 Spatial References

The fourth design choice of Gryphon is that it is a language in which brain scientists are able to operate with relative spatial terms. Brain scientists frequently use spatial terms such as parasagittal, in, out, mid-axial and near coronal, etc. in

their descriptions of DMRI visualizations in the 3D space. They also use a set of relative positioning terms, such as *above*, *under*, *on top of*, *across*, and *between*, etc., and more domain-specific ones such as *frontal*, *posterior*, *dorsal*, etc. At present, Gryphon contains a subset of these spatial terms.

In 3D data models such as those from DMRI, spatial relationships between data components are one of the essential characteristics (as is typical of 3D scientific data in genera). Accordingly, in composing a DMRI visualization one must be able to use spatial operators with domain-conventional terms in order to describe the process of visualization authoring. In response, Gryphon supports spatial operations by combining two approaches. First, three visible cutting planes that provide guidance in the three conventional anatomical views, the axial, coronal, and sagittal views, are integrated in the visualization view. Then, flexible manipulating operations upon the three planes are built into Gryphons spatial syntax definitions. This enables brain scientists to navigate in the dense 3D data model with a highly precise filtering fashion through numerical input. For instance, suppose the streamtube representation of a DMRI model being programmed is derived using unit seeding resolution from DMRI volumes of size $256 \times 256 \times 31$ captured at voxel resolution $0.9375 \times 0.9375 \times 4.52$ mm, 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 a suspect anomaly in the brain region of the occipital lobe, a brain scientists can filter the data model so that approximately only this region is kept. Relative movements can be imposed similarly on the sagittal plane as well using the code below.

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

3.1.6 Flat Control Structure

Gryphon is designed to provide a declarative language environment for brain scientists who may not have programming skills. Therefore, we purposely eliminate the conditional and iterative structures from the language design of Gryphon and keep only the most intuitive sequential structure. This gives Gryphon a flat control structure. Meanwhile, Gryphon uses high-level semantics to overcome its weakness in expressing user task requirements. First, the need for an iterative structure usually stems from the need to operate on multiple targets. In Gryphon, the operation target is a common term in all syntax patterns to indicate the scope of the data. These scopes are defined as enumeration and default terms in Gryphon syntax patterns. On the one hand, with enumeration, brain scientists simply list all targets in the target term, thus avoiding iteration. For example, suppose a user intends to select three bundles and then to change the size encoding for two of them; a Gryphon script can include:

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

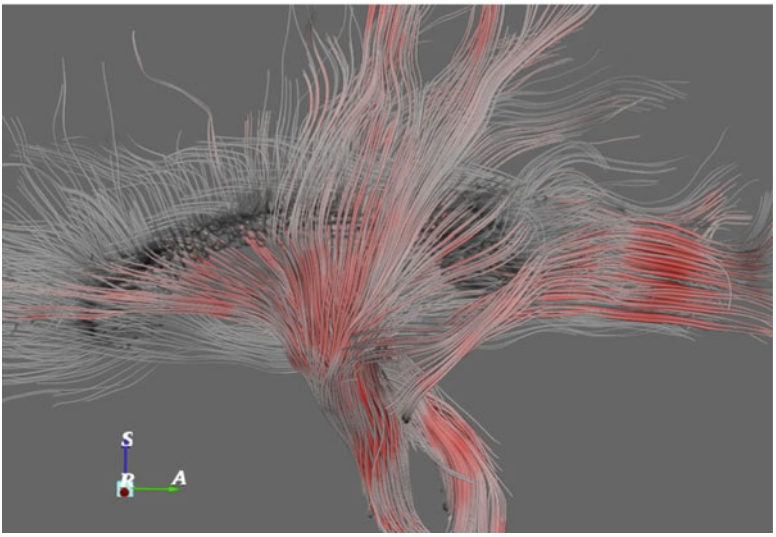

Fig. 3 LOCATE function only stores the data without necessarily making a selection

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

On the other hand, with term default, when a target term is lacking in a single statement, *ALL* is taken as the default scope, meaning the entire data model will be the regions to be manipulated. This rule is applicable to all types of Gryphon statements, that target term is optional in all Gryphon syntax patterns.

Second, requirement for a conditional structure comes from brain scientists' requests for a way to express conditional processing. For example, they often filter fibers according to *FA* thresholds. In Gryphon, a conditional expression can be flexibly embedded in a statement. We have 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 example snippet shows. Here *LOCATE* is an alternative to *SELECT* but it results in storage of the fibers filtered into a variable for later reference instead of highlighting those fibers immediately, as *SELECT* does (Fig. 3).

3.1.7 Fully Declarative Language

We designed Gryphon for brain scientists using natural descriptions over a classical programming language: elements close to those in a computer programming language have been changed to be as declarative as possible. In Gryphon, all types of statements are designed to follow a consistent pattern: begun by 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 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

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

write

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

As shown above, besides visually examining the graphical representations, brain scientists often need to investigate the DMRI data in a quantitative manner, such as average FA and number of fibers for assessing cerebral white-matter integrity. Accordingly, Gryphon provides built-in numerical routines to calculate some of the DMRI metrics in Correia et al. [7]. The following pattern shows the Gryphon data analysis syntax:

```
val = CALCULATE ${metricRoutine}$ IN|OUT target
```

MetricRoutine can be one of *AvgFA*, *AvgLA* and *NumFibers* to represent fiber integrity. In this syntax pattern, keeping the resulting value by evaluation is optional and sometimes useful when referred to afterwards. For example, in order to sum the fibers with average *FA* falling within a particular range and then figure out the average LA of these target fibers, a brain scientist can write in Gryphon to show its visualization in Fig. 4.

3.2 Language Implementation

Gryphon is declarative in general form, with support of some programming language features, such as variable referencing and arithmetical and logical operations. Current implementation does not support a fully featured interpreter or compiler but by a string-parsing-based translator of descriptive text to visualization pipeline components and manipulations. The core of Gryphon 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. Moreover, the support of language features, such as compositional data encoding, a group of new pipeline components like those for

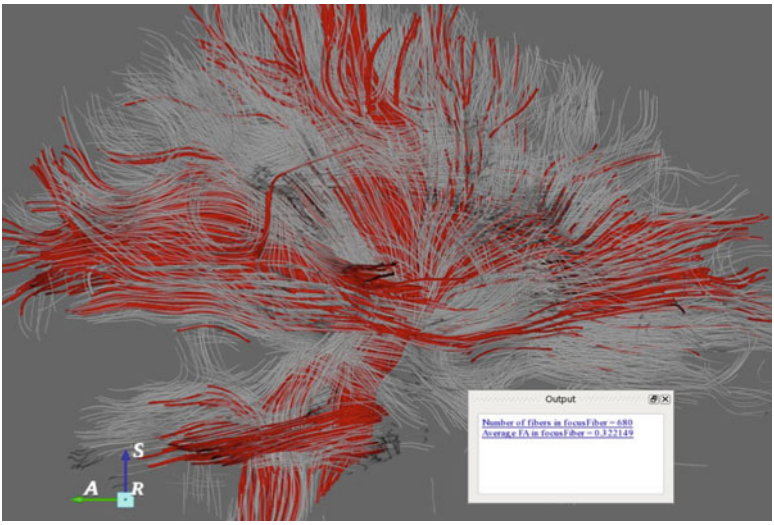

Fig. 4 Interface embedded to show query results

view-dependent per-vertex depth-value ordering has been added on top of related VTK classes, and some legacy VTK components have been tailored for specific needs of Gryphon visualizations.

In particular, the Gryphon script interpreter is implemented as data filters using the VTK visualization pipeline. As such, interpreting a Gryphon script involves translating the text, according to defined syntax and semantics, to data transformations in the VTK pipeline. For data encoding flexibility, multiple VTK data transformation pipelines have been employed.

Finally, the programming interface is implemented using Qt for C++. Interactions like triggering the execution of a Gryphon program, serializing and deserializing the text script, etc. are all developed with Qt widgets, although interactions with the visualization itself are handled using legacy VTK facilities with necessary extensions. Since our language targets non-programmer 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 employee an output window to prompt users with all error messages caused by invalid syntax or unrecognized language symbols. We have used GUI utilities of Qt for C++ to dump, after running a script, to process the error messages.

4 Scenarios of Use

We describe several sample task scenarios that can be performed by brain scientists on a brain DMRI model using the Gryphon language. The usage scenarios associated with the sample tasks are representative of typical real-world visualization tasks

of brain scientists with expertise in DMRI. The usages range from visualization customization and exploration to DMRI data analysis and cover the main language features and functionalities of the current Gryphon implementation.

In the following scenarios, a vascular neurologist (and a Gryphon end user), has a geometrical model derived from a brain DT-MRI data set and wants to com-pose and explore visualizations of the data for diagnostic purposes. In each of the scenarios, Lucy fulfills his task by programming a Gryphon script that describes his thinking process for that task and then clicks the Run button to execute the script. Lucy programs with Gryphon syntax references showing in 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.

4.1 Scenario 1: Examining ROIs

One common task brain scientists perform is to examine particular regions of interest (ROIs) rather than the whole brain. In this task, Lucy is interested only in all fibers within the temporal lobe area that belong to the CG bundle and CST fiber bundles in the parietal lobe area that have average LA value no greater than a threshold to be determined. The SELECT command with relative spatial operations using the anatomical planes enables Lucy to reach precisely the ROIs she desires. She first 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 bundle filters and conditional expression related to LA. Lucy begins with an estimate for the undecided LA threshold and then keeps refining until she gets the accurate selection of target fibers. In the end, she has a runnable script written in Gryphon (Fig. 5).

4.2 Calculating Metrics

In addition to visual examinations, neurologists often request quantitative investigations of their DMRI models. In this scenario, Lucy attempts to check white-matter integrity in a brain model to improve the limited reliability of DMRI tractography. For a rough estimation of the integrity, she 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 she has requested before, Lucy goes further to use it to kick out CST fibers with average FA below the bundle-wise average. Lucy writes the script in Fig. 6 to get the result.

```
LOAD "/home/lucy/braindti.data"
SELECT "axial +63.35"
SELECT "sagittal +70"
SELECT "coronal -48.5"
SELECT "sagittal -0.25"
SELECT "axial +7.2"
SELECT "CG"
SELECT "LA <= 0.275" in "CST"
```

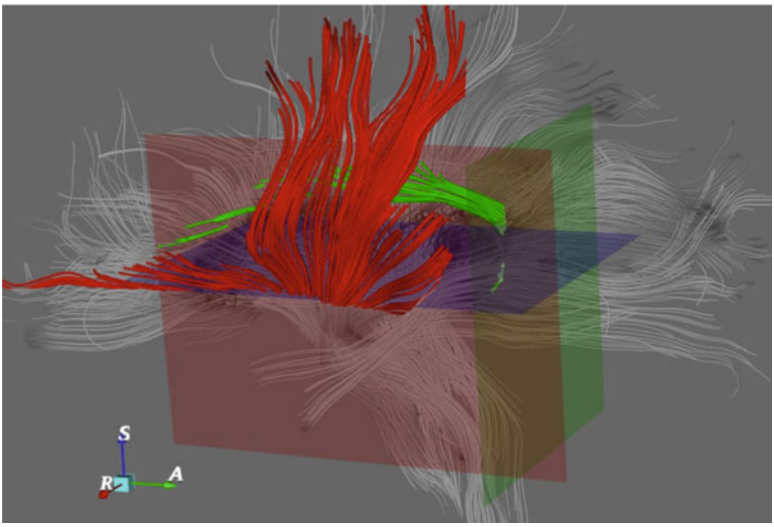

Fig. 5 Select regions of interest by relative unit specification

5 Discussion

5.1 Experimental Studies

The design was performed collaboratively with brain scientists. We performed a 2-h study with brain scientists. Generally, they liked its flexibility and found it easy to learn. They would like to encode more parameters calculated from their new studies. They would also prefer to have a graphical interface where these parameters can be put in directly.

5.2 Integrating Perceptual Principles

Current visualization methods in Gryphon are limited and are not guided fully by perceptual principles. In the long term, the Gryphon language should follow the rules that visualization should require minimal intervention on the part of the

```
LOAD "/home/lucy/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"
```

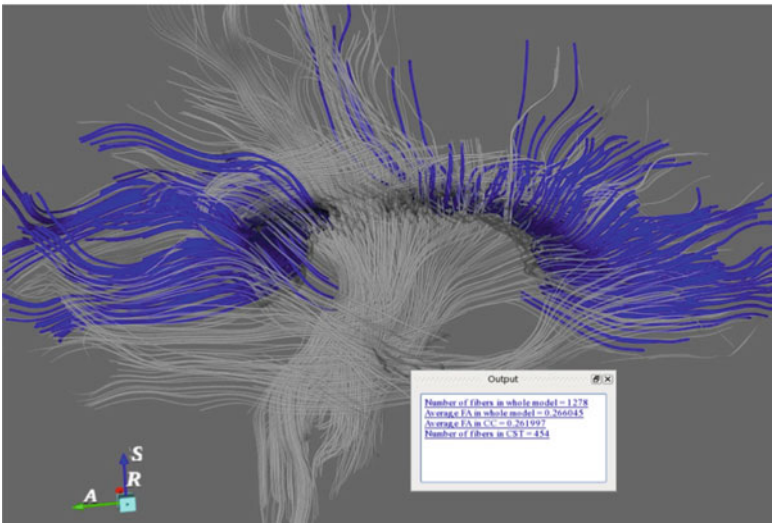

Fig. 6 Brain DMRI matrix calculation

designers, similar to that of APT. APT and its Tableau environment embodies a genuinely prescriptive theory of designing graphical encoding based on results from perceptual studies by analysis of data types. Similarly, ColorBrewer prescribes colors by data types, while providing a flexible user interface to adjust the values [14]. Our current implementation has limited availability of integrated principles, yet provides a language at a higher level so that brain scientists do not have to control low-level encoding details.

Several improvements include better color design, for example using new color-embedding methods to show fiber orientations [8]. The lighting needs to be improved to improve spatial structure presentation. Some perceptual-related rendering can be represented, e.g., ambient lighting to control dense line rendering for showing spatial relationships [24] or volume visualizations where structures are more pronounced [27]. It should be possible to generate ideas as such by visual composition in our Gryphon language.

5.3 Bidirectional Text (Programming Syntax) and Visualization Environment

Gryphon can be extended to rely on the coexistence of text display and visualization to form a bidirectional notation environment. By this we mean that the text and the visualization can augment each other, with the text becoming what is called "secondary notations" from software engineering and the visualization showing the results. The text scripts can be used to exhibit workflows and steps in arriving at the visualization that might otherwise be less accessible. On the other hand, the programmable notation is also our programming language that can carry a simple syntax based on brain scientists' tasks.

6 Conclusion

We have presented Gryphon, a simple domain-specific programming language for exploring 3D DMRI visualizations. We described the design process and results. Empirical studies suggested that user interfaces and more intuitive interaction techniques are desirable to have a more useful language.

Acknowledgements The authors thank the participants for their time and effort, Drs. Juebin Huang, Stephen Correia, and Judy James for their help on task analyses. We also thank Katrina Avery for her editorial support. This work was supported in part by NSF IIS-1018769, IIS-1016623, IIS-1017921, OCI-0923393, EPS-0903234, DBI-1062057, and CCF-1785542, and NIH (RO1-EB004155-01A1).

References

1. Akers, D.: CINCH: A cooperatively designed marking interface for 3d pathway selection. In: Proceedings of the 19th annual ACM symposium on User interface software and technology, pp. 33–42. ACM (2006)
2. Bentley, J.: Programming pearls: little languages. pp. 711–721. ACM (1986)
3. Bertin, J.: Semiology of graphics: diagrams, networks, maps (1983)
4. Bostock, M., Heer, J.: Protovis: A graphical toolkit for visualization. IEEE Transactions on Visualization and Computer Graphics **15**(6), 1121–1128 (2009)
5. Bostock, M., Ogievetsky, V., Heer, J.: D^3 data-driven documents. IEEE Transactions on Visualization and Computer Graphics **17**(12), 2301–2309 (2011)
6. Chiw, C., Kindlmann, G., Reppy, J., Samuels, L., Seltzer, N.: Diderot: a parallel dsl for image analysis and visualization. In: Proceedings of the 33rd ACM SIGPLAN Conference on Programming Language Design and Implementation, pp. 111–120. ACM (2012)
7. Correia, S., Lee, S., Voorn, T., Tate, D., Paul, R., Zhang, S., Salloway, S., Malloy, P., Laidlaw, D.: Quantitative tractography metrics of white matter integrity in diffusion-tensor mri. Neuroimage **42**(2), 568 (2008)

8. Demiralp, C., Hughes, J.F., Laidlaw, D.H.: Coloring 3D line fields using Boy's real projective plane immersion. IEEE Trans. on Visualization and Computer Graphics (Proc. Visualization '09) **15**(6), 1457–1463 (2009)
9. Demiralp, Ç., Zhang, S., Tate, D., Correia, S., Laidlaw, D.: Connectivity-aware sectional visualization of 3d dti volumes using perceptual flat-torus coloring and edge rendering. Eurographics (2006)
10. Elmqvist, N., Tsigas, P.: A taxonomy of 3d occlusion management for visualization. IEEE Transactions on Visualization and Computer Graphics **14**(5), 1095–1109 (2008)
11. Everts, M., Bekker, H., Roerdink, J., Isenberg, T.: Depth-dependent halos: Illustrative rendering of dense line data. IEEE Transactions on Visualization and Computer Graphics **15**(6), 1299–1306 (2009)
12. Forsberg, A., Chen, J., Laidlaw, D.: Comparing 3d vector field visualization methods: A user study. IEEE Transactions on Visualization and Computer Graphics **15**(6), 1219–1226 (2009)
13. Goldau, M., Wiebel, A., Hlawitschka, M., Scheuermann, G., Tittgemeyer, M.: Visualizing DTI parameters on boundary surfaces of white matter fiber bundles. In: Signal Processing, Pattern Recognition, and Applications/722: Computer Graphics and Imaging. ACTA Press (2011)
14. Harrower, M., Brewer, C.: Colorbrewer.org: an online tool for selecting colour schemes for maps. The Cartographic Journal **40**(1), 27–37 (2003)
15. Jiang, H., Van Zijl, P., Kim, J., Pearlson, G., Mori, S.: DtiStudio: resource program for diffusion tensor computation and fiber bundle tracking. Computer Methods and Programs in Biomedicine **81**(2), 106–116 (2006)
16. Johnson, C., Parker, S., Weinstein, D.: Large-scale computational science applications using the scirun problem solving environment. Citeseer (2000)
17. Koenig, M., Spindler, W., Rexilius, J., Jomier, J., Link, F., Peitgen, H.: Embedding vtk and itk into a visual programming and rapid prototyping platform. In: Proceedings of SPIE, vol. 6141, p. 61412O (2006)
18. Leemans, A., Jeurissen, B., Sijbers, J., Jones, D.: ExploreDTI: a graphical toolbox for processing, analyzing, and visualizing diffusion MR data. In: Proceedings 17th Scientific Meeting, International Society for Magnetic Resonance in Medicine, vol. 17 (2009)
19. Mackinlay, J.: Automating the design of graphical presentations of relational information. ACM Transactions on Graphics (TOG) **5**(2), 110–141 (1986)
20. Metoyer, R., Lee, B., Riche, N., Czerwinski, M.: Understanding the verbal language and structure of end-user descriptions of data visualizations. ACM CHI (2012)
21. Mori, S.: Introduction to diffusion tensor imaging. Elsevier Science (2007)
22. Mori, S., van Zijl, P.: Fiber tracking: principles and strategies–a technical review. pp. 468–480. Wiley Online Library (2002)
23. Myers, B., Pane, J., Ko, A.: Natural programming languages and environments. pp. 47–52. ACM (2004)
24. Peeters, T., Vilanova, A., Strijkers, G., ter Haar Romeny, B.: Visualization of the fibrous structure of the heart. In: Vision, Modeling and Visualization, pp. 309–316 (2006)
25. Pieper, S., Halle, M., Kikinis, R.: 3d slicer. In: IEEE International Symposium on Biomedical Imaging: Nano to Macro, pp. 632–635. IEEE (2004)
26. Reas, C., Fry, B.: Processing: a programming handbook for visual designers and artists. The MIT Press (2007)
27. Rheingans, P., Ebert, D.: Volume illustration: Nonphotorealistic rendering of volume models. IEEE Transactions on Visualization and Computer Graphics **7**(3), 253–264 (2001)
28. Rogowitz, B., Kalvin, A.: The which blair project: A quick visual method for evaluating perceptual color maps. In: Proceedings of the conference on Visualization, pp. 183–190. IEEE Computer Society (2001)
29. Roth, S., Kolojejchick, J., Mattis, J., Goldstein, J.: Interactive graphic design using automatic presentation knowledge. In: Intelligent User Interfaces, p. 237 (1998)
30. Stalling, D., Westerhoff, M., Hege, H., et al.: Amira: A highly interactive system for visual data analysis. pp. 749–67 (2005)

31. Tory, M., Moller, T.: Rethinking visualization: A high-level taxonomy. In: IEEE Symposium on Information Visualization, pp. 151–158. IEEE (2004)
32. Toussaint, N., Souplet, J., Fillard, P., et al.: Medinria: Medical image navigation and research tool by inria. In: Proceedings of MICCAI, vol. 7, pp. 1–8 (2007)
33. Wang, R., Wedeen, V.: Diffusion toolkit and trackvis. Proceedings of the International Society for Magnetic Resonance in Medicine **3720** (2007)
34. Wilkinson, L., Wills, G.: The grammar of graphics. Springer Verlag (2005)
35. Zheng, L., Wu, Y., Ma, K.: Perceptually based depth-ordering enhancement for direct volume rendering. IEEE Transactions on Visualization and Computer Graphics (to appear) (2012)