Programming by Sketch for Scientific Computing

Hanyu Liu
USM

Andrew Bragdon
Brown University

Attila Bergou
Brown University

Jian Chen*
USM

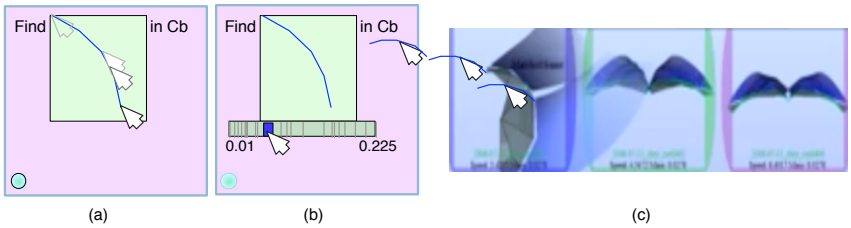

Figure 1: *(a). Programming-by-Sketch using our [verb] [objects] $<$ complement $>$ programming style; (b). a filter slider and the visualization windows in (c) pop up after executing the command line in (a); (c) shape can be further edited in place or by dragging a curve from the wing to the programming window.*

1 Introduction

Our long-term observations from working with bat biologists reveal that they often have to switch between multiple working environments, e.g., they use matlab to conduct analysis then port the results into a visualization system for confirmation. If more analysis is needed, they switch back to matlab to make changes. Too often, the separation between analysis and visualization caused by current systems can interrupt the analytical thinking process.

We present an early prototype of Programming-by-Sketch (Figure 1), which contributes a new way for scientists to explore their data in an interactive environment. The user can embed complex objects (such as shapes) in programs using simple syntax. We expand the idea of sketching for geometry match in three-dimensional (3D) environment [Wei et al. 2010] to a programming environment to allow more direct shape query by drawing, dragging, and filters.

2 Design

Our method combines and expands traditional syntax-based methods and sketch-based interfaces to allow users to embed shape sketches in their code, and hence it is called Programming-by-Sketch. The Programming-by-Sketch syntax, like our natural language, uses verbs to represent actions and objects to represent the verbs' targets. The follow notation is valid: *[verb] [objects]* $<$ *complement* $>$. Here, a valid vocabulary must include *[verb]* and *[object]*. If $<$ *complement* $>$ is not present, the program will search all data.

The rationale for providing this programming environment is to emphasize the human viewpoint. The syntax is a direct mapping from the task of querying to our programming langauge. We demonstrate a prototype and its use for bat wing shape analysis in a multiple-view environment. A meaningful analytic question in this field is to find when and where similar wing shapes are present during flight. Wing shape is measured using the curvature of the wing platform at the leading edge. The similarity measure uses the 2-norm distance between curvatures; and the curvatures are pre-computed and drawn at each wing beat at two predefined anatomical wrist and elbow locations.

For example, a biologist interested in finding a wing shape in *Cynopterus brachyotis* (a species of fruit-eating bats), can type *find* and specify a shape-of-interest (by drawing), followed by "in Cb " (which searches similar shapes only in the Cb species). Clicking the green button or *return* shows the results, and a filter slider also appears in the next line. The tick marks on the slider show all curvature distances. Images are displayed side-by-side to show the similar wing shapes ranked from small to large. Dragging the slider filters the shapes and updates the images shown on the right accordingly. The program can be excecuted line by line, as in interpretation languages, such as Matlab.

Here the shape-of-interest is defined not by coding numbers, but by direct manipulation of two kinds: users can sketch a curve in the window; alternatively, they can drag a wing shape directly from the bat flight visualization window (Figure 1(c)). The program automatically recognizes the user's intent and displays the shape of the current wingbeat next to the keyword *find* (Figure 1(b)). The gray background of the curve separates the syntax view and the shape view.

3 Pilot testing and results

We performed an informal two-hour study with one engineering scientist who was engaged in post-doctoral research at Brown University studying bat flight behavior. Our purpose was not to conduct a full user study but to gather preliminary comments on the design and whether it is sufficient for biologists' tasks. The participant found Programming-by-Sketch to be a new way to conduct analysis and liked the level of interactivity provided for queries. One suggestion was that the computational environment for handling shapes run in the background, but this may cause high subjective uncertainty on measurements. Then the UI should provide a data wrangling tool to let them define the shape measurement methods. Our informal discussion with bioinformatics researchers suggested they would like to use such a programming environment to query gene expression data.

References

WEI, J., WANG, C., YU, H., AND MA, K.-L. 2010. A sketch-based interface for classifying and visualizing vector fields. In *Proceedings of Pacific Visualization Symposium*.

*e-mail:jian.chen@usm.edu