pyNarian: A developing Python library for image processing and geometric morphometric trajectory analysis

K. James Soda[1], Dennis Slice[1]
[1]Department of Scientific Computing
Florida State University

Introduction

-Geometric morphometrics is a set of techniques for mathematically describing and analyzing the shape of structures without losing information on the structure's geometry (Adams et al 2013). Often such methods rely on the coordinates of a set of landmarks to characterize the specimen.
-Image processing, the retrieval of data from two-dimensional images, is a crucial step in many geometric morphometric studies. Yet almost all morphometric studies rely on a handful of programs for image processing. As a result, the routines that are available to researchers are often constrained to those available in these programs.
-One of the most common image processing programs used in morphometrics is tpsDig2 (Rolhf 2010). As a result, tpsDig2 has influenced the file formats commonly used in the field; the TPS file format is among the most common formats for storing the coordinates of landmarks.
-Geometric morphometrics is a growing discipline with new methods constantly being added. For instance, there has recently been increasing efforts to develop methods to analyze how the shapes of one or more specimens change over some period time, a field that could be termed shape trajectory analysis. These new methods can be statistically and computationally complex to the point of being inaccessible to many researchers. As a result, those developing these methods must often create software capable of implementing them.
-Here we introduce the library pyNarian. There are two major groups of routines in pyNarian, those for image processing and those for trajectory analysis. A major design goal for pyNarian is ease of use and extension. Therefore, it is written in the user-friendly language Python. However, it simultaneously can handle complex numerical and graphical operations through its incorporation of the libraries numpy (Oliphant 2007), matplotlib (Hunter 2007), and the Python Image Library (Pythonware 2012).

Image Processing

Key Features

-An image histogram lists the number of pixels in an image with each gray-level and can be a critical tool for selecting thresholds for image segmentation (da Fontoura Costa and Marcondes Cesar 2001). Based on an algorithm provided in da Fontoura Costa and Marcondes Cesar (2001), pyNarian generates ***image histograms*** for a desired image and outputs them in the form of a list.

-Two critical and intertwined tasks in image processing are image segmentation, that is the differentiation between an object of interest and its background, and edge detection,that is locating regions in an image where gray-levels rapidly change (da Fontoura Costa and Marcondes Cesar 2001). Based on algorithms in Pitas (2000), pyNarian provides two methods for image segmentation/edge detection, ***heuristic edge searches*** and ***Papert turtle***.

-There are already algorithms in tpsDig2 for automatically detecting complete outlines and converting these outlines to a set of landmarks. However, there is currently no way to ***automatically detect partial outlines***. Routines in pyNarian can automatically detect and record partial outlines between two landmarks and generate a TPS file for analysis in other programs. See Figure 1 for an example.

-The features described above are particularly designed with the goal of aiding in trajectory analysis.

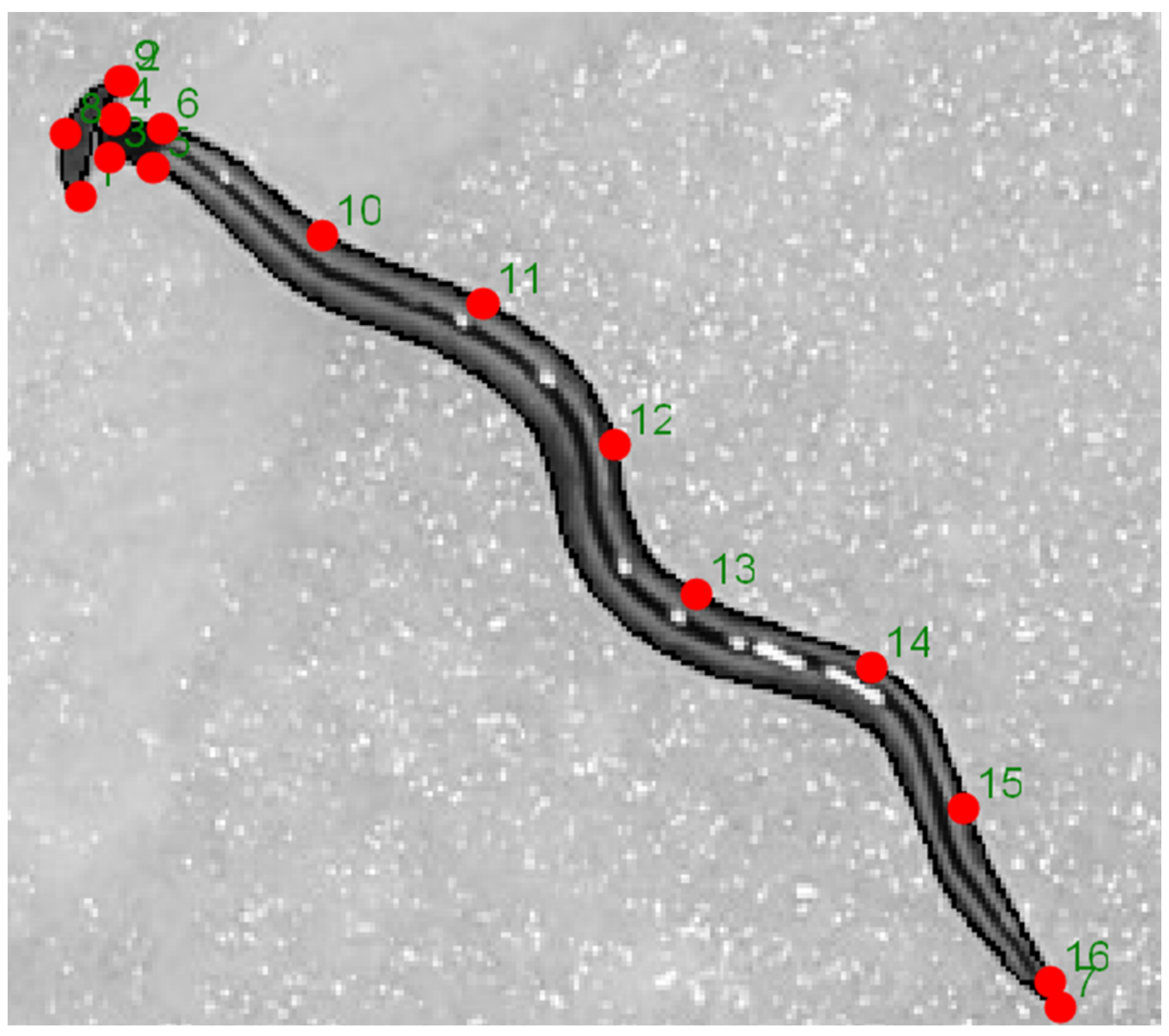

Figure 1. A partial outline, as viewed in tpsDig2, on the planarian *Bipalium* c.f. *vagum*. Using routines in pyNarian, the outline was detected, described using equally spaced points, and inserted into a standard TPS file processed in tpsDig2.

Trajectory Analysis

Key Features

-Upon completion of image processing, the user is able to begin trajectory analysis.

-Every trajectory is composed of multiple points in shape space, and every specimen can incorporate multiple trajectories. To address this hierarchical structure, pyNarian introduces three hierarchical data classes:

- The class ***PointTrajectory*** stores information about one shape in a shape trajectory.
- A series of PointTrajectory objects can then be provided to a ***DataTrajectory*** object, which will organize the PointTrajectory objects in sequential order to form a single trajectory through shape space.
- All the trajectories that belong to a single individual may be stored in a ***Specimen*** object.

-The pyNarian routine readTPS() allows for easy conversion of trajectory data into PointTrajectory, DataTrajectory, and Specimen objects.

-A 2D representation of the trajectory stored in each DataTrajectory object can be drawn using the function member drawTrajectory(). In the resulting diagram, a single point represents each point in the trajectory,and arrows indicate the order in which they appear. The color of each arrow indicates the number of time steps that separate each point in the trajectory. See Figure 2 for an example and explanation for each color. Every trajectory in a Specimen object can be drawn at once using the function member drawAllTrajectories().

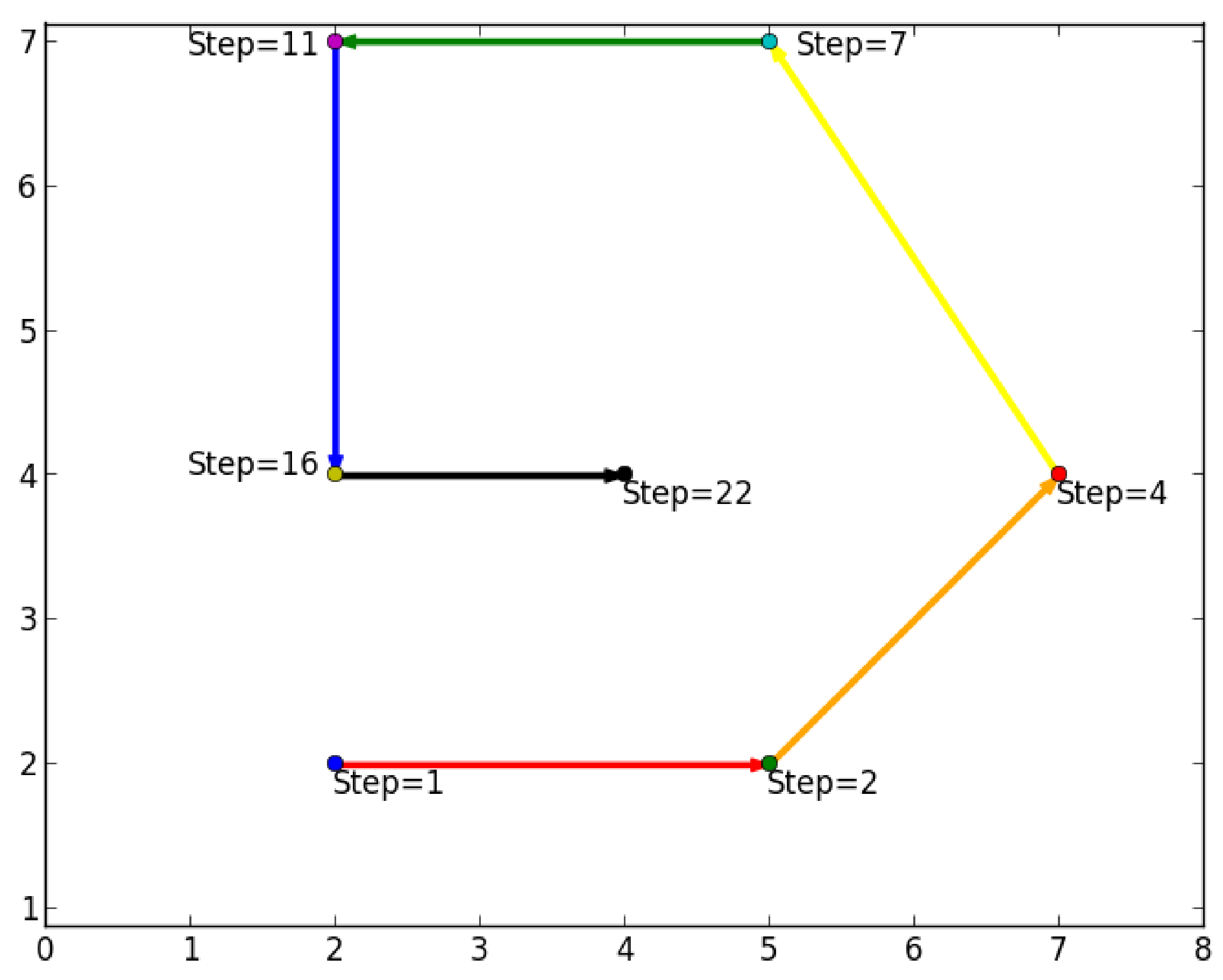

Figure 2. An example diagram illustrating how pyNarian draws trajectories through some representation of shape space. The color of the arrow illustrates how many time steps separate each data point (red = one step, orange = two steps, yellow = three steps, green = four steps, blue = five steps, black = over five steps).

References

- Adams, D.C., F.J. Rohlf, and D.E. Slice. 2013. A field comes of age: geometric morphometrics in the 21st century. Hystrix 24: 7-14.
- de Fontoura Costa, L. and Marcondes Cesar, R. 2001. Shape analysis and classification: theory and practice. CRC Press, Boca Raton, FL
- Hunter, J.D. 2007. Matplotlib: A 2D graphics environment. Computing in Science and Engineering 9: 90-95.
- Pitas, I. 2000. Digital processing algorithms and applications. New York: John Wiley and Sons, Inc.
- PythonWare. Python Imaging Library. PythonWare. Retrieved Feb 5, 2012, from http://www.pythonware.com/products/pil/
- Rohlf, F.J. 2010. tpsDig (ver. 2.16) [software]. Retrieved from http://life.bio.sunysb.edu/morph/.
- van Rossum, G. and F. L. Drake (eds). 2001. Python reference manual. Virginia, USA: PythonLabs. Available at http://www.python.org.