

COILING GEOMETRY OF PROBOSCIDEAN TUSKS

Senior Honors Thesis

Submitted for the degree of Bachelor of Science in Geological Sciences with Honors

By Alistair Hayden

Faculty Advisor: Daniel C. Fisher

Department of Geological Sciences

University of Michigan

August, 2011

Chapters

I.	Introduction	4
II.	Construction of a Digital Model	6
	a. Photogrammetry Using PhotoModeler Scanner	7
	b. Laser Scanning Using a HandyScan	13
III.	Growing a Tusk with Shell Growth Models.....	16
	a. Traditional Logarithmic Spiral as a Model of Central Structural Axis	16
	b. Ackerly Model to Describe a Tusk Surface	17
	c. Additional Free Parameters Used	17
IV.	Fitting a Generated Tusk Model to a Digitized Real Tusk	19
	a. Manual Optimization with a Graphical User Interface (GUI)	19
	b. Computer Optimization with Nonlinear Regression in MATLAB.....	20
	c. Optimization/GUI Technique	22
V.	Tusk Analysis.....	23
	a. Sources of Digitized Models	23
	b. Parameter Relationships	23
	c. Describing Tusk Coiling Geometrically	25
	d. Species and Sex Differences Expressed in Parameter Sets	28
	e. Discussion.....	29
	f. Future Work	30
VI.	Conclusion	32
	Acknowledgements	32
	Works Cited	32
	Appendix A – Construction of a Digital Model, Supplement	33
	Appendix B – Real and Model Tusks	34
	Appendix C – MATLAB Code Used	41
	<i>logtusk</i>	41
	<i>ackerlyfun</i>	42
	<i>newieGUI</i>	43
	<i>optimtuskfun</i>	48
	<i>ackerlyslowfun</i>	49
	<i>rotor</i>	51
	<i>circlepoints</i>	51

Tables, Figures, and Equations

Table 1 – Specimen names and select information are shown for all tusks in this study.....	5
Figure 1 – Pevek 1, as seen in an image used to generate its model	9
Figure 2 – Pevek 7, as seen in an image used to generate its model	9
Figure 3 a-g – Low-resolution screen shots of texture-mapped models of Pevek 1-7	12
Figure 4a&b – The right tusks of the Buesching and Hyde Park mastodons	14
Table 2 – The tusks and their methods of digitization.....	15
Eq 1 – Describing the logarithmic spiral model	16
Table 3 – A description of each parameter used in each model	18
Figure 5 – A screenshot of the GUI being used.....	20
Table 4 – Optimized tusk parameters for the seven tusks from Pevek	24

Eq 2 – Describing the relationship between δ and α 24

Figure 6 – Graphical relationship between α and δ 25

Table 5 – Parameters generating the best fits using the Ackerly model 26

Figure 7 – Pevek 6 with red lines indicating obvious twisting grooves..... 27

Figure 8 – An Ackerly-style model based on the parameters of Pevek 6 demonstrating twist.. 27

Table 6 - The photogrammetric parameters used to generate tusk models 34

Figure 9 – Adult *M. primigenius* tusks and their Ackerly-style models..... 34

Figure 10 – The *M. columbi* tusks and their Ackerly-style models..... 36

Figure 11 – The *M. americanum* tusks and their Ackerly-style models 37

Figure 12 – The remaining models that did not have good fits from shell-growth models..... 38

I. Introduction

The explanation of forms in the natural world using simple mathematical descriptions is one of the oldest pursuits of both biology and mathematics. The branch of math called geometry dating back to the Greeks even derives its name from measuring the Earth. An exciting development that allowed the description of many more forms was the discovery of logarithms based on the natural number e . The classic shape to use this number is the logarithmic spiral, a shape with many unique properties such as self-similarity that made it a good candidate for describing natural forms.

One of the most familiar logarithmic spirals in nature is the coiled shell, a form that has been explored in detail for over a hundred years (Moseley, 1838), (Thompson, 1917). Though specific morphologies have been described for this long, the full range of possible morphologies described by this simple spiral was first investigated by Raup, who quantitatively explored this coiling morphospace and found that not all possible coiled forms exist in nature (Raup, 1966), (Raup & Michelson, 1965). Later work based on that of Raup took different approaches that allowed description of even more coiling types, including more complex, allometric forms that had proven challenging using earlier models (Ackerly, 1989a), (Okamoto, 1988).

Tusks, such as those of elephants and other proboscideans, are also famous for their coiled form. Similarly to molluscan shells, they grow larger by continuous deposition of mineralized tissue throughout the life of an individual and have a distinctive, species-specific shape. Both shells and tusks form as stacks of cones, somewhat like a stack of ice cream cones. Tusks are solid because their component cones extend all the way to an apex, while molluscan shells are hollow because their cones effectively lack apices. In addition to the coiled form, another hallmark of tusks is that they have a series of grooves or ridges running from tip to base that twist around the surface of the tusk as it coils. Another characteristic feature of tusks, and one that is clearly anisometric and therefore different from

the other characters, is girth. The girth of a tusk increases when the animal is young but then stabilizes and eventually begins to decrease in old age, an ontogenetic change that is difficult to reconcile with the apparent constancy of coiling and twisting.

Just as with shells, there is a wide range of tusks forms available to study. They come from elephants and their many proboscidean relatives that have existed across the world for more than 40 million years, yielding quite diverse forms. Some proboscidean species are clearly distinct, such as Woolly mammoths and African elephants, but others can prove more challenging to distinguish. The Jeffersonian mammoth is one species that has proven difficult to classify. Some treat it as a separate species, but there is growing evidence that it is actually a hybrid between Woolly and Columbian mammoths. Sex classification can also be confusing at times. Some techniques for identifying sex are based on size, which can quickly become confusing in individuals that exhibit island dwarfing, such as those from the St. Paul Island population of Woolly mammoth.

The main goal of this work was to attempt description of the three main tusk characters, coiled form, twist, and girth change, using the logarithmic models previously developed to describe coiled shell growth and form. Other goals involved using these mathematical tusk descriptions to quantify sexual dimorphism and species-specific morphologies, especially in some of the difficult cases, like the Jeffersonian mammoth. To accomplish this, tusk models were generated using shell growth equations based on 22 digitized tusks representing 18 individuals (Table 1). The tusks came from adults and juveniles of both sexes and were from fifteen Woolly mammoths (*M. primigenius*), one Columbian mammoth (*M. columbi*), and three American mastodons (*M. americanum*). While the coiled tusk form could be accurately described by logarithmic models, with the twisting grooves being a byproduct of coiled growth, girth proved more challenging to model. Certain growth parameters were found to accurately describe the taxa and sex of some adult individuals.

Specimen Name	Species	Gender	Age	Right/Left	Source	Time
<i>Pevek 1</i>	<i>M. primigenius</i>	Female	Adult	Left	Wrangell Island,	12-4 ka

					Siberia	
<i>Pevek 2</i>	<i>M. primigenius</i>	Male	Adult	Left	Wrangell Island, Siberia	12-4 ka
<i>Pevek 3</i>	<i>M. primigenius</i>	Male	Adult	Left	Wrangell Island, Siberia	12-4 ka
<i>Pevek 4</i>	<i>M. primigenius</i>	Male	Adult	Left	Wrangell Island, Siberia	12-4 ka
<i>Pevek 5</i>	<i>M. primigenius</i>	Female	Adult	Left	Wrangell Island, Siberia	12-4 ka
<i>Pevek 6</i>	<i>M. primigenius</i>	Male	Adult	Left	Wrangell Island, Siberia	12-4 ka
<i>Pevek 7</i>	<i>M. primigenius</i>	Male	Adult	Right	Wrangell Island, Siberia	12-4 ka
<i>2000-245</i>	<i>M. primigenius</i>	???	8-10	Left	Taymyr Peninsula, Siberia	10-40 ka
<i>2000-246</i>	<i>M. primigenius</i>	Female	8-10	Left	Taymyr Peninsula, Siberia	10-40 ka
<i>2000-286</i>	<i>M. primigenius</i>	Male?	8-10	Right	Taymyr Peninsula, Siberia	10-40 ka
<i>ZCHM 19</i>	<i>M. primigenius</i>	Female?	10	???	Chukotka Peninsula, Siberia	10-40 ka
<i>ZCHM 20</i>	<i>M. primigenius</i>	Female?	11-12	???	Chukotka Peninsula, Siberia	10-40 ka
<i>ZCHM 22</i>	<i>M. primigenius</i>	Male	6	Right	Chukotka Peninsula, Siberia	10-40 ka
<i>ZCHM 23</i>	<i>M. primigenius</i>	Male	6	Left	Chukotka Peninsula, Siberia	10-40 ka
<i>St Paul</i>	<i>M. primigenius</i>	Male	Late 20s		St. Paul Island, AK	Late Holo
<i>UWY-R</i>	<i>M. columbi</i>	Male	Adult	Right	Rawlins, WY	12 ka
<i>UWY-L</i>	<i>M. columbi</i>	Male	Adult	Left	Rawlins, WY	12 ka
<i>HP-R (Hyde Park)</i>	<i>M. americanum</i>	Male	30-40	Right	Hyde Park, NY	12 ka
<i>HP-L (Hyde Park)</i>	<i>M. americanum</i>	Male	30-40	Left	Hyde Park, NY	12 ka
<i>B-R (Buesching)</i>	<i>M. americanum</i>	Male	Adult	Right	Ft. Wayne, IN	11 ka
<i>B-L (Buesching)</i>	<i>M. americanum</i>	Male	Adult	Left	Ft. Wayne, IN	11 ka
<i>Bothwell 2-14</i>	<i>M. americanum</i>	Female	22	Left	Hebron, IN	11 ka

Table 1 – Specimen names and select information are shown for all tusks in this study. Names ending in ‘-R’ are right tusks, and ‘-L’ indicates a left tusk. Specimen names beginning with the same designation (i.e. ‘Pevek’ or ‘2000’) were part of the same collection. Times expressed as a range are for tusks that have yet to be dated and reflect the current known age-range of individuals from that location.

II. Construction of a Digital Model

Many techniques for digitizing real forms exist, but the cheapest and most portable method is photogrammetry, which is computer-aided 3D reconstruction from 2D images. The major advantage to this method is that the 2D images can be obtained with cheap consumer cameras that can be easily

carried to the field, with the 3D processing done later. There are also high-resolution laser scanners, which require less time to use than photogrammetry but are usually expensive and therefore difficult to justify taking to the field where damage could occur. Another option for digitization is contact digitization, a process in which points are manually digitized by moving a probe along the surface, but this technique requires a great deal of time to get a model with high point density and is also not very portable. For this work, models were generated from field images by photogrammetry and from in-house tusks by laser scanning; additional models generated previously by contact and laser scanning methods were also analyzed.

Photogrammetry Using PhotoModeler Scanner

Several programs exist to do photogrammetric reconstruction from images taken with a consumer digital camera. There is a free option developed by Microsoft called PhotoSynth that creates a 3D model of a scene from just a set of images of an object and no other information; however, the images and resultant models then become property of Microsoft. PhotoSynth effectively functions as a simple interface for another software package, Bundler, which when used independently requires significantly more user input but allows the user to retain rights to the models. For this work, we used PhotoModeler Scanner, which also preserves user rights but requires less user input than Bundler. A project using this software generally requires taking many images of the object from different angles, cross-referencing points in the images so that the program can identify the shooting angle (this is generally aided by preparing the scene with identifiable points), trimming out parts of the images that are not necessary, then building the model. Details vary between projects.

In this study, seven tusk models were built using PhotoModeler Scanner from photographs of tusks at the Pevek Museum, in Chukotka, in northeast Siberia. The photographs were taken by Dan Fisher, who also carefully prepared the tusks to facilitate the photogrammetry. The software was able

to take the multitude of 2D images and process them into 3D tusk models. All these models had a high point density, with more than 10,000 points defining the surface of each tusk.

Tusks received different degrees of preparation before imaging because it was unclear whether marked tape would help 3D reconstruction by providing easily-identifiable points or hinder it by hiding natural texture from the pattern-matching algorithms. Individual tusk preparation varied from the most intense treatment of wrapping various parts of the tusk in strips of masking tape with identifying marks, to minimal treatment of a few, widely spaced squares of marked masking tape. Further preparation common to all tusks involved placing them on a floor with clear markings, which had many distinct points that could be recognized as identical between images. Other objects that also had distinctly recognizable points separated by a known distance, such as prepared cubes, could be placed and used to scale the images (see Figures 1 and 2 for images of prepared tusks).

The recognizable points were important because cross-referenced points in an image pair allowed the program to do a Bundle Adjustment to determine the shooting location of the camera for each photograph. Minimally, PhotoModeler Scanner required six different cross-referenced points in each image-pair. Because the large tusks could obscure some points in some views, close to twenty reference points were identified in each project, ensuring the minimum of six was met. The extra points also helped increase the accuracy of the Bundle Adjustment. In any project, the best configuration of reference points is one in which each photo has visible reference points surrounding the object of interest. This is achieved most easily by strategic placement of identifiable landmarks. Elevated reference points were especially useful for the large tusks because they could be visible, even when positioned behind a tusk.



Figure 1 – Pevk 1, the most extensively prepared tusk, as seen in an image used to generate its model. This is taken from the highest camera height and includes all the points used as references. The cubes are 8 cm on a side and each had the top four corners used as references and to scale the project. The centers of the geometric starbursts were also used as references. Photo and tusk preparation by Dan Fisher.



Figure 2 – Pevk 7, one of the minimally-prepared tusks, as seen in an image used to generate its model. This is taken from the lowest camera height and again includes all the points used as references. However, since the tusk is so big, the reference points behind it are obscured, except for those of the corners of the elevated cube on the right. The tusk was placed in the same location as Pevk 1 in Figure 1; the cubes are still 8 cm on a side. Photo and tusk preparation by Dan Fisher.

Once the reference points were established for a project, the tusk could not be moved with respect to them, meaning the floor obscured half the tusk. To image the full tusk, it was flipped and a

second, independent set of images was created with its own internally consistent reference points. The resultant half-models were later combined to create a whole model for that tusk. The photoset for each half-model contained images taken at 32 different positions spanning 360 degrees in azimuth around the tusk from 3 different camera heights, yielding a total of almost 200 photos per tusk. Because each photo requires processing before use, using them all would require an extremely large amount of time. However, using fewer photos in a project reduces the resolution and density of points in the final model. After much experimentation, a good balance between model resolution and time invested was the use of every other photo from the highest and lowest camera heights. This meant 32 photos (16 spaced by 22.5° at each of 2 heights) were used for each half-model. Once the photos for a project were selected, the prepared points could be cross-reference in order to do the Bundle Adjustment.

The photos also had to be corrected to remove distortion introduced by the camera lens; in PhotoModeler Scanner, this process is called idealization. This correction is based on camera-specific data that either exist in the program or are obtained through camera calibration (see Appendix A for details). The idealization only required a click of a button from the user, but required intensive use of the CPU, so the program sometimes crashed if too many other computer processes were running at the same time or if PhotoModeler Scanner had been running for some time. Therefore, before idealizing, it was good practice to restart the program and quit any other applications.

The next step was to have the program match points between photos and triangulate their locations in 3D space. A trim was used to mask out parts of the photo of low importance, decreasing the number of points to locate and thereby reducing computer run time. The point-matching algorithms were surprisingly complex and involved user-input parameters that appeared to have minimal physical significance and are therefore best described qualitatively. The optimal set of parameters differed for each project and perhaps even for each photograph pair, but for convenience, a single parameter set was employed in each project. The parameters adjusted in the course of this work were sampling rate,

matching region radius, and texture number. Sampling rate set the distance between points on the model, measured in units that were only defined after the project scale was set. Therefore, a high value for sampling rate decreased point density by increasing the spacing of points in the model. The matching region radius defined the size of the area that had its pixel values compared between the matching regions of the paired images to recognize points that were the same. When regions matched, the location of the center was triangulated and became a point on the model. A larger value generally meant that there was more smoothing and less noise. It was effective to change this variable in conjunction with the sampling rate to find an optimum of low noise and high point density. The final important adjustable parameter was 'texture', and according to the program documentation, accounted for variability in the texture (coloring pattern) of the object. When increased, it tended to drastically decrease point density, perhaps because the coloring patterns of the tusks were fairly constant, but other projects with irregular textures might benefit from increases in this parameter. It cannot be stressed enough that each project done in PhotoModeler Scanner has different optimum parameter values, but these tusk digitizations were all similar enough that the best parameter sets were similar, though not identical, between projects.

Parameter sets were evaluated by running the point matching process several times on a single photo pair. The best set of parameters was the one that accurately resolved a large number of points in a relatively short time. Since there were 32 photo pairs in each half-model, small differences in the number of points and run time could add up to big differences in resolution and time consumption over the course of the project. A good starting point for any parameter set evaluation is a matching region radius of 20 and a texture of 1, with the sampling rate left up to the user [Dan Miller, personal communication]. See Appendix A for additional comments on parameter selection.

Once the two half-models for a tusk were created, they had to be merged to create the complete tusk. This was done by marking at least three points that were the same between the two

half-models and giving each point pair its own unique name. The tape preparation was especially useful here because the sharp corners or markings on the tape could be identified easily. Once several points were marked, one model was merged into the other, with the labeled points telling the program how to combine them. Refer to Table 6 in Appendix A for a complete listing of the tusk models by name, with the settings used, the number of points generated, and the final overall relative model quality. For this work, the macroscopic coiled tusk form was of main concern, so even the lowest-quality models were considered successful because they captured this, though they failed to capture fine-scale surface topography. See Figure 3 for examples of the final models.



Figure 3 a-g – These are low-resolution screen shots of the texture-mapped photogrammetric models of tusks Pevek 1-7, NOT shown to scale (each is 1-2m long). The occasional white dots are the cross-referenced points used in the bundle adjustment and surround the tusk in most views.

Minimally, each tusk required about 20 minutes of manual cross-referencing, 40 minutes of run-time to complete the Bundle Adjustment, 30 minutes of manual masking, 30-60 minutes of run-time to create both half-models for a tusk, and 30 minutes of manual merging and trimming for a total of about 3 hours. In practice, it usually took around 4 hours or longer because different settings had to be explored for each tusk. The intensive CPU usage sometimes also caused program crashes, losing data and increasing the processing time.

Laser Scanning With A HandyScan

Another method for creating digital models of a tusk was to use a laser scanner such as the HandyScan. The device was self-orienting, which meant it could determine where it was scanning by recognizing special reflective dots manually placed 4-10 cm apart on the item to be scanned (if it was large, like a tusk) or its background (if it was smaller, like a tusk tip). The dots were placed to cover the tusk surface without a regular pattern, so the configuration visible at each surface point is identified uniquely. The dots were placed closely enough that at least four to five of them could be seen by the laser scanner from any given position, but far enough apart that the software was able to distinguish them. More dots than necessary were placed because they frequently fell off, especially on the casts that had been treated with an anti-stick substance to facilitate removal from their molds. Dots could also be added without issue during scanning, but fallen dots were not replaced because they could not be replaced accurately. Inaccurate dot placement would introduce errors into location determination.

Once most of the dots were scanned in, the surface itself could be scanned. The software determined the surface by looking at the shape of the laser crosshairs, which changed based on the surface onto which they were projected. Complicated surface topography was difficult for the software to reconstruct, and so the program just left a hole when it was unable to determine the shape. These holes could generally be reduced in size or eliminated by scanning the troublesome spot from different

directions, angles of incidence, and scanner roll orientations. However, some topographies, such as sharp edges, could prove tricky to scan, even with several passes with the scanner in different orientations. Because doing too many scans could overflow the CPU memory, holes that could not be filled after several passes were better filled using the 'Cap Holes' function in 3D Studio Max. The final hole-free models achieved very high resolutions; the laser-scanned tusks had surfaces of upwards of 100,000 unique points. See Figure 4 for screenshots of the rendered models generated from laser scans.



Figure 4a&b – The right tusks of the Buesching (a) and Hyde Park (b) mastodons, NOT to scale (each is close to 2m long). Notice the intricate surface detail captured by the laser scanner, especially on the Buesching tusk.

Once the 3D files were created (generally as an *.stl or some other type of file native to a 3D rendering software package like 3D Studio Max), the best way to get them into MATLAB was to export them from a rendering application as an ASCII Scene Rendering file. This format identified points on the surface and defined how to connect them to make the facets used in 3D software. Therefore, to get a file of pure point coordinates that was easily readable by MATLAB, the extraneous header and facet definitions were deleted in a text editor, a process that also drastically reduced the size of the file. This new text file could be parsed easily by MATLAB (using *uiimport* for example), and then the surface data could be processed from there. In order to avoid performing this procedure every time the model was needed, it was useful to convert the imported surface data points to single precision and paste them into a MATLAB script, known as an mfile, that could then be called to generate the tusk data model with a single command. See Table 2 for information on the models generated and used in this project.

Specimen Name	Digitization	Raw NOP	Downsampled NOP	Digitization Source
<i>Pevek 1</i>	<i>Photogrammetry</i>	18,000	3,300	This work
<i>Pevek 2</i>	<i>Photogrammetry</i>	50,000	3,400	This work
<i>Pevek 3</i>	<i>Photogrammetry</i>	86,000	3,500	This work
<i>Pevek 4</i>	<i>Photogrammetry</i>	46,000	3,400	This work
<i>Pevek 5</i>	<i>Photogrammetry</i>	25,000	3,300	This work
<i>Pevek 6</i>	<i>Photogrammetry</i>	80,000	3,500	This work
<i>Pevek 7</i>	<i>Photogrammetry</i>	15,000	3,400	This work
<i>2000-245</i>	<i>MicroScribe</i>	4,500	2,200	Adam Rountrey, UM
<i>2000-246</i>	<i>MicroScribe</i>	3,100	3,100	Adam Rountrey, UM
<i>2000-286</i>	<i>MicroScribe</i>	3,200	3,200	Adam Rountrey, UM
<i>ZCHM 19</i>	<i>MicroScribe</i>	3,500	3,500	Adam Rountrey, UM
<i>ZCHM 20</i>	<i>MicroScribe</i>	4,300	4,300	Adam Rountrey, UM
<i>ZCHM 22</i>	<i>MicroScribe</i>	12,000	3,000	Adam Rountrey, UM
<i>ZCHM 23</i>	<i>MicroScribe</i>	15,300	3,100	Adam Rountrey, UM
<i>St Paul</i>	<i>MicroScribe</i>	16,500	3,300	Randy Tedor, U of AK
<i>Bothwell 2-14</i>	<i>MicroScribe</i>	5,100	5,100	Dan Fisher, UM
<i>UWY-R</i>	<i>HandyScan</i>	25,300	2,500	Dan Fisher, UM
<i>UWY-L</i>	<i>HandyScan</i>	35,600	3,600	Dan Fisher, UM
<i>HP-R (Hyde Park)</i>	<i>HandyScan</i>	126,700	3,400	This work
<i>HP-L (Hyde Park)</i>	<i>HandyScan</i>	414,800	3,200	Dan Fisher's Lab, UM
<i>B-R (Buesching)</i>	<i>HandyScan</i>	123,700	3,100	This work
<i>B-L (Buesching)</i>	<i>MicroScribe</i>	3,500	3,500	Dan Fisher

Table 2 – The tusks and their methods of digitization, the number of points (NOP) before and after downsampling the points (done to speed the analysis of Chapter V), and the source of the digitization.

III. Growing a Tusk Using Shell Growth Models

Coiled shell growth has long been known to be described by a logarithmic spiral form (Thompson, 1917). One body of work that did much to reinvigorate investigation of this topic introduced modern computer simulation technology and described coiled shell growth as a circular curve enlarging and spiraling around a coiling axis (Raup, 1966). Because actual shell growth occurs relative to the existing shell and not to an abstract external coiling axis, this model was modified to mimic biological reality more closely by describing the propagation of the curve in a moving reference frame intrinsic to the shell (Ackerly, 1989a). Complicated shell forms were explained by incorporating differential geometry, the calculus-based study of curved spaces employed in fields such as general relativity, and allometric growth into the internal reference frame models (Okamoto, 1988). This work seeks to describe tusk coiled form, groove/ridge twist, and girth change with a logarithmic spiral form. Several models including one based on Raup's work were initially investigated to gain familiarity with them and with programming functions in MATLAB, but only a simple logarithmic spiral model and an Ackerly-style surface model were used in the tusk analysis described in Chapter V.

Traditional Logarithmic Spiral as a Model of the Central Structural Axis

The traditional logarithmic spiral was used to test and gain familiarity with the optimization and GUI software because of its simplicity. Because it is just a spiral line without a surface it was only intended to describe the central structural axis of a coiled form and be an initial indicator of whether tusks could be described logarithmically. The model is based on the traditional Cartesian 2D logarithmic spiral equation, except that it has additional parameters to stretch it into 3D space (Eq 1). The code for the MATLAB mfile that builds the model, *logtusk*, is presented in Appendix C.

$$\begin{cases} x = Ae^{Dt} \cos t \\ y = Be^{Et} \sin t \\ z = Ce^{Ft} \end{cases} \quad \text{Eq 1}$$

Parameters A, B, and C when increased, stretch the tusk spiral in the x, y, and z directions, respectively, by a scale factor, while parameters D, E, and F stretch the tusk spiral in those directions by changing the size of the growth step. Therefore, when $A = B$ and $D = E$ this reduces to a traditional logarithmic spiral in the xy plane with C and F defining how much it stretches in the z direction; their signs define coil handedness.

Ackerly Model to Describe a Tusk Surface

To describe the twisting grooves and ridges on the surface of a tusk, the generated tusk model had to have a surface rather than just a coiled axis. The Ackerly model was selected to create this because it describes growth using an internal reference frame that is likely to be biologically realistic. The model is based on a paper by Ackerly (Ackerly, 1989a), Appendix B of which was useful in writing the code. The code for the MATLAB mfile that generates this model, *ackerlyfun*, is presented in Appendix C of this document.

To summarize, the model takes a circular aperture and successively enlarges, translates, then rotates it based on the values of the parameters. The succession of consecutive positions of the aperture generates a coiled surface. The aperture size changes based on δ , which is the apical angle of the cone describing the tusk surface, and since the translation and rotation magnitudes are affected by the aperture radius, δ also affects the tightness of the 2D coil. The parameter α also describes how tightly the 2D logarithmic spiral coils while γ defines the angle between the growth direction and the horizontal and thereby determines how much the coil spirals into 3D space. The parameters that define the difference in orientation between the aperture and the translation direction, σ and Φ , were not found to improve the fit of the generated model and were therefore ignored. The starting aperture radius r_0 was also found to function as a scale factor for the entire tusk.

Additional Free Parameters Used

In addition to the model-specific parameters that describe the form of the generated tusk, there are several other parameters to describe its length and spatial orientation. These additional parameters are a length parameter to describe how far the tusk has grown, an erosion parameter to remove part of the tip as might naturally occur, three rotation parameters, and three translation parameters, for a total of eight additional parameters. These transformation parameters were necessary for aligning the generated and digitized models, but were not actually used in the tusk analysis since they just describe position and orientation of the tusk in space. Refer to Table 3 for a summary of these parameters and their uses. The length and erosion parameters were fixed to extend beyond the tusk during optimization, described in Chapter IV, because the optimizer just looks at the nearest points to the digitized tusk and ignores anything beyond it.

Parameter	Description
<i>Logarithmic Spiral Model</i>	
A/B/C	Stretch factor in the x/y/z directions
D/E/F	Step size in the x/y/z directions
<i>Ackerly Surface Model (as defined in Ackerly 1989a)</i>	
α	'Tightness' of 2D coil
δ	Angle of cone describing tusk surface; also affects coiling
γ	3D angle of vertical coiling
σ/ Φ	Angles between aperture orientation and growth direction
r_0	Initial radius
<i>Parameters for all Models</i>	
Length	How far the tusk has grown
Start Point	Where along the generated model the digitized tusk tip is, to simulate erosion
$\theta/\varphi/\psi$	Rotation
X/Y/Z	Translation

Table 3 – A description of each parameter used in each model.

IV. Fitting a Generated Model to a Digitized Real Tusk

Optimization involves finding an extreme value of a function, usually considered the “best” value that describes something especially well. An optimization program seeks out a minimum value of a function much as water seeks out a minimum value on a topographic surface. Like water on a landscape, an optimizer can get stuck in a local minimum, yielding a mountaintop lake in the water analogy, rather than a global minimum. Fortunately, optimizers are not constrained to flow to adjacent points and can be constructed to reduce the likelihood of getting stuck in a mountaintop lake. The more parameters that must be optimized, the more chances there are of getting trapped in a local minimum instead of the global minimum. As with water, the closer the optimizer starts to the global minimum the more likely it is to be found, so a good starting guess is quite important to locating the global minimum.

Manual Optimization With A Graphical User Interface (GUI)

A Graphical User Interface (GUI) called *newieGUI* was developed for this work to help visualize the digitized tusks and explore how well they were described by the shell growth models (see Figure 5 for its appearance and Appendix C for code and usage). Sliders control the free parameter values and therefore shape and spatial orientation of the generated model, dropdown menus provide selections of both the digitized tusk and of the shell growth model to fit to it, and the quality of the fit can be visually inspected by rotating the scene in the center. This visualization is very important because it helps achieve a good starting estimate of tusk form and review the output of computer optimization.

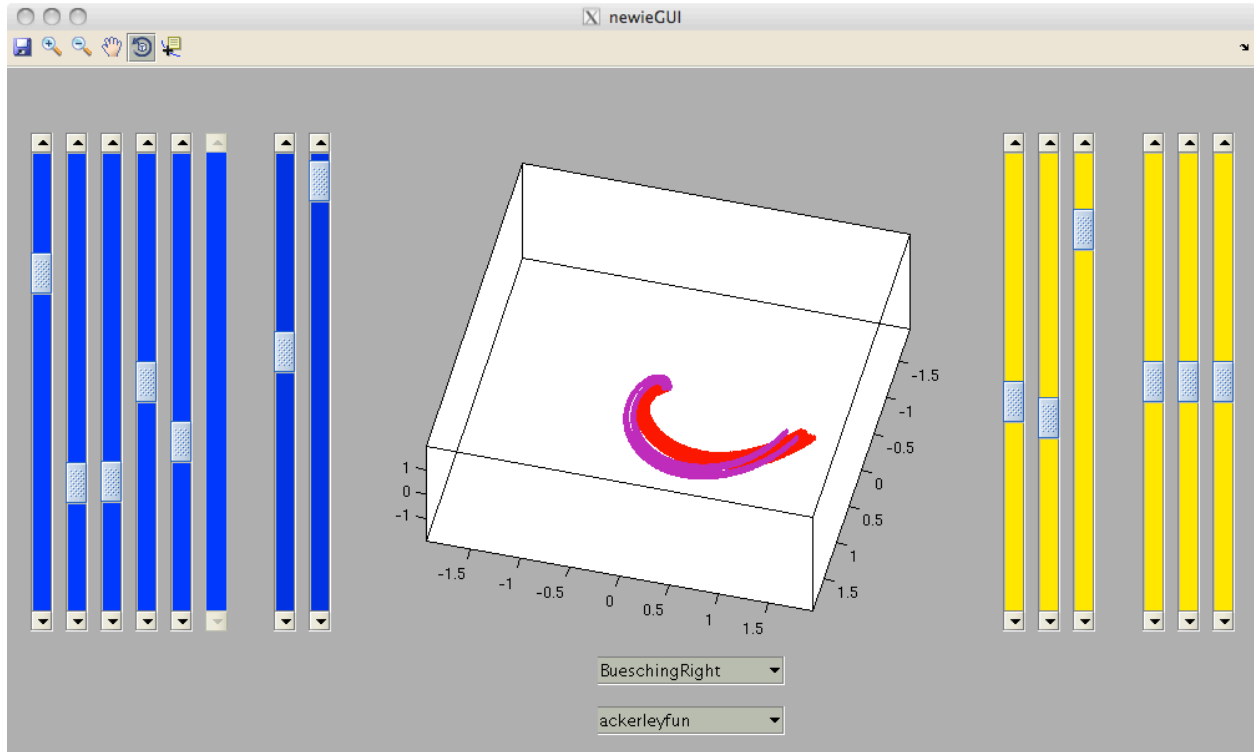


Figure 5 – A screenshot of the GUI being used to optimize an Ackerly-style model (purple) to the right tusk of the Buesching mastodon (red). This is not yet close to being a good fit. The first six light blue sliders control tusk parameters (unused slider grayed out), the next two dark blue sliders control length and start point, and the six yellow sliders control spatial orientation.

Computer Optimization With Nonlinear Regression In MATLAB

Once a set of tusk parameters that represents a decent visual approximation of a digitized tusk was found with the GUI, it was used as a starting point in a nonlinear regression optimizer that then found a set of tusk parameters that further minimized the objective function. Nonlinear regression was necessary because the function being optimized was not linearly dependent on the input variables. Many appropriate optimization functions exist, but for this work the *fminsearch* function from the *optimtool* toolbox in MATLAB was used. This function searches without bounds for the minimum of an objective function given a starting estimate of the parameter values. In the complicated objective functions optimized here that had up to fourteen different parameters, there were many local minima in which an optimizer could have been trapped, so a good starting estimate from the GUI was extremely important.

Picking appropriate optimization and objective functions was also critical to getting good results. Generally an objective function written to fit two things together, as was needed here, takes the sum of the squares of the distances between points on the two objects, and this sum is what is minimized by the optimization function. The objective function used in this work, *optimtuskfun* (code in Appendix C), goes over each point in the data model and finds the distance to the nearest point on the generated model. The sum of the squares of these inter-model distances was the value minimized and was named the goodness-of-fit (GOF), and could be observed during optimization by checking the box in the *optimtool* window labeled 'Function Value'. When the parameters were optimized to minimize the GOF, this objective function stretched the generated model to be near the digitized tusk. A deceptively similar approach sums the distances from each point in the generated model to its nearest point on the digitized model. However, optimizing this function yields a generated tusk that is miniscule and wrapped around a single point on the digitized tusk surface. Clearly, care must be taken to choose the appropriate objective function.

It is difficult to determine the absolute meaning of the goodness-of-fit (GOF) value because it can be affected by a variety of factors, though a lower value generally means a better fit. Comparing results for a single tusk, a lower GOF means that the generated model describes the digitized model better, but it loses its meaning when comparing between tusks because a variety of things can affect it. Between two identical tusks that differ only in that one has been scaled up, the large tusk will have a higher GOF because all of its points are farther apart. Two identical models differing only in number of points on their surfaces will also differ in their GOF, with the surface with more points having a higher GOF because there are more points contributing to the sum. The exact locations of the points on the surface will also affect the GOF in a non-predictable way because they could on average be closer or farther from points on the generated surface than those of another model.

Because of this variability, a threshold could not be set for the ‘best fit’ of generated models and a quantitative definition was hard to determine. The definition used required that there be a single set of parameters that corresponded to the lowest GOF and that the GOF was not much higher than that of the best fit for similar tusks. If there were multiple different parameter sets all yielding the same GOF value, this would mean that none were any better than others and no best fit could be claimed. A qualitative definition that also had to be achieved to declare a fit “good” was that the generated and digitized tusk models must resemble each other.

Optimization/GUI Technique

There are two general types of tusk-fitting studies contained in this work; both employ the GUI and optimizer as above, but with different goals. The first explores relationships between the different parameters in the models, and the second finds the best fits of generated models to digitized tusks. The best-fit studies were done first to get an initial estimate of the shape of each tusk, then parameter-relationship studies were done based on patterns observed in the best-fit studies to reduce the number of variables needed, thereby improving optimization speed and accuracy. Finally, best-fit studies were done again with the improved optimization to yield final results for pattern analysis. The results of all these studies are presented in Chapter V.

In the best-fit studies, the GUI and MATLAB optimizers were used alternatively with the improved results of one seeding the next. Once improvements could no longer be made with the GUI, either because its increments changed the values in larger steps than could improve the fit or because the fit looked good, the optimizer was used repeatedly with its results seeding its next round of optimization. If the results of this series settled to a particular value, it was evaluated with the GUI and if it looked good, it was declared a good fit. If it did not settle to a particular value and each optimization yielded a different parameter set, then the process was started over from the beginning to

see if a different starting point yielded a better fit. Once a fit was declared good, other starting points were tried as well in an attempt to ensure that it was indeed the best fit.

A single tusk that had an exceptionally good fit in the first best-fit study was selected for each parameter relationship study. The parameters being explored were fixed at various values, and the resulting optimized values for the other parameters and GOF were examined. Relationships were determined by graphical analysis of the resultant data.

V. Tusk Analysis

The goals of this work were to determine if three important tusk characters could be described by a logarithmic spiral model, and if so, if the parameters could be used to distinguish different species and sexes. The techniques of all the previous chapters were employed to generate tusk models based on shell growth models and fit them to digitized real tusks to obtain parameter values for the real tusks. This allowed a quantitative analysis of their forms.

Sources of Digitized Models

In the course of this work, the seven woolly mammoth tusks from Pevek were digitized photogrammetrically in PhotoModeler Scanner, and the HandyScan was used to scan the right tusks of the Buesching and Hyde Park mastodons. The work also analyzed models digitized previously by other workers. The left tusk of the Hyde Park mastodon and both tusks of the University of Wyoming mammoth had been scanned by the HandyScan laser scanner. The contact MicroScibe digitizer had been used to create models of the ZCHM, Taymyr, Wrangell Island, and St. Paul Island mammoths as well as the Bothwell mastodon and the left tusk of the Buesching mastodon. Refer back to Table 2 in Chapter II for a complete listing of the tusk digitization methods.

Parameter Relationships

In the logarithmic spiral model, the most important parameter relationship was that fixing D, E, and F to be equal. This improved optimization because it allowed spiral stretching by varying just A, B,

and C (for parameter descriptions, refer back to Chapter III and Table 3). It was also found that A and B were frequently equal, indicating that most of the spirals were well approximated by the traditional logarithmic spiral. Therefore, to improve the speed and accuracy of the optimization, A and B were set to covary in the optimizer. Runs were also done with them independent for comparison. For Pevek 2, 3, 5, and 6, even when A and B were allowed to vary independently, both parameters settled to the same value. The remaining tusks were different, though; the GOF of Pevek 1 was improved 1% when A and B varied independently. Pevek 7 was improved 4%, and Pevek 4 was improved 10%. For additional details, see Table 4, below.

Tusk	A	B	C	D=E=F	Θ	ϕ	Ψ	GOF	A/B
Pevek 1	0.281	-	6.392	0.05	-1.91	0.26	-2.3	1.41	-
Pevek 1	0.234	0.279	6.511	0.05	-1.91	0.23	-2.17	1.39	0.84
Pevek 2	0.355	-	5.175	0.05	-2.02	-0.57	-2.07	3.2	-
Pevek 2	0.355	0.353	5.17	0.05	-2.02	-0.59	-2.06	3.2	1.01
Pevek 3	0.476	-	4.313	0.05	-2.3	-0.67	-2	3.97	-
Pevek 3	0.471	0.475	4.287	0.05	-2.3	-0.67	-2	3.96	0.99
Pevek 4	0.325	-	4.882	0.05	-1.979	-0.73	-2.05	2.8	-
Pevek 4	0.332	0.379	4.699	0.05	-2.07	-0.71	-2.07	2.51	0.88
Pevek 5	0.268	-	4.902	0.05	-1.94	-0.24	-2.11	1.53	-
Pevek 5	0.264	0.268	4.899	0.05	-1.94	-0.25	-2.1	1.53	0.99
Pevek 6	0.597	-	5.973	0.05	2.41	0.59	-0.14	5.02	-
Pevek 6	0.572	0.597	6.31	0.05	2.4	0.63	-0.17	4.99	0.96
Pevek 7	0.17	-	-2.258	0.05	0.04	1.13	2.42	0.81	-
Pevek 7	0.165	0.177	-2.095	0.05	0.01	1.14	2.43	0.77	0.93

Table 4 – Optimized tusk parameters for the seven tusks from Pevek. The first of each pair, shaded in gray, were the parameters when A and B were set to co-vary while the second of each pair, in white, were the parameters when they were allowed to vary independently. GOF is the goodness-of-fit value.

Some of the parameters in the Ackerly model did not improve the optimizations. The aperture orientation parameters σ and Φ were not found to improve the fit and were subsequently set to zero, meaning the aperture translated in the direction it faced until it was rotated for the next translation. Because translation step sizes were based on aperture size, the starting aperture size, r_0 , was effectively a scale factor. This is important because the tusks had all been scaled to use the same units. A surprising linear relationship was found between α and δ ; they were almost always related by Equation 2.

$$\delta = -0.084\alpha + 0.13 \quad \text{Eq 2}$$

The notable exceptions to this relationship were the juvenile tusks, for which best fits were difficult to find, and especially ZCHM 20, known for its unusually straight shape. See Figure 6 for the graph in which this relationship is observed.

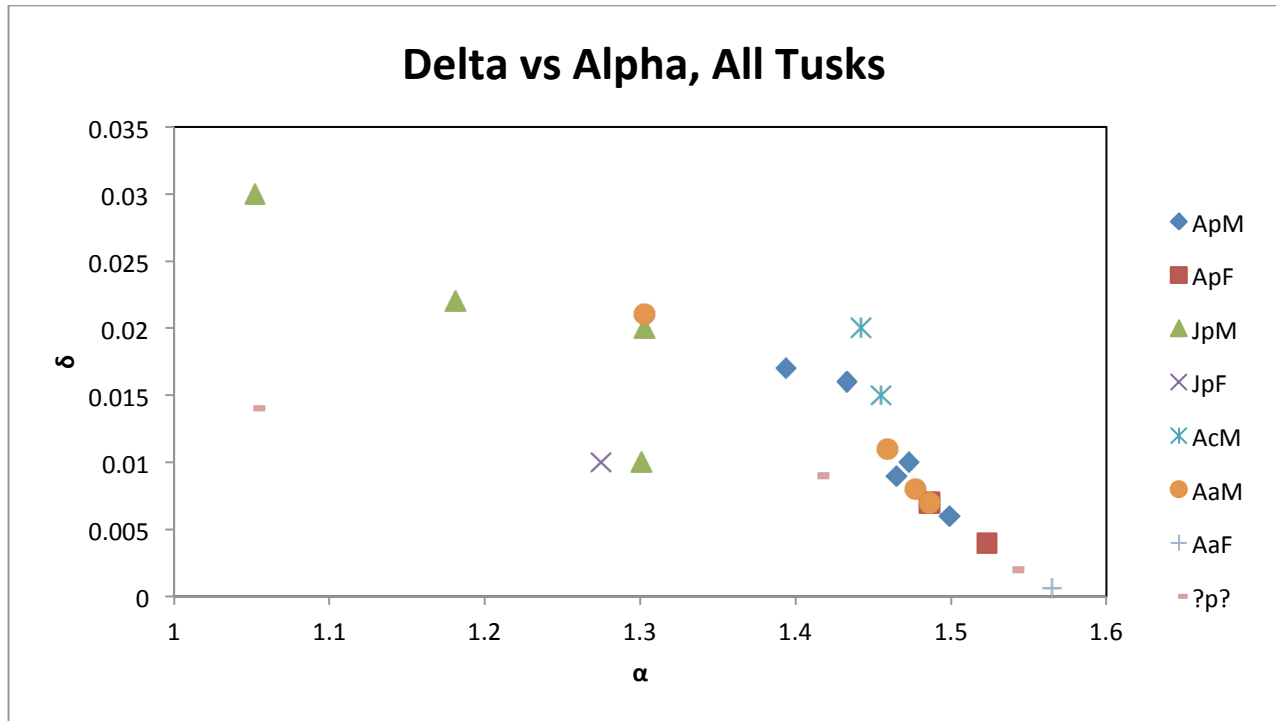


Figure 6 – Graphical relationship between α and δ . Only Adult *M. primigenius* and *M. americanum* appear to exhibit this trend. Each symbol represents a different descriptor. The first, capitalized character is the age (Adult or Juvenile), the second, lowercase letter is the first letter of the species (*primigenius*, *columbi*, or *americanum*), and the final, uppercase letter is the sex (Male or Female). The unknown individuals are ZCHM 19 and 20.

Describing Tusk Coiling Geometrically

Internal structural axes of the seven *M. primigenius* tusks from Wrangell Island were described well by the logarithmic spiral (Table 4, above) and all tusks were described well by the logarithmic-spiral-based Ackerly-style model (Table 5). Mammoth tusks had models that fit them much better than the mastodon tusks, with an average GOF of 0.26 for adult mammoths versus 0.58 for adult mastodons. The points defining the Ackerly-style model also exhibited a twisting similar to that exhibited by the grooves and ridges on real tusks (Figures 7&8), though this was not quantitatively evaluated. See Appendix B for a complete set of tusks and their Ackerly-style models.

Tusk	a	δ	γ	$\sigma=\Phi$	r_0	θ	ϕ	ψ	GOF
Pevek 1	1.523	0.004	0.873	0	0.031	-1.893	0.161	-2.225	0.1
Pevek 5	1.486	0.007	0.705	0	0.028	-1.969	-0.269	-2.136	0.12
Pevek 2	1.473	0.01	0.653	0	0.04	1.039	-2.41	1.225	0.22
<i>UWY-R</i>	<i>1.455</i>	<i>-0.015</i>	<i>0.641</i>	<i>0</i>	<i>0.054</i>	<i>-0.794</i>	<i>-1.9</i>	<i>0.214</i>	<i>0.37</i>
<i>UWY-L</i>	<i>1.442</i>	<i>0.02</i>	<i>0.631</i>	<i>0</i>	<i>0.062</i>	<i>1.356</i>	<i>-0.921</i>	<i>2.375</i>	<i>0.61</i>
Pevek 4	1.394	0.017	0.59	0	0.028	0.929	-2.202	1.265	0.19
Pevek 7	1.433	0.016	-0.569	0	0.018	2.877	2.02	-1.005	0.06
Pevek 6	1.465	0.009	0.487	0	0.05	2.472	0.664	-0.064	0.41
Pevek 3	1.499	0.006	0.427	0	0.041	0.788	-2.415	1.224	0.29
B-L	1.477	0.008	0.404	0	0.05	-0.724	-0.946	-1.538	0.41
HP-R	1.459	0.011	-0.385	0	0.057	2.8611	-0.5258	0.3194	0.95
B-R	1.303	0.021	-0.306	0	0.03	-0.711	-0.254	-2.245	0.36
HP-L	1.486	0.007	0.28	0	0.06	-0.55	1.138	-0.585	0.65
St Paul	1.54	0.002	0.254	0	0.085	1.966	-2.697	2.324	0.84
Bothwell	1.565	0.0006	0.047	0	0.069	0.919	1.532	0.901	0.51
2000-245	1.301	0.01	0.495	0	0.012	0.198	1.115	2.166	0.014
2000-246	1.275	0.01	0.52	0	0.01	0.913	0.568	-0.509	0.013
2000-286	1.303	0.02	-0.516	0	0.01	1.136	2.847	2.62	0.026
ZCHM19	1.415	0.009	0.334	0	0.04	1.206	-2.6	0.398	0.2
ZCHM20	1.052	0.014	0.017	0	0.031	-1.63	1.728	-0.072	0.26
ZCHM22	1.181	0.022	0.02	0	0.049	1.561	1.519	0	0.37
ZCHM23	1.052	0.03	0.049	0	0.038	-1.578	-3.127	-0.002	0.37

Table 5 – Parameters generating the best fits using the Ackerly model for each tusk, sorted by decreasing γ (juveniles, which had no best descriptive optimization, are below the dotted line as they did not contribute to the analysis). Females are in bold, mastodons are shaded gray, and the one pair of *M. columbi* tusks are in italics. In this way, it becomes obvious that the female mammoths have the highest γ , followed by the male mammoths, with the mastodons having the lowest γ (except for the St. Paul individual which exhibits island dwarfism). See Appendix B for these parameters used to make models and compared to actual tusks.



Figure 7 – Pevek 6 with red lines indicating obvious twisting grooves

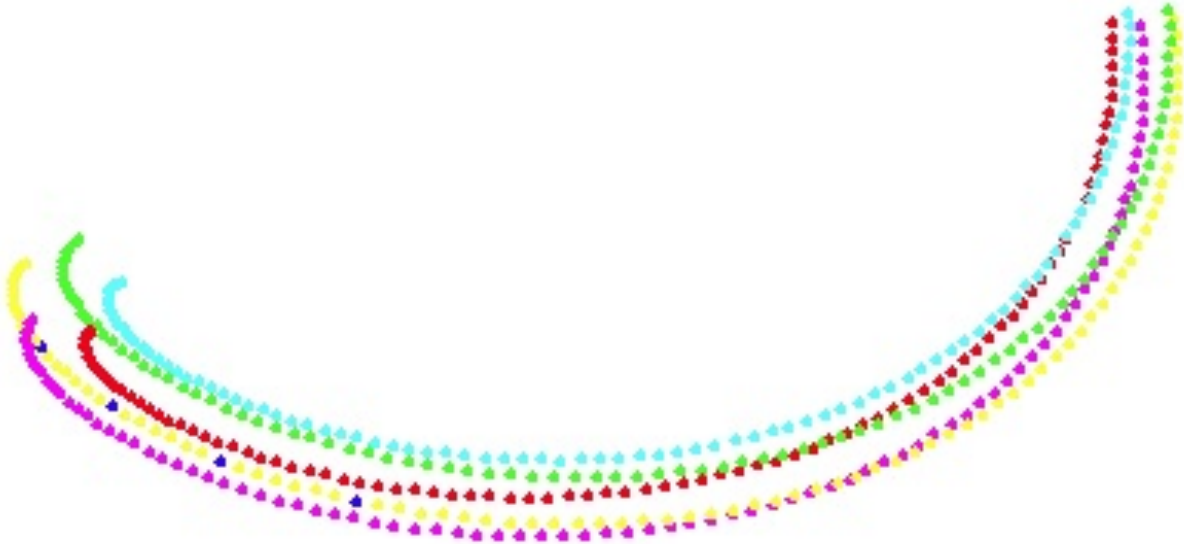


Figure 8 – An Ackerly-style model based on the parameters of Pevek 6 demonstrating a twist comparable to that of the real tusk. Compare the yellow trace here to the longer trace in Figure 7, the red trace here is similar to the shorter trace in Figure 7.

It was difficult to find best fits for the tusks of the juveniles (designated by 2000 or ZCHM) because their shape could be described equally well by many different parameter sets. Different forms were tested by fixing γ and optimizing the remaining parameters. For 2000-246, the juvenile tusk with the lowest GOF, γ was set at 0.1 increments from 0 to 1 and Pevek 7, the adult tusk with the lowest GOF,

had γ set at 0.1 increments from -0.4 to -0.7 (the negative sign indicating opposite tusk handedness). For the juvenile tusk, the GOF only varied from 0.013 to 0.20, with an average magnitude of change of 0.0009 for each increment of γ , while the adult tusk varied from 0.13 to 0.06 and back again, with an average magnitude of change of 0.05. Therefore, the parameters found for the juveniles are unlikely to be the “true” values and were left out of the analyses on sexual dimorphism and species-specific forms. This would happen with any tusk that lacks a well-defined coil, so it is likely that the Bothwell tusk could also be represented equally well by other parameter sets.

Species and Sex Differences Expressed in Parameter Set

Ignoring the parameters for juveniles, which were found to not necessarily be the best representation of their form, allows an analysis that yields several patterns. Gamma, the parameter that describes how much the coil is stretched out of the 2D plane, was found to be a strong indicator of whether an individual was a mammoth or mastodon, and whether a mammoth was a male or female. Adult *M. primigenius* females had this value greater than 0.70, tusks from the *M. primigenius* males and *M. columbi* male had values between 0.41 and 0.70 in, and tusks of adult *M. americanum* of both genders had values below 0.41, though the females of this species were only represented by the Bothwell tusk, which is likely to have other parameter sets that describe it well. The St. Paul tusk gets classified as coming from a mastodon under this scheme, and it falls into the range of male forms. Delta, the parameter that describes girth change over time and also affects coiling, was found to be less indicative. The general trend was for females to have lower δ , though this did not hold in all cases. The maximum δ for males was an order of magnitude higher than that of their female counterparts, but there were always some males with a value in the range of the females.

A few individuals had both the right and left tusks evaluated. For the Hyde Park mastodon and University of Wyoming mammoth, the right and left tusks had nearly identical parameters, while the Buesching mastodon had all its parameters differ greatly between its two tusks.

Discussion

The best indication of species/sex identity based on tusk coiling parameters is γ , though only adult tusks could be differentiated in this way. The γ difference between male and female *M. primigenius* is consistent with qualitative observations because female tusks appear straighter, which is a consequence of the extreme stretching into the 3rd dimension overwhelming the 2D coil. The low γ in *M. americanum* also is consistent with qualitative observations because they appear closer to a planispiral coil. The tusk from the St. Paul mammoth has an extremely low γ for a *M. primigenius*, but it is visibly different from other Woolly mammoths in other ways, such as its small size. The γ value does, however, fall below the minimum value to be a female, so this supports the idea that it represents a male individual. If juvenile tusks could have been optimized to a single result like the adults, their trends could also have been analyzed.

The higher maximum δ in males also matches well with observation because females have a tusk of fairly constant girth all along the tusk, while males increase more markedly. That this was not true for all tusks is interesting, but it could be caused by the nonlinear changes in girth along the length of a tusk. Male tusks especially change girth drastically which could lead to an inappropriate generalization of δ for the whole tusk. Since δ and α are related, the possible inaccuracy of δ means that α likely will also unfortunately not show a pattern. Therefore, if either parameter is to be used as an identifier, some method of determining them more accurately must be invented.

The GOF values for mammoths were much better than for mastodons, leaving the question of whether this was due to a true difference in quality of fit or just because the tusks were inherently different. All the tusks had similar numbers of points and the adult tusks were of similar sizes, so it seems likely that the reduced fit quality of mastodon tusks is true. The mammoth tusks were digitized long before the models of the mastodon tusks were obtained, meaning that they were fit to shell-growth models more often. This allowed more familiarity with them and also more chances to improve

their fits, so it is possible that with more optimizations the mastodon tusks could be fit with comparable quality.

Future Work

Analyzing more tusks is an obvious step for future work. Several of the species and sex groups were just represented by one individual, and none of the juvenile tusks contributed usable parameter sets to the analysis. More individuals would help validate the proposed boundaries between species and sex groups. Tusks from other species, such as modern elephants or narwhals, would also be interesting to examine to see if there are similar identifiable parameter values for them. Since one of the goals moving forward is to use these techniques to determine if Jeffersonian mammoths are indeed their own species or a hybrid between Woolly and Columbian mammoths, individuals of that species will need to be evaluated, as will more Columbian mammoths.

A useful tool for this work would be a fast and accurate way of getting 3D tusk data. The current best option, the HandyScan laser scanner, requires at least four hours of work. This means that a maximum of two tusks can be scanned in a normal work day, but this setup only works for tusks (or their casts) that can be brought to wherever the expensive laser scanner is used. The photogrammetric method is the only way to get models from tusks that cannot be transported to a laser scanner or contact digitizer, but it takes several hours to set up the photo shots and then an additional four hours per tusk to do the photogrammetric reconstruction. Possible breakthroughs in scanning technology include cheap, portable, homemade laser scanners or an adaptation of technology like the Microsoft Kinect.

Alternatively, a large improvement to analysis could be development of an analog system for determining the parameter values by making physical measurements on a tusk. This system could be similar to Ackerly's method of stereographic projection (Ackerly, 1989b), which tracks known points along their growth history to determine the orientation of the axis of coiling and value of σ . Such an

analytical measurement could be made on a small part of the tusk, and it should hold true for the rest of the tusk because logarithmic growth is isometric, with all parts being mathematically similar to all other parts. However, the anisometric girth change could invalidate this assumption and might require sampling of a larger part of the tusk.

A definitive way of determining the best fit of a shell-growth model to a real tusk would also be useful. Using the current methods requires a lot of time to determine if a fit is indeed the best because many different parameter sets must be tried to determine the best one. This method is dependent on the initial manual fit because the computer optimizer is incapable of changing it drastically. An indicator that could be used to help would be that tusks from the same individual should have similar r_0 and length values since they likely started and ended growing at the same time. However, this only works for the few individuals that have both the right and left tusks digitized.

Another way to improve the optimizations would be to use a better optimization function; MATLAB's *fminsearch* worked decently but could surely have been improved. A better optimization function would be to use Markov chain Monte Carlo (MCMC) methods. These functions hop around the parameter space, rather than moving in discrete steps, which improves the chances of finding the global minimum in a parameter space riddled with local minima. However, these functions are not included in the standard MATLAB package and would need to be explored using open-source modules or in a different programming language, such as Python, which has a simple add-on for MCMC optimization.

There are many other side projects that could also improve this overall work. One thing that might be interesting to evaluate is the effect of making the model allometric, with the parameters changing at various points along the tusk. The tusk girth, which should correspond to δ , is observed to change nonlinearly along the tusk length, which makes a strong case for the importance of exploring this.

It would also be important to examine other models; just because the logarithmic model describes tusks well does not mean that it describes tusks best. Other spirals such as an Archimedean or hyperbolic spiral exist and models based on these could be built and tested.

VI. Conclusion

The form of mammoth and mastodon tusks is described well by a logarithmic spiral that is stretched logarithmically into the third dimension, and expanding this model to describe a tusk surface also introduces a twisting similar to that observed in real tusks. The exact parameters of the logarithmic spiral vary between tusks, which is why they look different from each other. The parameter γ from the Ackerly model can be used to distinguish mastodon tusks from mammoth tusks and to determine the gender of a mammoth. By this method, the St. Paul tusk was determined likely to be from a male. Other parameters in the Ackerly-style model, such as δ and α , could possibly also be used to distinguish individuals if the values were made to more accurately reflect the tusk. There is much more work to be done, but this work makes progress in establishing the techniques needed in future projects.

Acknowledgements

This project would not have been possible without Dan Fisher for all his advice, support, and ideas throughout. Dan Miller provided lots of support with the photogrammetry and insight into its inner workings. Many people helped with ideas for how to do the optimization; Dan Fisher of course, Andreas Hayden, and various graduate students in the Department. The assistance from Dan Fisher, Tomasz Baumiller, Andreas Hayden, and Emily Thompson in reviewing this manuscript was also greatly appreciated.

Works Cited

- Ackerly, S. C. (1989a). Kinematics of Accretionary Shell Growth, with Examples from Brachiopods and Molluscs. *Paleobiology*, 15 (2), 147-164.
- Ackerly, S. C. (1989b). Shell Coiling in Gastropods: Analysis by Stereographic Projection. *Palaios*, 4 (4), 374-378.

Moseley, H. (1838). On the Geometrical Form of Turbinated and Discoid Shells. *Philosophical Transactions of the Royal Society of London* , 128, 351-370.

Okamoto, T. (1988). Analysis of Heteromorph Ammonoids by Differential Geometry. *Palaeontology* , 31 (1), 35-52.

Raup, D. M. (1967). Geometric Analysis of Shell Coiling: Coiling in Ammonoids. *Journal of Paleontology* , 41.

Raup, D. M. (1966). Geometric Analysis of Shell Coiling: General Problems. *Journal of Paleontology* , 40 (5), 1178-1190.

Raup, D. M., & Michelson, A. (1965). Theoretical Morphology of the Coiled Shell. *Science* , 147 (3663), 1294-1295.

Thompson, D. W. (1917). *On Growth and Form*. Cambridge University Press.

APPENDIX A - Construction of a Digital Model

Camera Calibration in PhotoModeler Scanner

An important preparatory step that must be done in PhotoModeler Scanner is a camera calibration, to teach the program how to correct an image for the curved lens that took it. This is done by using the same camera that took the images in the project to take a series of pictures of a sheet of dots that can be printed from the program. The pictures of the sheet need to be taken with the camera in its normal position and rolled 90° to each side and need to have each of those three positions looking at the sheet from each of its four sides, for a total of twelve pictures. The dots must fill as much of the image as possible because their locations inform the calibration. The program is not always able to recognize all the dots in all twelve images, but six successful images is minimal for a successful calibration. To increase the success rate of the images, the sheet can be printed on very white paper and taken in bright light so there is high contrast. The paper can also be affixed to a flat surface with tape. The biggest thing to vary is to take pictures from different angles of incidence, but making sure they are low enough to produce distinct images (taking all the pictures from straight over the paper negates the uniqueness between the different camera rolls and paper rotations).

Selecting the Best Parameters for Point-Matching in PhotoModeler Scanner


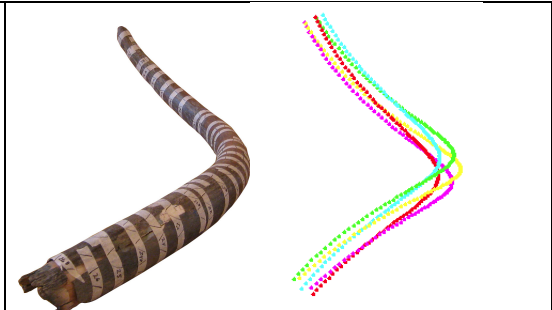
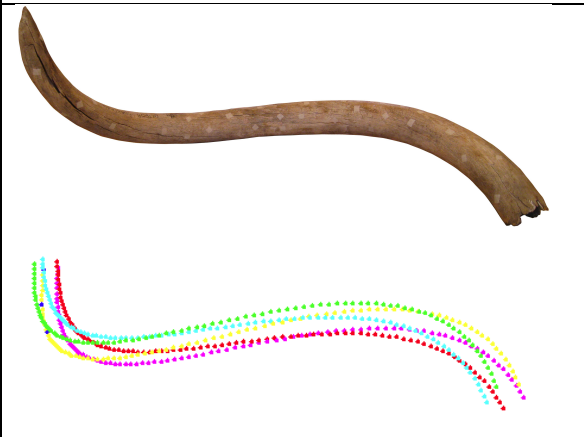
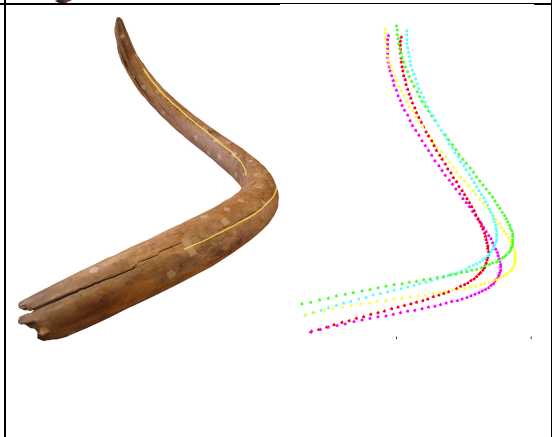
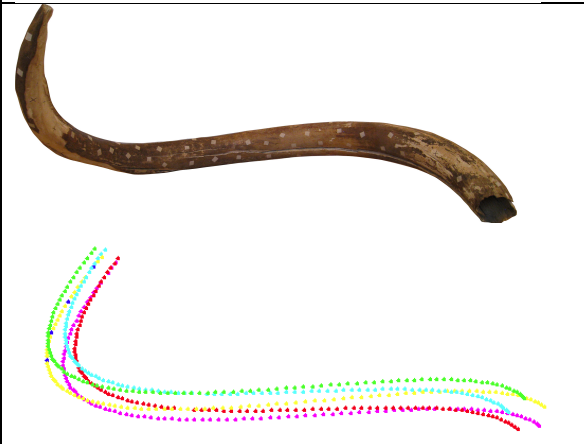
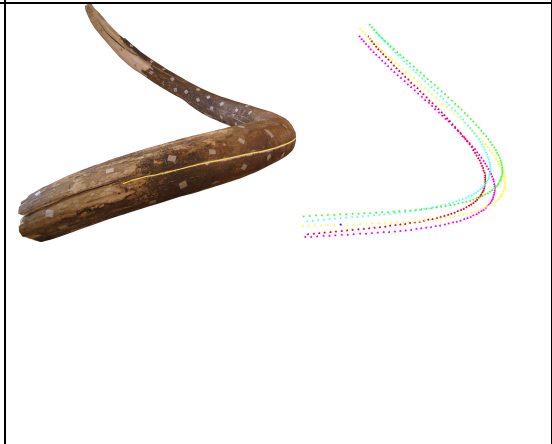
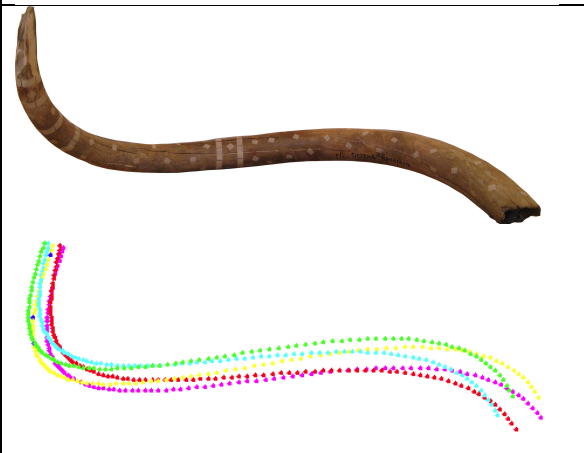
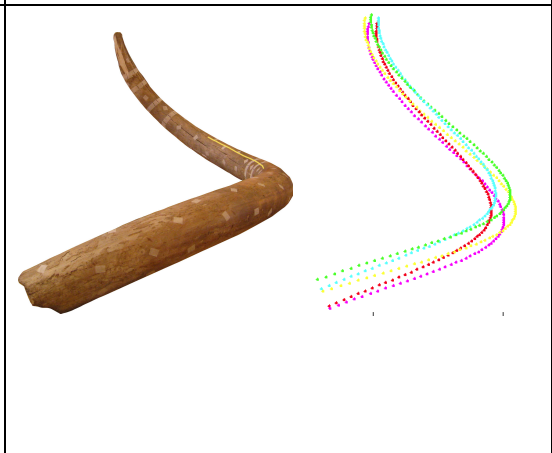
After some experimentation, the best combination of settings for one of the tusks was found to be a sampling rate of 15 mm, matching region radius of 15, and a texture of 1 with the other parameters left at the default. This averaged 250 points generated and 40 seconds of computer time required per image pair. Other parameter sets were found to be optimal for other tusks in the work, though this first set remained a good starting estimate for them. For tusks with minimal masking tape preparation, setting sampling rate to 30, matching region radius to 20, and texture to 1 yielded a decent model with exceedingly low noise. The tusks that had received lots of masking tape preparation proved difficult to process (probably because the smooth white tape did obscure some of the natural texture) so the matching region radius was set much lower, yielding tusk models that were excellent except for having huge amounts of noise that had to be manually trimmed out. Generally, a matching region radius of 5 or 10 was successful for these difficult tusks, but an additional 15-30 minutes was required to manually trim out the noisy points and leave behind a good model.

Name	Sampling Rate	Matching Region Radius	Texture	Raw # of Points	Quality Ranking	Probable Main Difficulty to Photogrammetry
<i>Pevek 1</i>	15 mm	5 mm	1	18,000	4	Received most tape bands
<i>Pevek 2</i>	15 mm	15 mm	1	50,000	1	Minimal tape squares
<i>Pevek 3</i>	15 mm	15 mm	1	86,000	3	Abundant tape squares
<i>Pevek 4</i>	15 mm	15 mm	1	46,000	2	Few tape squares, bands
<i>Pevek 5</i>	15 mm	5 mm	1	25,000	6	Difficult coloration
<i>Pevek 6</i>	30 mm	20 mm	1	80,000	5	Difficult coloration
<i>Pevek 7</i>	15 mm	5 mm	1	15,000	7	Difficult coloration

Table 6 - The photogrammetric parameters used to generate each tusk model, indicators of its quality, and likely quality detractors

APPENDIX B – Real and Ackerly-Style Model Tusks

TUSK	FRONT VIEW	SIDE VIEW
-------------	-------------------	------------------

<p>Pevek 1</p>		
<p>Pevek 2</p>		
<p>Pevek 3</p>		
<p>Pevek 4</p>		

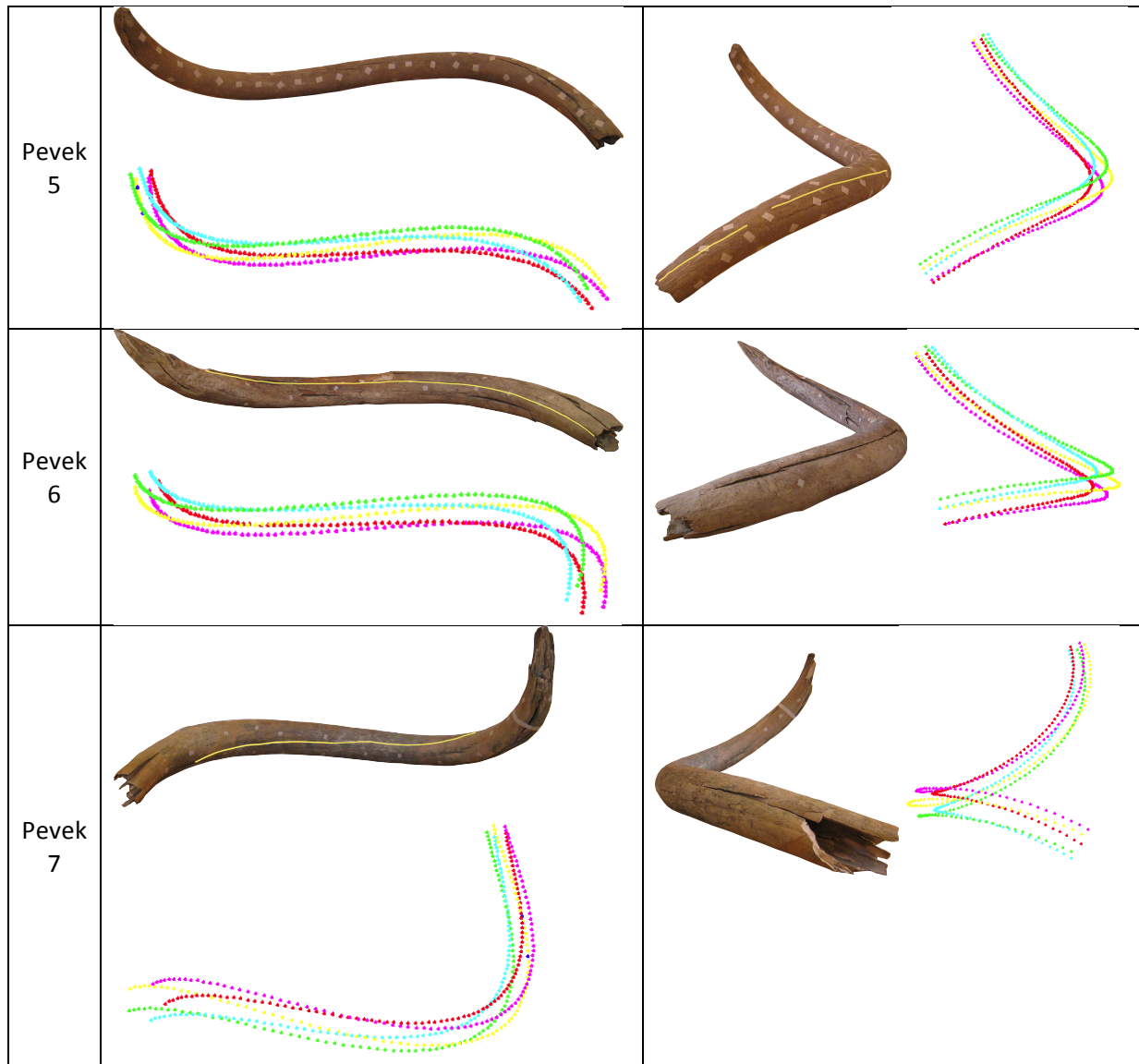
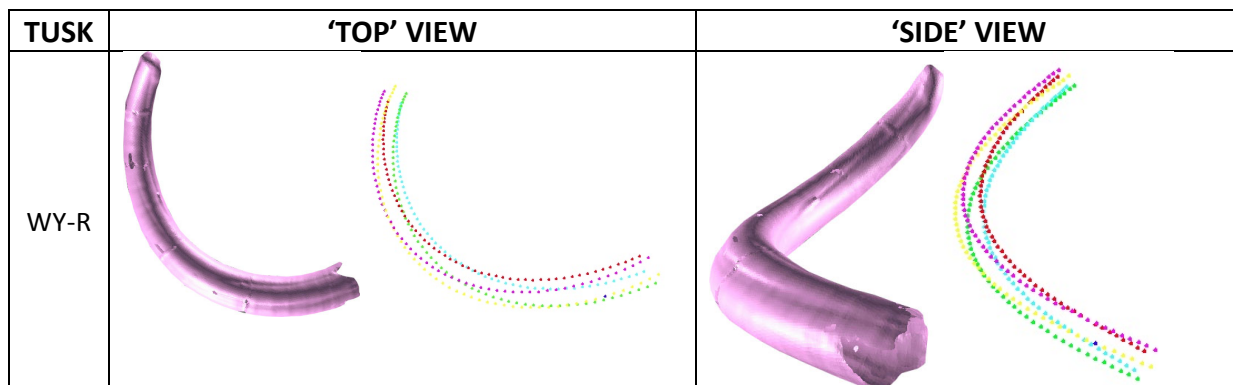


Figure 9 – Adult *M. primigenius* tusks (photos here, models in Figure 3 in Chapter II) and their accompanying Ackerly-style models (from parameters of Table 5 of Chapter V) in each of two views. Compare the twist visible in all the models to that of the actual tusks, highlighted in Pevek 6&7 Front View and Pevek 2, 3, &5 Side View. It was difficult to select matching views of the real tusk and model, which is why they are not identical. The models were specially generated with *ackerlyslowfun*, Appendix C. NOT scaled



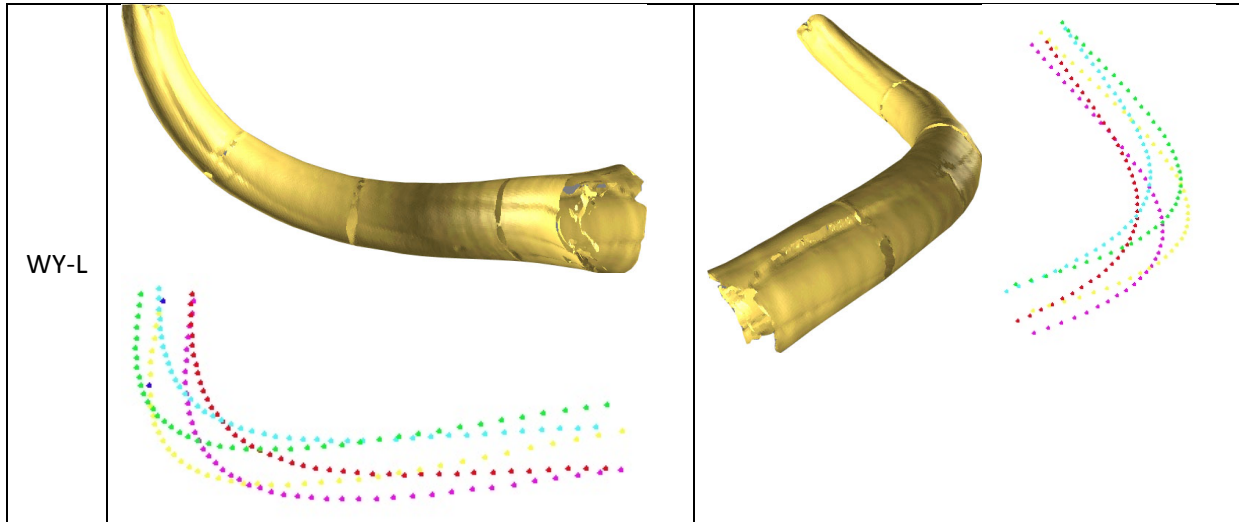
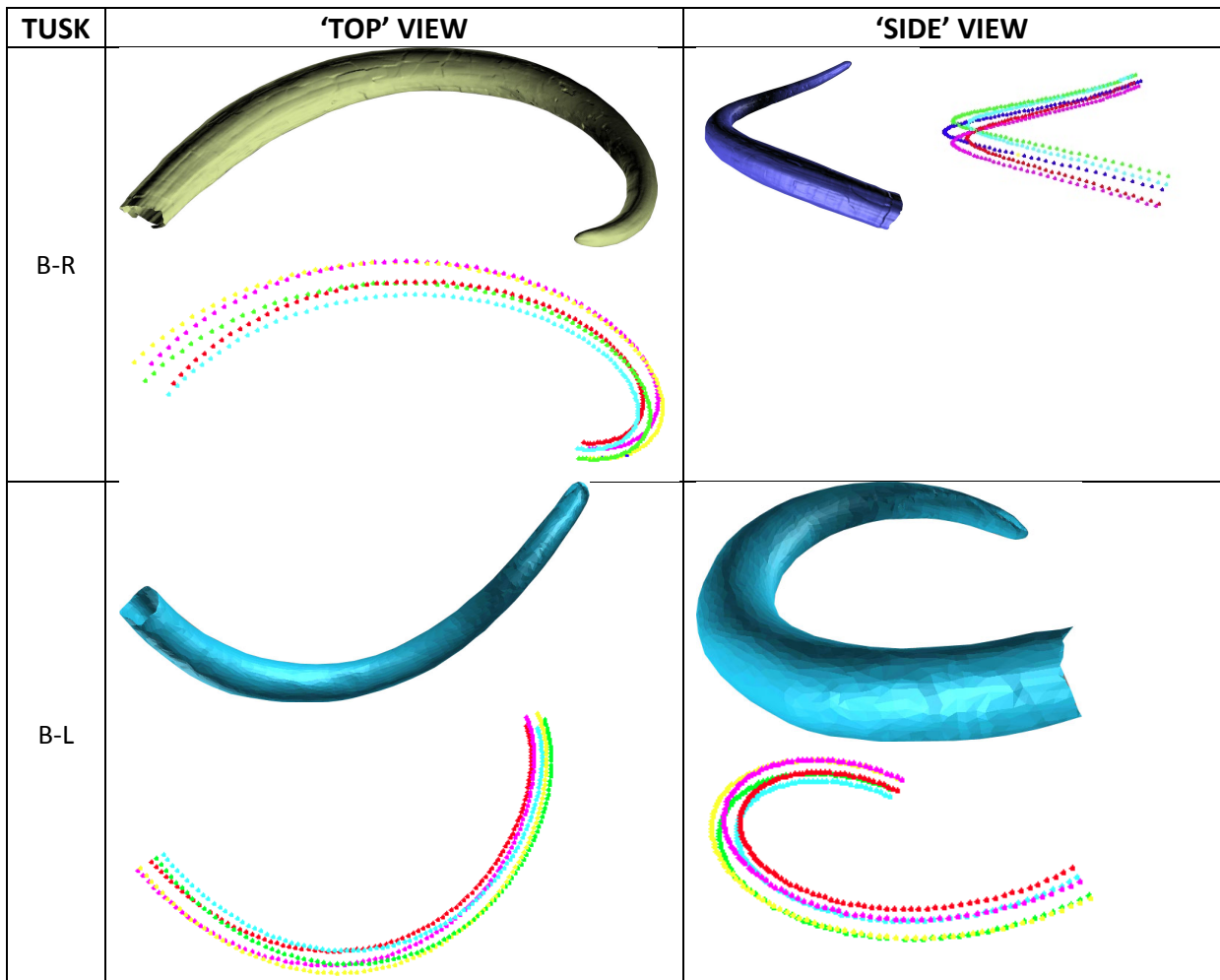


Figure 10 – The laser-scanned *M. columbi* tusks and their accompanying Ackerly-style models (from parameters of Table 5 of Chapter V) in each of two views. The actual tusks are broken at the tips and mounted with metal bands, which are visible as missing bands in the digitized models. It was difficult to select matching views of the real tusk and model, which is why they are not identical. The models were specially generated with *ackerlyslowfun*, Appendix C. NOT scaled



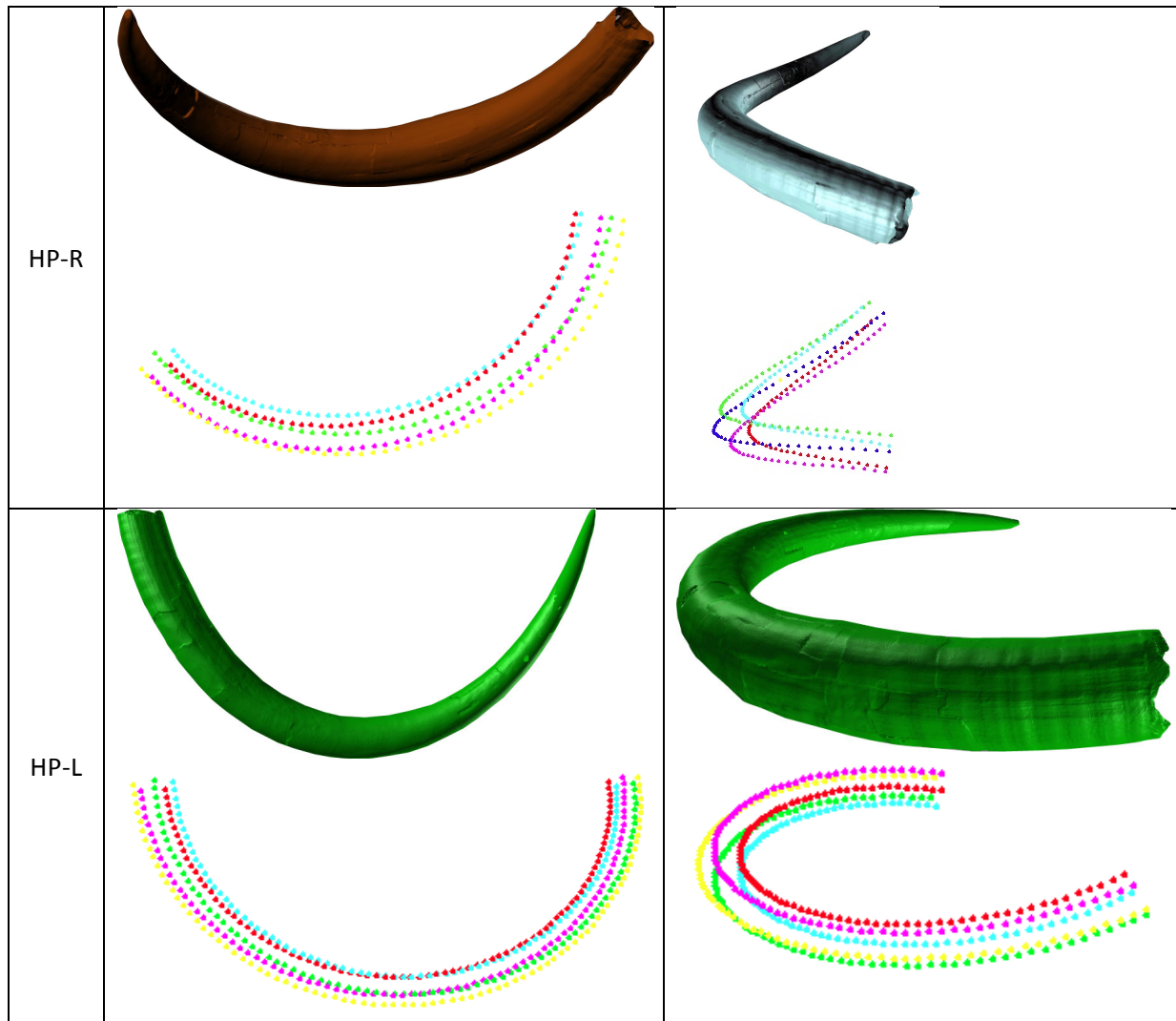
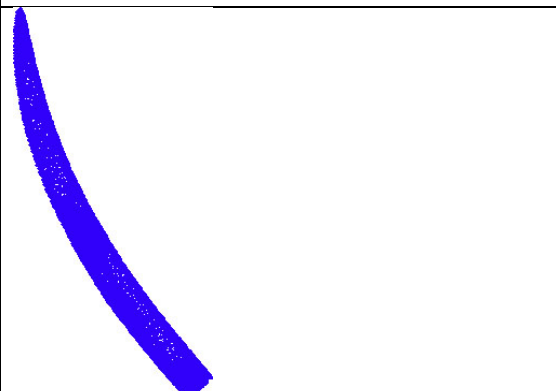
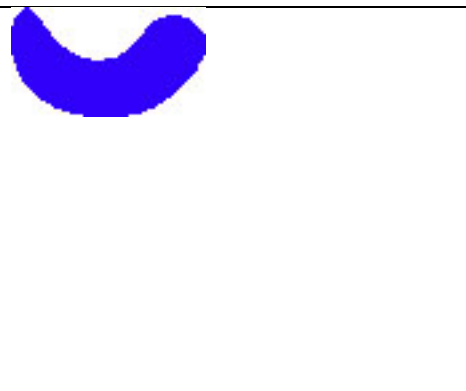


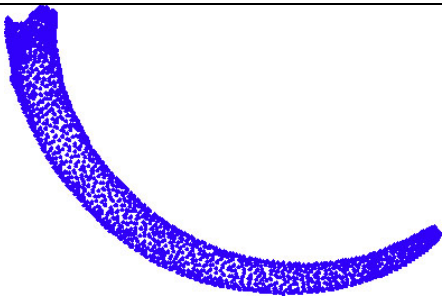

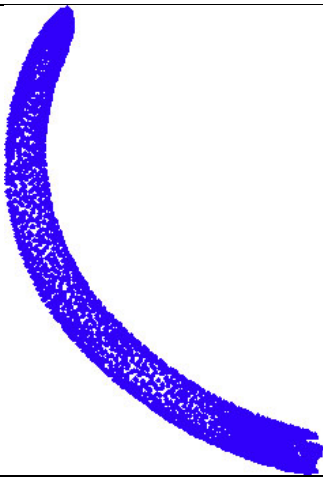



Figure 11 – The laser-scanned (HP-R, HP-L, B-R) and contact-digitized (B-L, Bothwell) *M. americanum* tusks and their accompanying Ackerly-style models (from parameters of Table 5 of Chapter V) in each of two views. It was difficult to select matching views of the real tusk and model, which is why they are not identical. The models were specially generated with *ackerlyslowfun*, Appendix C. NOT scaled.

TUSK	FRONT VIEW	SIDE VIEW
Bothwell		

2000-246		
2000-286		
ZCHM 19		

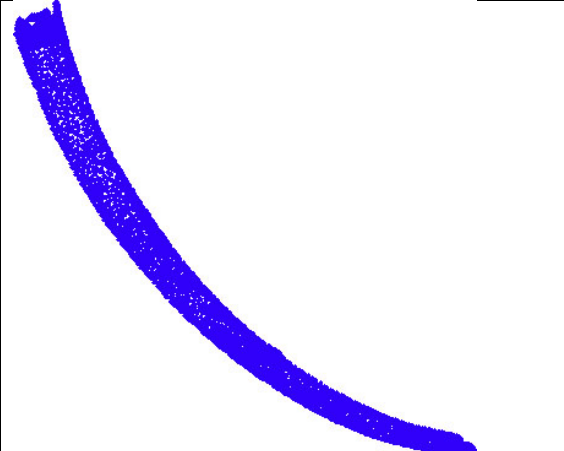
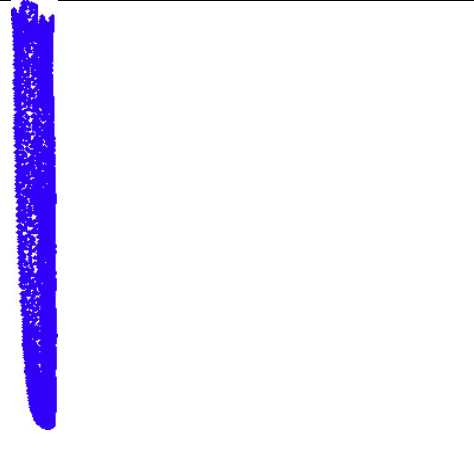
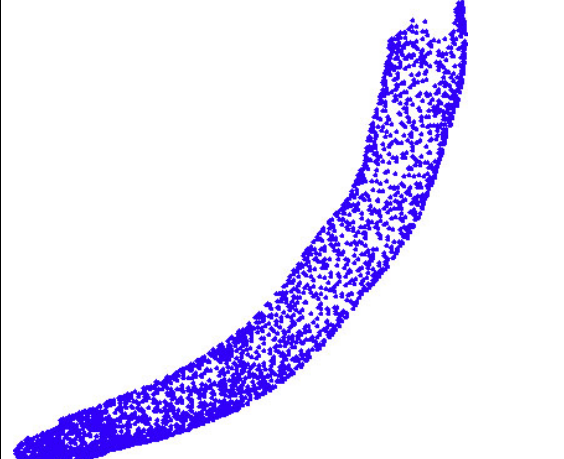
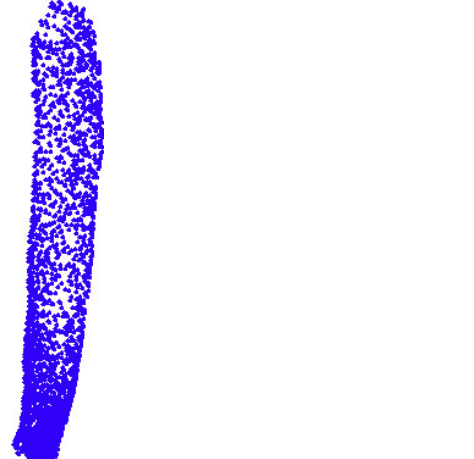
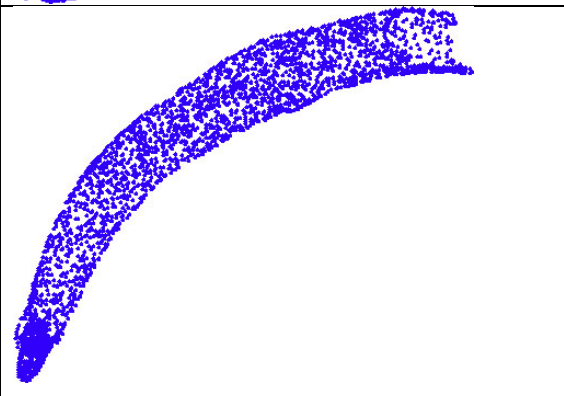
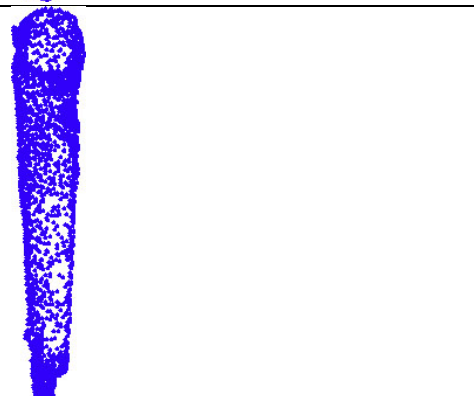
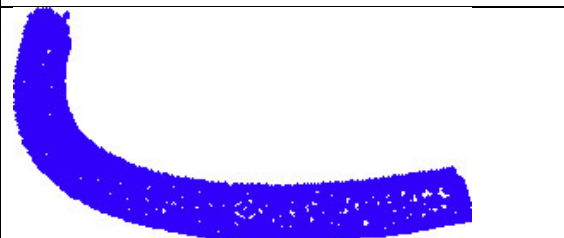
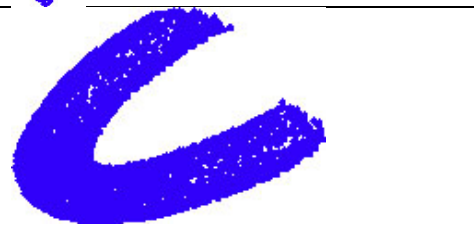
ZCHM 20		
ZCHM 22		
ZCHM 23		
St. Paul		

Figure 12 – The remaining contact-digitized models that did not have good fits from shell-growth models. These were generally too straight to be characterized by a single best model and were mostly excluded from analysis and therefore do not have fit models included here. However, Bothwell and St. Paul were both included because there were questions about them; Bothwell is the sole female mastodon and St. Paul was of questionable gender. NOT to scale.

APPENDIX C – MATLAB Code Used

All the programs used in this work are listed in their entirety here. They are not the most streamlined versions possible as they were built from the ground up and tweaked as necessary. There are likely many spandrels. Other programs not mentioned in the text include *ackerlyslowfun* (to generate the tusks with different color traces rather than one discrete set), *rotor* (rotated points around the origin based on input yaw, pitch, and roll), and *circlepoints* (based on freeware on MATLAB help forums, generated a circle of points representing an aperture).

logtusk

This MATLAB program creates a model of the central axis of a spiraled tusk given parameter values. It is based on the traditional equations for a logarithmic spiral.

```
function [x,y,z] = logtusk(A,B,C,D,E,F,leng,tuskleft,THETA,PHI,PSI,X,Y,Z)
%A,B,C,D,E,F,length are properties of the function, THETA/PHI/R/X/Y/Z are
%transformational properties

E = D; %Based on observation, D=E=F found to give best optimization
F = D;
R = 1; %this is an artifact from a previous version

NOP = 1000; %Number Of Points in curve
t=(1:NOP)/NOP*leng;
for i=1:NOP
    x(1,i) = A*(exp(D*t(1,i))*cos(t(1,i))-exp(D*t(1,1))*cos(t(1,1)));% the subtraction to zero out the model
    so it rotates correctly
    y(1,i) = B*(exp(E*t(1,i))*sin(t(1,i))-exp(E*t(1,1))*sin(t(1,1)));
    z(1,i) = C*(exp(F*t(1,i))-exp(F*t(1,1)));
    shiftx(1,i) = X;
    shifty(1,i) = Y;
    shiftz(1,i) = Z;
end
%rotation matrices
Rotx = [1,0,0;0,cos(THETA),-sin(THETA);0,sin(THETA),cos(THETA)];
Roty = [cos(PHI),0,sin(PHI);0,1,0;-sin(PHI),0,cos(PHI)];
Rotz = [cos(PSI),-sin(PSI),0;sin(PSI),cos(PSI),0;0,0,1];

pts = rot90(Rotz*Roty*Rotx*[x;y;z]+[shiftx;shifty;shiftz]);
[a,b,c] = cart2sph(pts(:,1),pts(:,2),pts(:,3));
%this allows it to be scaled by R, now an obsolete variablea
[x,y,z] = sph2cart(a(1:floor(NOP*tuskleft)),b(1:floor(NOP*tuskleft)),c(1:floor(NOP*tuskleft))*R);
```

```

pts = rot90(Rotz*Roty*Rotx*[x;y;z]+[shiftx;shifty;shiftz]);
[a,b,c] = cart2sph(pts(:,1),pts(:,2),pts(:,3));
[x,y,z] = sph2cart(a(1:floor(NOP*tuskleft)),b(1:floor(NOP*tuskleft))*R);
%-----End of Program

```

akerlyfun

This MATLAB program generates a tusk based on the Ackerly-style model (Ackerly, 1989).

```

function [x,y,z] = akerlyfun(alpha,delta,gamma,sigma,r_0,~,leng,startspot,THETA,PHI,PSI,X,Y,Z)

```

```

%This program is after Ackerly, Kinematics of accretionary shell growth,
%with examples from Brachiopods and Molluscs, Paleobiology 15.2.147-164,
%1989. All figure and equation references refer to that paper.
%alpha defines spiral angle, fig 3B
%delta - angle of growth cone, fig 3B
%gamma - spiraling downward angle
%sigma - angle between translation vector and previous aperture pole, fig 3A
%-----

```

```

%program parameters
NOP = 4; %Number Of Points in each generating curve
epsilon = 0.5; %growth step parameter
steps = 100*leng;
centroid = [0,0,0];

```

```

%initialization
r = r_0;
psi = 0;
surfleng = 0;

```

```

%comment out for opposite tusk handedness (negative is right tusk)
gamma = -gamma;

```

```

%get index of place where data tusk to start on grown tusk
startspot = floor(startspot);
if startspot<2
    startspot = 2;
elseif startspot>steps
    startspot = floor(steps);
end

```

```

for i=2:steps
    %should follow order of translation then rotation, but this starts on
    %second step, so the rotation that occurs first is actually for the
    %previous growth step

```

```

%update orientations
psi(i) = psi(i-1)+epsilon*tan(delta)*tan(alpha); %Appendix eq. B6
%expand radius
r(i) = r(i-1)*(1+epsilon*tan(delta)); %Appendix eqs. B4 & B5
%move centroid
centroid(i,:) = centroid(i-1,:)+epsilon*r(i)*[cos(gamma)*cos(psi(i)-sigma+pi/2),cos(gamma)*sin(psi(i)-
sigma+pi/2),sin(gamma)]; %Appendix eq. B3
%plot generating curve

[tusksurface(surfleng+1:surfleng+NOP,1),tusksurface(surfleng+1:surfleng+NOP,2),tusksurface(surfleng+1
:surfleng+NOP,3)] = circlepoints(centroid(i,:),r(i),NOP,gamma-pi/2,0,psi(i)-sigma);
surfleng = length(tusksurface(:,1));
end
zeroedtusk(:,1) = tusksurface(:,1) - centroid(startspot,1);
zeroedtusk(:,2) = tusksurface(:,2) - centroid(startspot,2);
zeroedtusk(:,3) = tusksurface(:,3) - centroid(startspot,3);
tusk = rotor(THETA,PHI,PSI)*transpose(zeroedtusk); %rotates
x = tusk(1,:)+X;
y = tusk(2,:)+Y;
z = tusk(3,:)+Z;
%-----End of Program

```

newieGUI

A MATLAB Graphical User Interface (GUI) started with the help of MATLAB's GUIDE, but mostly programmed independently. Lines 113 and 114 could be changed to allow use of different data models and generated models, respectively.

```

function varargout = newieGUI(varargin)
% NEWIEGUI MATLAB code for newieGUI.fig
% NEWIEGUI, by itself, creates a new NEWIEGUI or raises the existing
% singleton*.
%
% H = NEWIEGUI returns the handle to a new NEWIEGUI or the handle to
% the existing singleton*.
%
% NEWIEGUI('CALLBACK',hObject,eventData,handles,...) calls the local
% function named CALLBACK in NEWIEGUI.M with the given input arguments.
%
% NEWIEGUI('Property','Value',...) creates a new NEWIEGUI or raises the
% existing singleton*. Starting from the left, property value pairs are
% applied to the GUI before newieGUI_OpeningFcn gets called. An
% unrecognized property name or invalid value makes property application
% stop. All inputs are passed to newieGUI_OpeningFcn via varargin.
%
% *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one

```

```

% instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help newieGUI

% Last Modified by GUIDE v2.5 14-Mar-2011 22:08:31

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',    mfilename, ...
                  'gui_Singleton', gui_Singleton, ...
                  'gui_OpeningFcn', @newieGUI_OpeningFcn, ...
                  'gui_OutputFcn', @newieGUI_OutputFcn, ...
                  'gui_LayoutFcn', [], ...
                  'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before newieGUI is made visible.
function newieGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% varargin command line arguments to newieGUI (see VARARGIN)

% Choose default command line output for newieGUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes newieGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = newieGUI_OutputFcn(hObject, eventdata, handles)

```

```

% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

%-----The part where I start writing...

%Initialize variables so they can be global
datums.A = 1.5;
datums.B = 0;
datums.C = 0;
datums.D = 0;
datums.E = 0.01;
datums.F = 0;
datums.length = 10;
datums.startspot = 680;
datums.THETA = 0;
datums.PHI = 0;
datums.PSI = 0;
datums.R = 1;
datums.X = 0;
datums.Y = 0;
datums.Z = 0;

%pevek1 guess [0.6156,0.552,9.8214,0.0276,-2.09,0.45,-3.02]

%Define sliders, blue on the left and maize on the right of the graph
Aslider =
uicontrol(hObject,'BackgroundColor','b','Style','slider','Max',1.6,'Min',1,'Value',datums.A,'Position',[20
100 18 400],'Callback',{@slider_Callback,handles,'A'});
Bslider =
uicontrol(hObject,'Enable','on','BackgroundColor','b','Style','slider','Max',0.05,'Min',0,'Value',datums.B,'P
osition',[48 100 18 400],'Callback',{@slider_Callback,handles,'B'});
Cslider =
uicontrol(hObject,'BackgroundColor','b','Style','slider','Max',1,'Min',0,'Value',datums.C,'Position',[76 100
18 400],'Callback',{@slider_Callback,handles,'C'});
Dslider = uicontrol(hObject,'Enable','off','BackgroundColor','b','Style','slider','Max',0.6,'Min',-
0.6,'Value',datums.D,'Position',[104 100 18 400],'Callback',{@slider_Callback,handles,'D'});
Eslider =
uicontrol(hObject,'Enable','on','BackgroundColor','b','Style','slider','Max',0.1,'Min',0,'Value',datums.E,'Po
sition',[132 100 18 400],'Callback',{@slider_Callback,handles,'E'});
Fslider =
uicontrol(hObject,'Enable','off','BackgroundColor','b','Style','slider','Max',1,'Min',0,'Value',datums.F,'Posit
ion',[160 100 18 400],'Callback',{@slider_Callback,handles,'F'});
%-----

```

```

lengthslider = uicontrol(hObject,'BackgroundColor',[0 0
0.9],'Style','slider','Max',100,'Min',0,'Value',datums.length,'Position',[215 100 18
400],'Callback',{@slider_Callback,handles,'length'});
startspotslider = uicontrol(hObject,'BackgroundColor',[0 0
0.9],'Style','slider','Max',1000,'Min',2,'Value',datums.startspot,'Position',[243 100 18
400],'Callback',{@slider_Callback,handles,'startspot'});
THETAslider = uicontrol(hObject,'BackgroundColor',[1 0.92 0],'Style','slider','Max',pi,'Min',-
pi,'Value',datums.THETA,'Position',[800 100 18 400],'Callback',{@slider_Callback,handles,'THETA'});
PHIs slider = uicontrol(hObject,'BackgroundColor',[1 0.92 0],'Style','slider','Max',pi,'Min',-
pi,'Value',datums.PHI,'Position',[828 100 18 400],'Callback',{@slider_Callback,handles,'PHI'});
PSIs slider = uicontrol(hObject,'BackgroundColor',[1 0.92 0],'Style','slider','Max',pi,'Min',-
pi,'Value',datums.PSI,'Position',[856 100 18 400],'Callback',{@slider_Callback,handles,'PSI'});
% Rslider = uicontrol(hObject,'Enable','on','BackgroundColor',[1 0.75
0],'Style','slider','Max',10,'Min',0,'Value',datums.R,'Position',[884 100 18
400],'Callback',{@slider_Callback,handles,'R'});
Xslider = uicontrol(hObject,'BackgroundColor',[1 0.92 0],'Style','slider','Max',0.5,'Min',-
0.5,'Value',datums.X,'Position',[912 100 18 400],'Callback',{@slider_Callback,handles,'X'});
Yslider = uicontrol(hObject,'BackgroundColor',[1 0.92 0],'Style','slider','Max',0.5,'Min',-
0.5,'Value',datums.Y,'Position',[940 100 18 400],'Callback',{@slider_Callback,handles,'Y'});
Zslider = uicontrol(hObject,'BackgroundColor',[1 0.92 0],'Style','slider','Max',0.5,'Min',-
0.5,'Value',datums.Z,'Position',[968 100 18 400],'Callback',{@slider_Callback,handles,'Z'});
datachooser =
uicontrol(hObject,'Style','popupmenu','string',{'Pevek1','Pevek2','Pevek3','Pevek4','Pevek5','Pevek6','Peve
k7','BueschingLeft','BueschingRight','wyleft','wyright','tusk1','tusk2','tusk3','Bothwell','HydeParkLeft','Hy
deParkRight','M43','StPaul','Taimir3','Taimir245','Taimir246','Taimir286','ZCHM19','ZCHM20','ZCHM22','
ZCHM23','none'},'position',[475 50 150 30],'Callback',{@datachooser_Callback,handles});
modelchooser =
uicontrol(hObject,'Style','popupmenu','string',{'ackerleyfun','none','ackerlycentroids','logtusk','neglogtusk
','oka_tusk'},'position',[475 10 150 30],'Callback',{@modelchooser_Callback,handles});

%lower bounds [0,0,0,0,-pi,-pi/2,0,-1.5,-1.5,-1.5]
%upper bounds [0.5,3,3,7,pi,pi/2,1,1.5,1.5,1.5]
%start point [0.3,1,0.6,3.2,0.6,-1.4,.15,-0.3,-0.6,-1]

%Pevek1 best guess bounds
%lower [0.15,0.85,0.5,3,0.3,-1.6,0.1]
%upper [0.35,1.15,0.7,4,0.7,-1.3,0.2]
%start [0.43,0.33,0.34,2.8,0.68,-1.16,0.63]

%Initialize the graph
%Define graph axes
datums.checkplot =
axes('Parent',hObject,'ActivePositionProperty','outerposition','OuterPosition',[.27 .12 .5 .8],'Box','on');
hold on %so that camera position is preserved
%Set tusk source data
Pevek1maker;
datatusk = Pevek1;
%save and process the data

```

```

datums.datatusk = datatusk(1:5:length(datatusk(:,1)),:);
tuskplot = scatter3(datatusk(:,1),datatusk(:,2),datatusk(:,3),'.');
%fix the axes to a cube based on extrema of the tusk
axisend = 1.3*max([abs(min(datatusk(:,1))) abs(max(datatusk(:,1))) abs(min(datatusk(:,2)))
abs(max(datatusk(:,2))) abs(min(datatusk(:,3))) abs(max(datatusk(:,3)))]);
axis([-axisend axisend -axisend axisend -axisend axisend]);
datums.modeltoget = 'ackerleyfun'; %Set model type
setappdata(handles.figure1,'datums',datums); %Make all the data accessible
datums %print initial values
plotit(hObject,handles); %plot the model

```

%Callbacks-----

```

function slider_Callback(hObject, eventdata, handles, valtoget)
%Do something when the sliders are moved
%get/set data based on the slider
datas = getappdata(handles.figure1,'datums');
eval(horzcat('datas.',valtoget,' = get(hObject,"Value"'));
setappdata(handles.figure1,'datums',datas); %Make all the data accessible
delete(datas.modelplot);
plotit(hObject,handles); %plot the stuff

```

```

function datachooser_Callback(hObject, eventdata, handles)
%Choose source data for comparison
clearvars -except hObject eventdata handles datums datas
datas = getappdata(handles.figure1,'datums');
%read in the appropriate data
stringthing = get(hObject,'String');
datatoget = char(stringthing(get(hObject,'Value')));
if length(datatoget)==5&datatoget(1:4)=='tusk'
    eval(horzcat('tusk',datatoget(5),'axispoints;'));
    eval(horzcat('datatusk = tusk',datatoget(5),';'));
    eval(horzcat('clear tusk',datatoget(5)));
    axisend = 2;
elseif length(datatoget)==4&datatoget(1:4)=='none'
    datatusk = [0,0,0];
    axisend = 2;
else
    eval(horzcat(datatoget,'maker;'));
    eval(horzcat('datatusk = ',datatoget,'(1:7:length(',datatoget,'(:,1),:);'));
    eval(horzcat('clear ',datatoget));
    axisend = 1.3*max([abs(min(datatusk(:,1))) abs(max(datatusk(:,1))) abs(min(datatusk(:,2)))
abs(max(datatusk(:,2))) abs(min(datatusk(:,3))) abs(max(datatusk(:,3)))]);
end
%reset everything, including the figure
cla(datas.checkplot)
clear datas.datatusk

```

```

% datas.checkplot =
axes('Parent',hObject,'ActivePositionProperty','outerposition','OuterPosition',[.27 .12 .5 .8],'Box','on');
% hold on %so that camera position is preserved
%save and process the data
datas.datatusk = datatusk;
scatter3(datatusk(:,1),datatusk(:,2),datatusk(:,3),'.');
%fix the axes to a cube based on extrema of the tusk
axis([-axisend axisend -axisend axisend -axisend axisend]);
setappdata(handles.figure1,'datums',datas); %Make all the data accessible
plotit(hObject,handles);

function modelchooser_Callback(hObject,eventdata,handles)
%Select model for comparison to data
datas = getappdata(handles.figure1,'datums');
clearvars stringthing
stringthing = get(hObject,'String');
datas.modeltoget = char(stringthing(get(hObject,'Value')));
setappdata(handles.figure1,'datums',datas); %Make all the data accessible
delete(datas.modelplot);
plotit(hObject,handles);

function plotit(hObject,handles)
%Plotting function
datas = getappdata(handles.figure1,'datums');
%plot stuff
eval(horzcat('[tuskx,tusky,tuskz] =
',char(datas.modeltoget),'(datas.A,datas.B,datas.C,datas.D,datas.E,datas.F,datas.length,datas.startspot,
datas.THETA,datas.PHI,datas.PSI,datas.X,datas.Y,datas.Z);'));
datas.modelplot = scatter3(tuskx,tusky,tuskz, '.');
setappdata(handles.figure1,'datums',datas); %Make all the data accessible
%-----End of Program

```

optimtuskfun

This is the objective function used during computer optimization. Starting parameters were entered in to *optimtool* as [a(1) a(2) a(3) ... a(i)]. They could also be fixed by changing the appropriate line of code, which allowed the parameter relationship studies. Changing the name of a tusk in its three occurrences in lines 25&26 is how different tusks were evaluated.

```

function totaldist = optimtuskfun(a)
%Evaluates the distance from each point on the surface to the model, which
%by minimization stretches the model to fit down the center of the surface.
%**Make sure that the two places marked CHANGE with **** are set, otherwise
%the wrong data will be optimized to the wrong model
A = a(1);

```



```

B = a(2);
C = a(3);
D = a(4);
E = a(5);
F = a(6);
leng = a(7);
startspot = a(8);
THETA = a(9);
PHI = a(10);
PSI = a(11);
R = 1; %obsolete, but still needs to be left in
X = a(12);
Y = a(13);
Z = a(14);
totaldist = 0; %initialize the summation variable

%upload the data
%CHANGE 3 variables in the next 2 lines *****
HydeParkRightmaker;
testtusk = HydeParkRight(1:10:length(HydeParkRight),1:3);

%generate model
%CHANGE model*****
[x,y,z] = ackerleyfun(A,B,C,D,E,F,leng,startspot,THETA,PHI,PSI,X,Y,Z);

%find the minimum distances of model points from data points - minimizing
%this pulls the model (tusk axis) toward all parts of the surface data
for j=1:length(testtusk)
    for i=1:length(x)
        alldist(i) = (x(i)-testtusk(j,1))^2+(y(i)-testtusk(j,2))^2+(z(i)-testtusk(j,3))^2;
    end
    totaldist = totaldist+min(alldist);
end
%-----End of Program

```

ackerlyslowfun

A recharacterization of ackerlyfun to plot the tusk in 3D with traces of different colors to make it easy to distinguish them. This program was considerably slower than *ackerlyfun* because that was designed to go quickly for optimization while this was designed to make a figure that looked good. It is called with the same inputs as *ackerlyfun* but its only output is a figure.

```
function [x,y,z] = ackerleyfun(alpha,delta,gamma,sigma,r_0,~,leng,startspot,THETA,PHI,PSI,X,Y,Z)
```

```
%program parameters
```

```

NOP = 6; %Number Of Points in each generating curve
epsilon = 0.5; %growth step parameter
steps = 100*leng;
centroid = [0,0,0];

%initialization
r = r_0;
psi = 0;
surfleng = 0;

%get index of place where data tusk to start on grown tusk
startspot = floor(startspot);
if startspot<2
    startspot = 2;
elseif startspot>steps
    startspot = floor(steps);
end

figure
hold on

for i=2:steps
    %should follow order of translation then rotation, but this starts on
    %second step, so the rotation that occurs first is actually for the
    %previous growth step

    %update orientations
    psi(i) = psi(i-1)+epsilon*tan(delta)*tan(alpha); %Appendix eq. B6
    %expand radius
    r(i) = r(i-1)*(1+epsilon*tan(delta)); %Appendix eqs. B4 & B5
    %move centroid
    centroid(i,:) = centroid(i-1,:)+epsilon*r(i)*[cos(gamma)*cos(psi(i)-sigma+pi/2),cos(gamma)*sin(psi(i)-
sigma+pi/2),sin(gamma)]; %Appendix eq. B3
    %plot generating curve

[tusksurface(surfleng+1:surfleng+NOP,1),tusksurface(surfleng+1:surfleng+NOP,2),tusksurface(surfleng+1
:surfleng+NOP,3)] = circlepoints(centroid(i,:),r(i),NOP,gamma-pi/2,0,psi(i)-sigma);
    surfleng = length(tusksurface(:,1));
    %plot in multiple colors
    [aperture(:,1),aperture(:,2),aperture(:,3)] = circlepoints(centroid(i,:),r(i),NOP,gamma-pi/2,0,psi(i)-
sigma);
    plotsurf = rotor(THETA,PHI,PSI)*transpose(aperture);
    for j = 1:NOP
        colors(j,:) = [floor(j/4) mod(floor(j/2),2) mod(j,2)]; %breaks if NOP reaches 8
        scatter3(plotsurf(1,j),plotsurf(2,j),plotsurf(3,j),'marker','.', 'CData',colors(j,:));
    end
end
end
%scale the axes so they are square

```

```

axisend = 1.2*max([abs(min(plotsurf(:,1))) abs(max(plotsurf(:,1))) abs(min(plotsurf(:,2)))
abs(max(plotsurf(:,2))) abs(min(plotsurf(:,3))) abs(max(plotsurf(:,3)))]);
axis([-axisend axisend -axisend axisend -axisend axisend]);
%-----End of Program

```

rotor

A simple program that allowed a full rotation matrix to be generated from the yaw, pitch, and roll in a single line of code without taking up too much space. It multiplies together three individual right-handed rotation matrices.

```

function answer = rotor(theta,phi,psi)
%Rotates the point (x,y,z) 'theta' around the x axis, 'phi' around the y axis,
%and 'psi' around the z axis. Returns a new (x,y,z).
answer = [cos(psi),-sin(psi),0;sin(psi),cos(psi),0;0,0,1]*[cos(phi),0,sin(phi);0,1,0;-
sin(phi),0,cos(phi)]*[1,0,0;0,cos(theta),-sin(theta);0,sin(theta),cos(theta)];
%-----End of Program

```

circlepoints

A program to generate a circle of points, representing a tusk aperture, given its information. The program is a version of one called *circle* downloaded from the MATLAB help forum, modified to allow rotation of the aperture in the function call. The documentation on the original is left in for reference.

```

function [X,Y,Z]=circlepoints(center,radius,NOP,A,B,C)
%-----
% H=CIRCLE(CENTER,RADIUS,NOP,STYLE)
% This routine draws a circle with center defined as
% a vector CENTER, radius as a scalar RADIS. NOP is
% the number of points on the circle. As to STYLE,
% use it the same way as you use the routine PLOT.
% Since the handle of the object is returned, you
% use routine SET to get the best result.
%
% Usage Examples,
%
% circle([1,3],3,1000,':');
% circle([2,4],2,1000,'--');
%
% Zhenhai Wang <zhenhai@ieee.org>
% Version 1.00
% December, 2002

```

```
%-----  
  
%if (nargin <3),  
% error('Please see help for INPUT DATA. ');  
%elseif (nargin==3)  
% style='b-';  
%end;  
THETA=linspace(0,2*pi,NOP);  
RHO=ones(1,NOP)*radius;  
[X,Y] = pol2cart(THETA,RHO);  
Z(1:NOP)=0;  
PTS=rotor(A,B,C)*[X;Y;Z];  
X=PTS(1,:)+center(1);  
Y=PTS(2,:)+center(2);  
Z=PTS(3,:)+center(3);  
%-----End of Program
```