Automated Generation Of Geometric Eye Models

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at George Mason University

By

Bassam Mutawak
Bachelor of Science
George Mason University, 2019

Director: Dr. Qi Wei, Professor
Department of Bioengineering and Computer Science

Fall Semester 2022
George Mason University
Fairfax, VA

Table of Contents

List of Tables

List of Figures

Abstract

AUTOMATED GENERATION OF GEOMETRIC EYE MODELS

Bassam Mutawak

George Mason University, 2022

Thesis Director: Dr. Qi Wei

Visualization of the ocular motor system is an innovative technique to examine the underlying causes of different ocular disorders. Creating three-dimensional (3D) ocular models, including the extraocular muscles and other ocular structures, is one method for ocular system visualization. Effective examination of the different ocular disorders necessitates these 3D models to be developed in a patient-specific manner, using medical imaging techniques to image a patient's ocular structures and laborious post-processing to generate the three-dimensional models. Biomechanical simulators employ these patient-specific models to simulate eye movements such as fixations and saccades in normal or abnormal conditions. Such realistic computational simulation can be helpful to quantitatively study factors contributing to eye movement disorders and effective surgical treatment procedures. Current patient-specific ocular modeling, however, is limited due to the lengthy initial static model creation process. Furthermore, a recognized pipeline to create these static models does not exist. In this thesis, we introduce an automated pipeline to generate patient-specific 3D ocular models that stream-lines and unifies the multi-step model creation process. Several solutions are compared at step to optimize quantitative accuracy to real-world experimental results. The pipeline is implemented as a plugin in Autodesk Maya and seven subject datasets are used to demonstrate modeling fitness. Modeling creation time is drastically

reduced, enabling quicker turnaround of ocular visualization and allowing for a broad set of ocular models to be leveraged in the development of biomechanical simulators.

Chapter 1: Introduction

1.1 Basics of an Eye Model

Relevant features for ocular modeling has greatly varied based on the utilization of the model. A common feature set of all ocular models are the six extraocular muscles. Actions of these muscles, with their associated connective tissue, exert forces on the globe (eyeball) resulting in various types of eye movement [4]. The medial rectus muscle (MR) and lateral rectus muscle (LR) are the antagonist muscle pair that allow for horizontal rotation of the eyeball. The superior and inferior (SR, IR) rectus muscles rotate the eye vertically. Superior oblique (SO) and inferior oblique (IO) muscles allow for torsion around the line of sight and also contribute to vertical eye movement [4].

Due to anatomical and/or neural deficits, these extraocular muscles can deteriorate, losing strength and effectiveness to cause various ocular conditions. Strabismus is an oculomotor dysfunction in which the eyes are not properly aligned when looking at a fixed point in space. According to the American Association for Pediatric Ophthalmology and Strabismus, 4% of the U.S. population have strabismus [5]. This condition can oftentimes be corrected. A surgical intervention to mitigate the effects of this condition involves recessing or resecting the involved extraocular muscles to balance the forces acting on the eyeball [6]. Multiple clinical exams are usually employed to determine the site and amount of correction.

A possible diagnosis and treatment planning technique for corrective operations against oculomotor disorders is through biomechanical modeling. In general, an accurate biomechanical model requires incorporation of the necessary features in their correct anatomical and mechanical fashion to best represent the system. In the case of biomechanical modeling

eye, this includes: the six extraocular muscles, the ocular globe, and the associated connective tissue. Static models provide geometrically correct representations of these features but do not model muscle actions or eye movement. As a result, these models are less effective for true analysis of oculomotor dysfunctions. Dynamic simulators are more effective in understanding functions of ocular structures and mechanism of eye movement. Along with a static model, dynamic simulations must incorporate realistic biomechanical characteristics of the eye muscles and have to handle muscle neural control and the resultant eye movement [7].

A data-driven ocular modeling solution can leverage ocular features traced from medical imagery to build a subject-specific representation of the extraocular muscles, ocular globe, and connective tissue. A common imaging modality used for this is magnetic resonance imaging (MRI). A patient's ocular features are iteratively imaged using a sequence of image "slices", with the distance between any two slices being 2 millimeters. These 2-dimensional (2D) representations are then registered in 3-dimensions (3D) and several modeling primitives are then adopted to represent the relevant ocular features.

1.2 Overview of Current Eye Models

Muscles are nominally modeled by lines of force that can bend around kinematic wrapping surfaces [7]. Sueda et al. [8] used rigid bodies for bones and spline-based strands to simulate tendons and muscles. This was based on the realistic anatomical structure of muscle, consisting of fibers that contract when muscles are activated. Kaufman et al. [7] incorporate this strand schema into a hand muscle model creation tool. Their process involves constructing the muscle model using an embedded plug-in for an animation tool and exporting the model and animation key frames to a simulation environment. The simulator then calculated the necessary muscle innervations to match the key frames provided and the results were exported back into the animation tool to show movement in a realistic way. While novel, this process is too generalized and cannot be used effectively for an ocular model which has specialized functions different from the musculoskeletal system.

Biomechanical models of the oculomotor plant must incorporate eye muscle anatomies and define new mechanical constraints specific to the eye. Wei et al. [9] developed a strand-based 3D dynamic model of the eye. This model included the distribution of material properties and forces along extra-ocular muscles (EOM) as well as muscle mass. Wei et al. also simulated dynamic change in eye movements, providing more capabilities in examining different types of eye movement than the static models [9]. This simulation environment incorporates specific constraints relevant to the eye such as those on the pulley connective tissues to achieve realistic dynamic movement and is currently the most effective ocular model.

Regardless of the simulation environment, incorporating patient-specific characteristics to any model is an arduous procedure. Each patient-specific dynamic model requires numerous hours of preprocessing and registration to generate the correct geometric disposition and shape of the eye and the extraocular muscles. To aide further research into ocular modeling, this thesis will create a pipeline to stream-line the process of generating clinical data-driven, patient-specific biomechanical models of the eye. The result of this pipeline is a patient-specific, geometrically and mechanically accurate ocular model that can be used in a simulation environment to model dynamic movement of the eye and to examine specific types of strabismus.

1.3 Upcoming Chapters

This remaining chapters will describe the end-to-end pipeline solution adopted in this thesis to automatically generate geometric ocular models. Chapter 2 will introduce the input data and illustrate the registration method adopted. Chapter 2 will then introduce several methods to model the ocular globe with modeling results of each. Next, chapter 3 walks through the modeling of each extraocular muscle and the primitives adopted to represent a muscle strand. With the relevant features geometrically modeled, chapter 4 presents results across several patient sets, a timing analysis, and usage information for the produced plugin. Finally, chapter 5 concludes the thesis.

Chapter 2: Geometric Globe Modeling

Subject-specific modeling necessitates the availability of data illustrating the geometry and position of features of interest (EOM, Globe). Magnetic resonance imaging (MRI) is a modality commonly leveraged for ocular modeling due to the clear outlines of each extra ocular muscle captured.

The datasets used in this project are comprised of T2-Weighted MR images of the oculus sinister (OS, left) and oculus dextrus (OD, right) eyes in both the sagittal and coronal imaging planes. Image slices in each plane are captured at 2 mm intervals, allowing for detailed capture of geometry and position of each feature of interest. Each image slice in available imaging planes is carefully annotated in ImageJ [10] for desired features. Certain features of a muscle's geometry can only be seen in one imaging plane, requiring annotation of each muscle in each plane. Resulting annotations are stored coupled with the image data. Figure illustrates several ocular features at different slice indices and imaging planes. The resulting annotations containing raw point clouds of each interested feature in 3-dimensions are ingested by the upcoming pipeline for further processing.

2.1 Registration

Due to the time and space differential between data capture in different imaging planes, point clouds in the sagittal plane are first spatially registered to the coronal plane. An iterative closest point (ICP) procedure was implemented to register features in the sagittal plane to coronal.

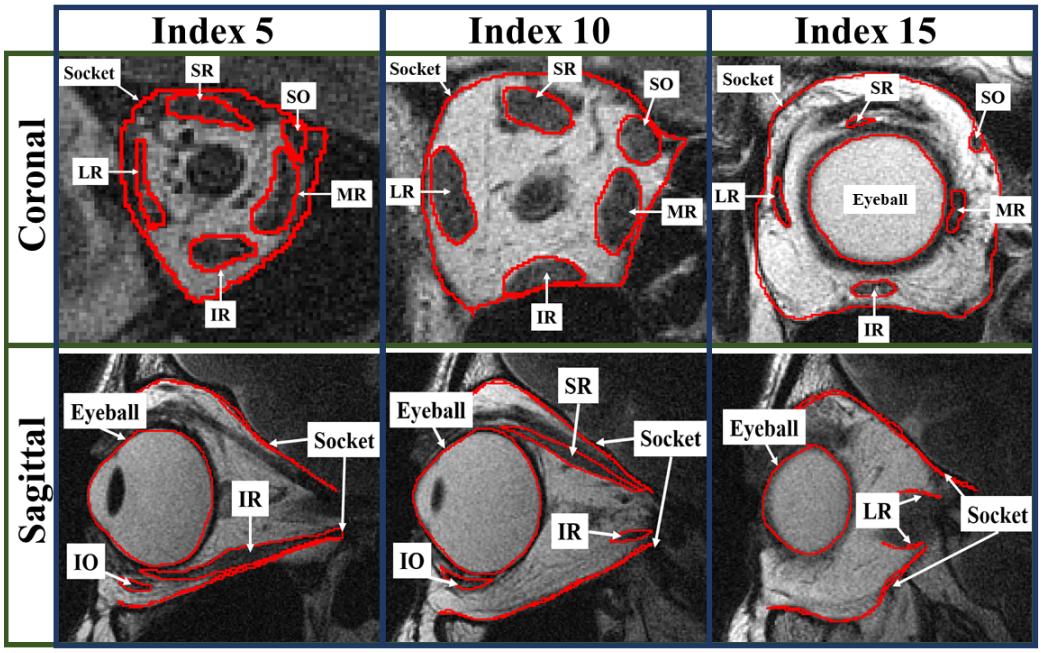

Figure 2.1: MR image slices with desired features annotated at different positions in the eye. Coronal plane (top-row) and sagittal plane (bottom-row).

2.1.1 Pre-Processing

Prior to any registration, input point clouds are processed and formatted to correct for imaging plane orientation and to remove undesired data points. Due to the procedure enacted during imaging of patient data, data points collected in the coronal plane are subject to non-consistent orientation. This can be observed in the left in figure 2.2. A correct orientation of the coronal plane point cloud is illustrated in the right in figure 2.2, where a line segment intersecting both the medial and lateral rectus centroids is parallel with the horizontal axis (yellow line). Using the medial and lateral rectus centroids, the angle θ is computed using Pythagorean Theorem, where the adjacent and opposite lines are the yellow and blue segments respectively. This θ is computed for three separate slice indices in the coronal plane and an average θ_{avg} is used to rotate the coronal point cloud around the antero-posterior axis.

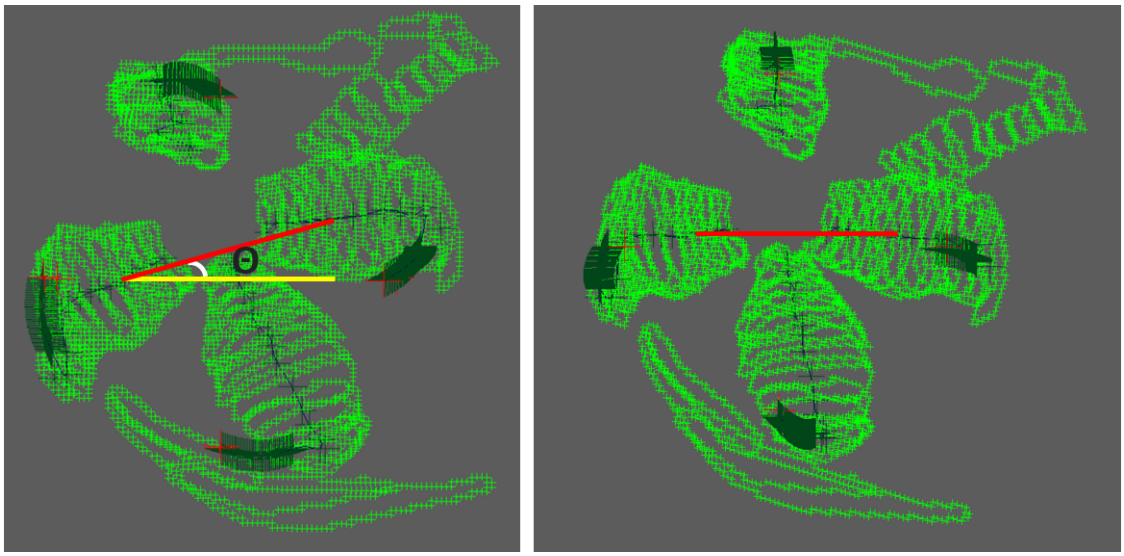

Figure 2.2: Coronal point cloud before (left) and after (right) orientation correction. Yellow line is the line parallel with the horizontal axis and the red line is the line segment connecting the centroids of the MR and LR muscle point clouds. The orientation angle theta is computed as described above.

2.1.2 Registration Algorithm

Iterative closest point is a registration algorithm commonly used in robotics applications for pose estimation [11]. Generally, in the point-to-plane implementation of this algorithm, each point in the data to be registered (desired set) is matched with the closest point in the target data (target set). The estimated rotation and translation necessary to align the desired set to the target set is computed using singular value decomposition (SVD). The rotation and translation matrices computed each iteration is then applied to the desired set. This procedure is repeated until a desired objective is met. In our implementation, the root mean square (RMS) error was used as the distance metric during computation as well as the convergence criteria. Furthermore, during each iteration errors outside of two standard deviations from the mean of all errors were removed from consideration during SVD. This is necessary to account for non-overlapping points between the desired and target sets as well as to account for noisy point clouds due to tracing inaccuracy.

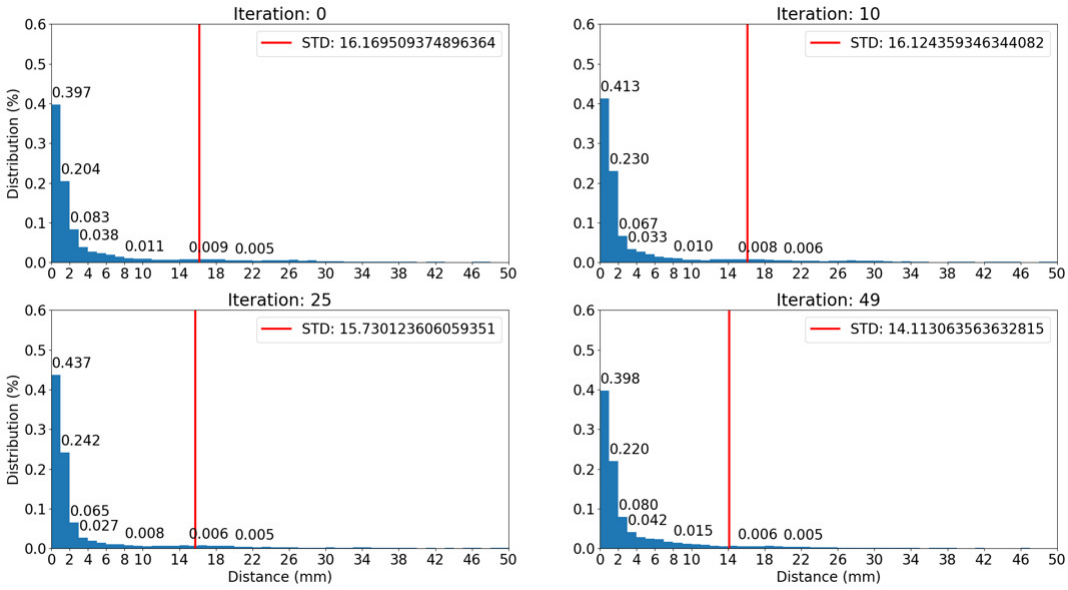

Figure 2.3: Distance distribution of points during ICP at different iterations. Two standard-deviation line (red vertical line) shown for illustration.

Due to the coarse image slice resolution (2mm) compared to pixel resolution as well as shape of the ocular structures, sagittal features depict the ocular structures differently from the coronal features. Therefore, it is hard to find a uniform alignment applicable to all ocular features. We decided to use only a subset of features in the sagittal plane that most match the coronal features registration and the resulting matrix is then applied to all sagittal point clouds. Multiple sagittal features were evaluated for optimal alignment (lowest RMS error). Table 2.1 depicts the various combinations explored with the average RMS error of all sagittal points after the resulting matrix was applied. An aggregate point cloud composed of the eyeball and eye socket in the sagittal plane resulted in the lowest RMS error. This result is expected due to the anatomical properties of the eyeball and socket. The eye socket is a rigid structure which remains constant in shape between imaging procedures. While the eyeball is non-rigid, shape is normally maintained during primary gaze. In addition, these two ocular structures have comparable dimensions in coronal and sagittal planes, which result in sufficient features extracted in either directions.

Table 2.1: Average RMS error (mm) and runtime (ms) observed across all feature combinations tested.

	Sagittal Features			
	LR + MR	SR + IR	Eyeball + SO	Eyeball + Socket
Average RMS Error (mm)	32.1	30.8	24.5	23.8
Runtime (s)	0.238	5.170	9.840	103.30

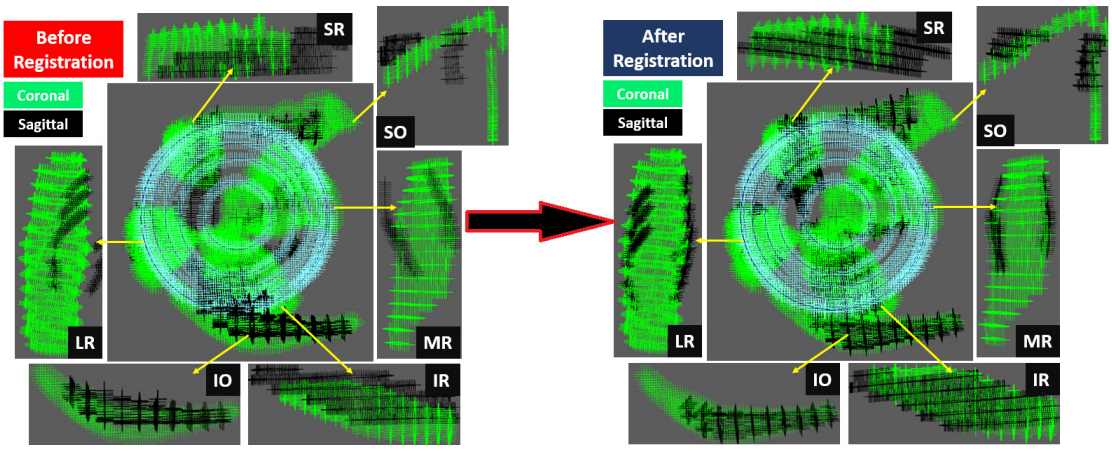

Figure 2.4: Feature point clouds before (left) and after (right) registration.

2.2 Globe Fitting

With the sagittal and coronal point clouds now registered, multiple imaging planes can be leveraged in modeling geometric features. Subject-specific modeling of the eyeball is a critical step in the mechanical model development process as each eyeball geometry is unique. Lever arms in muscle torque computations during mechanical simulation are highly dependent on the radius of the eyeball, thus any geometry being fit to the eyeball point cloud must be accurate.

Ultimately, the eyeball is an ellipsoid with the lengths of the three axis (especially the long axis) being entirely variable between patients. Fitting a geometry to a set of points in 3D has been studied extensively in literature [12–15], with each method adopting different assumptions and criteria. Several geometry fitting techniques were explored with the goal of generating a shape that satisfiably encapsulates the eyeball point cloud and is an agreement with literature on certain eyeball properties (verified in chapter 2.3). The globe modeling space explored is subject to two challenges. First, due to imaging protocol, there are "dead zones" which lack points imaged (e.g. anterior half of the globe in coronal plane). Second, there may be human error in traces ingested by the module. As a result, the optimal globe fitting method must not fail in either (or both) scenarios. The methods explored were: least-squares spherical fit, least-squares ellipsoid fit, and an iterative ellipsoid fit. Each method was implemented and evaluated using features from select imaging planes (coronal, sagittal, and both planes combined) for an optimal fit.

A sphere's general equation in 3D Cartesian coordinates can be described as the following:

$$(x - x_0)^2 + (y - y_0)^2 + (z - Z_0)^2 = R^2 \tag{2.1}$$

where the sphere center is located at (x_0, y_0, z_0) with radius R. A least-square method of approximating a sphere using a point-cloud can be done by rearranging equation 2.1 to the below equation.

$$x^2 + y^2 + z^2 = 2xx_0 + 2yy_0 + 2zz_0 + R^2 - x_0^2 - y_0^2 - z_0^2 \tag{2.2}$$

This can be put into matrix form and the sum of square residuals minimized using least-squares.

$$\vec{f} = A\vec{c} \tag{2.3}$$

$$\vec{f} = \begin{bmatrix} x_i^2 + y_i^2 + z_i^2 \\ \vdots \\ x_n^2 + y_n^2 + z_n^2 \end{bmatrix} \qquad A = \begin{bmatrix} 2x_i & 2y_i & 2z_i \\ \vdots & \vdots & \vdots \\ 2x_n & 2y_n & 2z_n \end{bmatrix} \qquad \vec{c} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ R^2 - x_0^2 - z_0^2 \end{bmatrix}$$

A similar least-square fit for an ellipsoid can be performed. The equation for an ellipsoid is described in general below (equation 2.4), with $\beta_a = \frac{1}{a^2}, \beta_a = \frac{1}{b^2}, \beta_a = \frac{1}{c^2}$. Placing this into the least-square archetype, the model parameters that minimize the L2 norm between the model and point cloud are desired.

$$\beta_a x^2 + \beta_b y^2 + \beta_c z^2 = 1 \tag{2.4}$$

An iterative method to fit an ellipsoid to data points was introduced by Kovac et al. [16]. The authors adopted an algebraic approach using an implicit equation of an ellipsoid,

and propose to solve a minimization problem described in the following equation (2.5)[16].

$$\begin{aligned} &\underset{(A,b,c)\in M}{\text{minimize}}\, G(X,A,b,c), \\ &\text{where } \{(A,b,c) \in R^{d\times d} \times R^d \times R, A \geq 0, Tr(A) = 1\}, \\ &G(X,A,b,c) = \sum_{i=1}^{n} (\langle x_i, Ax_i\rangle + \langle b, x_i\rangle + c)^2 \end{aligned} \tag{2.5}$$

Per equation 2.5, given a set of n points $X = (x_i)_{i\leq i\leq n}$ in R^d where $d = 3$, one can minimize the residual $G(X, A, b, c)$ over a set M of admissible triplets (A, b, c) using an implicit equation of an ellipsoid.

Each geometry fitting method described above was implemented in Python, leveraging author-provided resources where applicable. Intuitively, one can expect a least-square fit using an ellipsoidal point cloud to show the least desirable results (qualitatively because the eyeball is not a perfect sphere), but was explored nonetheless for completeness. A least-square minimization of an ellipsoid equation is expected to generate fit qualitatively closer to the point cloud than a sphere fit, but is susceptible to noise. This geometry fitting method does not account for a point cloud not aligned with the coordinate system axis. As a result, for point clouds with unique orientation or non-standard rotation, this method is expected to fail. The iterative method proposed by [16] is both robust to non-aligned point clouds and implemented for an ellipsoidal geometry. Table 2.2 illustrates the three fitting methods explored.

Table 2.2: Comparison of globe fitting methods explored in this thesis.

Fitting Method	**Expected Geometry**	**Dimensions Accounted For**
Least-Square Fit	Sphere	Translation
Least-Square Ellipsoid	Ellipsoid	Translation
Iterative Method[16]	Ellipsoid	Translation, Rotation

Different combinations of point clouds were used to generate eyeball geometric models using each of the methods described. For each method, either the sagittal eyeball point cloud, coronal eyeball point cloud, or an aggregate point cloud of sagittal and coronal eyeball points were used. Robustness to noise in orientation and rotation, qualitative fit, and comparison to expected clinical results (see section 2.3) were all used to evaluate each geometry fitting method. Figure 2.2 displays a qualitative comparison between each of the geometry fitting methods implemented. While difficult to illustrate in two-dimensions, the least squares sphere fit generally computed a globe larger than the input point cloud across all datasets tested. Expectedly, this issue was most prevalent in areas with missing data points (anterior using the coronal point cloud) and less so when both coronal and sagittal point clouds were combined. Conversely, the least-square ellipsoid fit resulted in computed globes generally smaller than the input point clouds. As more parameters are estimated in this method (radii in three axis), explicitly accounting for noise in multiple dimensions (translation, rotation) is necessary. The robust iterative ellipsoid fit procedure accounts for this noise and is further improved by use of a combined point cloud as depicted.

2.3 Clinical Comparison

While a qualitative inspection of the fitted geometry to the eyeball point cloud is necessary, a quantitative comparison to currently used clinical methods is helpful. The measure adopted in this study focuses on the center point of the generated geometry using the eyeball point cloud and was introduced by Robert Clark and is summarized in the following [1, 2]. In that study, patient-specific MR images were examined and the eyeball is meticulously traced using ImageJ [10] when visible in a given image slice. The origin of the coordinate system is defined by the MR image containing the globe-optic nerve junction. The top-left point of the image along with the antero-posterior distance (mm) are defined as the origin, with the x, y being in the image coordinate system. Three segmentations from separate image slices are extracted around the equator of the eyeball. The anterior distance, area, and

Table 2.3: Comparison of globe center positions estimated by proposed methods and method used by Dr. Clark et al. [1,2]. OD: oculus dextrus - right eye; OS: oculus sinister - left eye.

	OD			OS		
	Lateral	Superior	Anterior	Lateral	Superior	Anterior
Coronal Ellipsoid Fit	37.64	41.44	11.83	42.59	43.81	15.37
Sagittal Ellipsoid Fit	38.60	41.77	11.17	41.63	43.80	13.98
Coronal + Sagittal Ellipsoid Fit	38.08	41.57	11.43	41.78	43.69	14.29
Clinical Standard	38.02	41.54	12.33	41.97	43.69	13.77

2-dimensional centroid are extracted using ImageJ and used as input to compute the 3D center of the globe.

With the above 2D centroids, the clinical method of computing the globe center simply takes the average x and y positions computed from the extracted centroids to approximate as the globe center x, y. Two MR image datasets of patients were used to compute the globe center using Dr. Clark's method and the geometry fitting approach described in section 2.2. An automated script was provided by Dr. Clark to compare our results to his. Table 2.3 compares the globe center estimated results. The ellipsoid geometry fit using only the sagittal eyeball point cloud displayed the maximum difference to the average 2D method (0.86 mm average across all datasets tested). An iterative ellipsoid fit using an aggregate point cloud composed of the sagittal and coronal eyeball points results in the closest estimates with an average 0.63 mm distance from the clinical standard. Based on these qualitative evaluations and quantitative comparisons to the existing method, the iterative ellipsoid fitting methodology was chosen to model the geometry of the eyeball in all datasets.

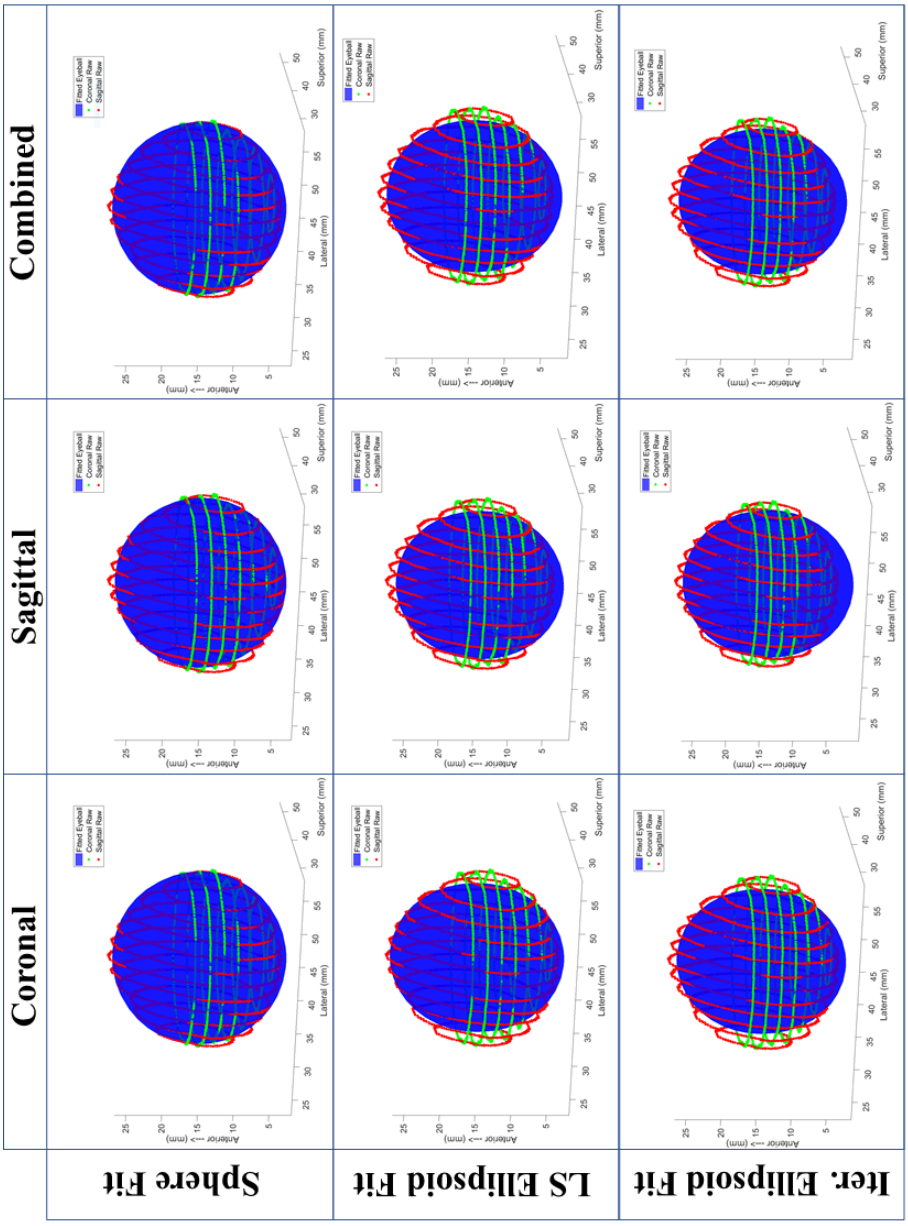

Figure 2.5: Qualitative comparison between each eyeball geometry fitting method using various point clouds as input.

Chapter 3: Eye Muscle Modeling

Accurately representing the extraocular muscles using MR data is critical to ensure the patient's anatomical characteristics are incorporated. The geometry (length) and path (curvature around globe) a patient's extraocular muscles display can vary. A patient-specific model must be flexible and account for the MR data. To our knowledge, this data-driven EOM modeling has not been done yet. To address these challenges, a data-driven pipeline to automatically generate modeling primitives is implemented - conforming to a given subject dataset's unique characteristics while maintaining accurate muscle length and paths from literature. Figure 3 depicts the high-level pipeline implemented in this chapter.

With the required point clouds sufficiently registered, several modeling primitives are generated to represent each feature of interest. First, muscle centroids are computed from each traced muscle feature. Next, muscle insertion locations in 3D are computed using expected locations in literature and through data-driven methods. B-Spline curves are then fit using the computed centroids and insertion points for all extra-ocular muscles (generating

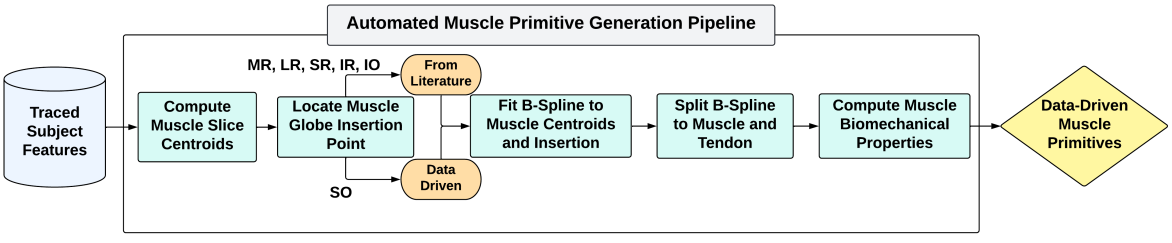

Figure 3.1: High-level overview of automated data-driven muscle primitive generation pipeline implemented in this chapter.

six curves total) representing the whole muscle and tendon structure. Third, the fitted b-spline are split into muscle and tendon portions based on expected lengths from literature. Finally, several mechanical properties (required for biomechnical modeling) are computed for each muscle. This results in a complete static model of all relevant ocular features for dynamic simulation.

3.1 Muscle Bsplines

3.1.1 Muscle Centroids

The MR images ingested by the implemented pipeline contain per-image traces of each present extra-ocular muscle. The traces are labelled to correspond to a single muscle. The distance between two traces in sequence is 2mm. Using this information and the previously modelled globe, all input points are transformed to a coordinate system such that the origin $(X = 0, Y = 0, Z = 0)$ is the center of the globe, $+Z$ is the direction anterior to the globe, $+Y$ is superior to the globe, and $+X$ is medial (towards the mid-line of the body) to the globe.

To seed generation of muscle primitives, centroids are computed for each sequence of muscle traces by computing the average X and Y coordinate of each polygon trace for a given slice Z. These sets of muscle centroids are strictly data-driven, with no assumptions made beyond traces being sequential. These extracted centroids are used to represent a portion of the muscle path, and a literature-driven approach is used to add the insertion muscle insertion points on the globe.

3.1.2 Insertions

Extraocular muscle insertion points are the muscles' attachments on the eyeball. Accurate modeling of the insertion locations is critical as they impact the forces exerted on the eyeball. Table 3.1 displays the relative angular distance (in mm) from the cornea of each EOM insertion that were obtained from literature [3]. These latitude α and longitude θ

were used to compute the 3-D coordinates of of the five EOM using equation 3.1.

$$X = R * cos(\alpha) * sin(\theta)$$

$$Y = R * sin(\alpha) * sin(\theta) \tag{3.1}$$

$$Z = R * cos(\theta)$$

Each of the EOM listed in table 3.1 follow a simple path from the muscle insertion. As evident in figure 2.2, each of these muscles extend either straight posteriorly (MR, LR, SR, IR) or medially (IO) after the insertion point. The superior-oblique muscle however displays a more complex path. The SO extends medially then continues laterally and posteriorly in a sharp turn as displayed in figure 4.2. Furthermore, the SO muscle tendon is not visible in MR imaging acquired which is an added modeling challenge. Due to this complex variability, a more intelligent placement of the SO muscle insertion is necessary to maintain a natural wrapping around the globe with respect to each subject's SO muscle anatomical characteristics.

To accomplish this, a data-driven method to generate SO muscle points that wrap around the globe is adopted. First, a vector is fit to the sagittal (black) SO muscle centroids closest to $Z = 0$ (as shown in figure 4.2), starting at the most medial ($+X$) sagittal point cloud and pointing to the most lateral ($-X$). This segment is then projected down onto the globe to simulate the SO muscle wrapping around the globe. Additional points are added in the direction of the vector to a length expected of the SO muscle. These generated points (along with the centroids) enable the most natural wrapping of the SO muscle following the trajectory of the traces.

3.1.3 Muscle Path

Muscle paths are defined by the registered point clouds and trajectory. With the muscle control points (centroids and insertions) computed previously, modeling primitives can be

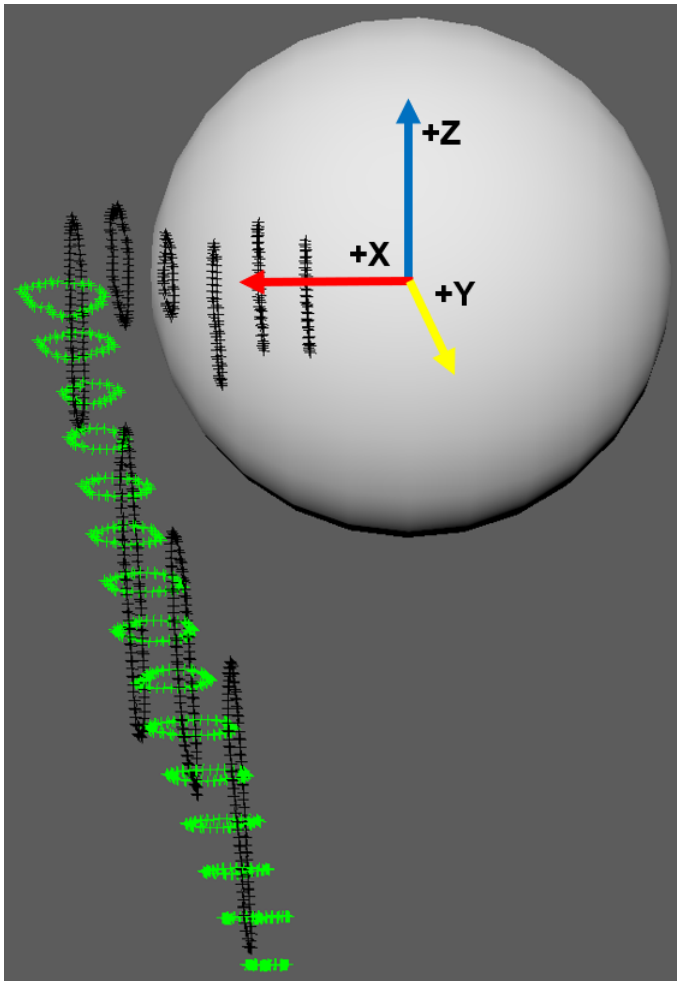

Figure 3.2: Top-down view of SO muscle point cloud containing the sagittal traces (black) and coronal traces (green) after registration described in chapter 2. The coordinate system shown is globe-centric.

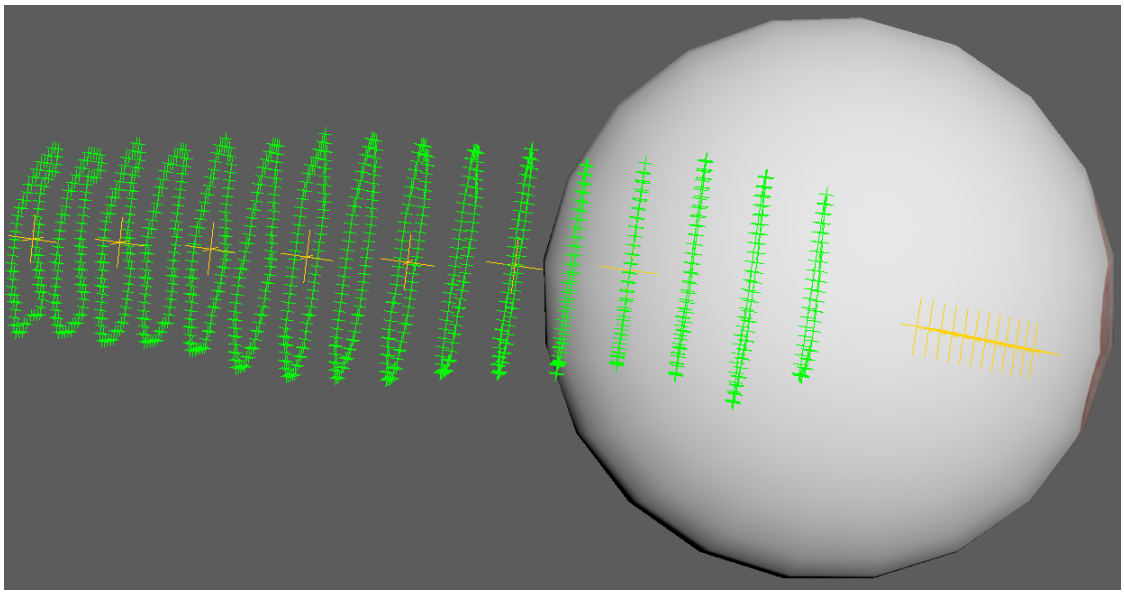

Figure 3.3: Medial-Rectus muscle control points (yellow points) after discarding every-other point to reduce model and tracing noise.

estimated to reflect the subject-specific anatomical properties of each muscle. To enable a broader view of the muscle path and be less susceptible to noise, certain control points were discarded. Beginning with the first (furthest anterior) muscle control point computed in chapter 3.1.1, every other centroid is discarded and the remaining centroids were used to estimate a quadratic b-spline. Figure 3.1.3 depicts the MR muscle control points (in yellow) after this cleaning process. Note only every other muscle trace (green) contains a centroid location. In addition, as ocular muscles are not completely traced through their entire length due to imaging complexity, additional points were added connecting the tangent line of the muscle b-spline path and the globe to the muscle insertion point provided by literature. Using this procedure that supplies path approximation points, control points were estimated to define each muscle path as a continuous and smooth b-spline curve.

Compared to the other extraocular muscles, the SO muscle exhibits a complex structure that necessitates both coronal and sagittal point clouds to describe its geometry (see figure 3.4C). As a result, a more intelligent solution is required. First, it is important to note

the overlapping regions in the coronal and sagittal point cloud for the SO muscle which is near the troclea. While this overlapping region is critical to register the SO muscle between the different imaging planes, the sagittal overlapping region is not as useful as those in the same region but from the coronal planes. The reason is that the coronal tracings provide anatomically more accurate approximation of the SO surface. Therefore, during SO muscle path generation, tracing points in the sagittal planes near the troclea region is first removed from the overall SO point cloud and the remaining slice centroids are used to compute the control points similarly to the other muscles.

3.1.4 Muscle B-Spline

The primitive adopted in this study to represent a muscle path is the b-spline curve, which is the geometric representation of a strand [8]. A b-spline curve is formally defined as a piece-wise polynomial curve composed of a linear combination of control points P_i and b-spline basis functions $N_{i,k}(t)$. The parameter k denotes the degree of each polynomial segment. This is formulated in equation 3.2.

$$P(t) = \sum_{i=0}^{n} P_i N_{i,k}(t) \tag{3.2}$$

The Maya command *fitBspline* was used to fit a b-spline curve to the previously computed control points. Refer to the Maya documentation here for implementation details. Each b-spline was computed with degree 3. The resulting b-splines generated from this procedure are displayed in figure 3.4. The figure illustrates how the generated b-spline accurately reflects the traced muscle coordinates.

3.2 Muscle and Tendon Length

Most EOMs are modeled as a muscle component and a tendon component connecting together. This is a property of all EOMs except for the inferior oblique muscle with no

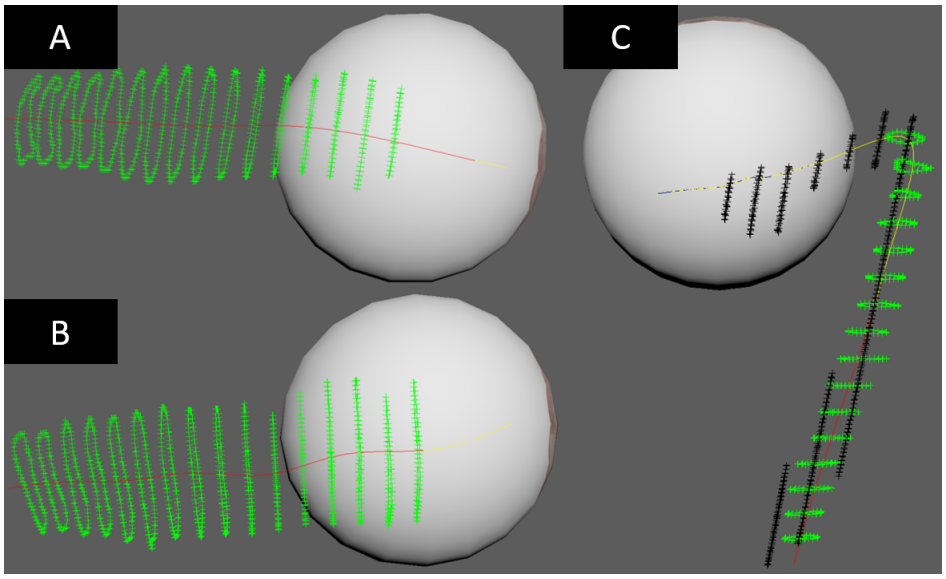

Figure 3.4: EOM geometries represented by b-spline curves that are generated using the procedure. Green and black point clouds are muscle traces in the coronal plane and sagittal plane respectively. The MR (A), LR (B), and SO (C) muscle b-splines are shown.

tendon component. As a result, the generated muscle b-splines must be split into their muscle and tendon components. This is done based on the length of the generated b-spline and the known lengths of muscle and tendon pairs provided by literature [3]. Starting at the muscle insertion point, part of the b-spline curve of each muscle (except IO) is defined as tendon based on the tendon length. The remaining of the b-spline curve is defined as the muscle. Further, the muscle portion is extended posteriorly to maintain a common origin plane for all muscles. This is necessary because MR imaging does not scan sufficiently posterior to depict origins of the EOMs. Table 3.2 depicts the muscle and tendon pair lengths of one subject data sets compared to literature. As shown, all generated muscle and tendon b-spline adhere to the anatomical measurements in literature. Literature only supplies an average length for the SO and IO muscles (40mm and 37.5mm respectively), thus a range was inferred in this paper.

3.3 Mechanical Properties of Muscles

With muscle geometry generated, the final step in the pipeline is to couple each EOM with the necessary properties for biomechanical simulation. This includes muscle pulleys and radii in addition to various other mechanical properties.

MRI studies have shown that the paths of rectus muscles are fixed relative to the orbital wall. This negates any side-slip of muscles during movement. Pulley locations are defined as the locations where the rectus extraocular muscles change their paths in vertical eye movement. Additionally, this point is used with the muscle origin (most posterior point), to emulate the orbital layer of each muscle [4].

For muscle (and tendon) radii, each control point on the EOM b-splines is mapped to the closest muscle trace. The muscle (and tendon) radii at that point is computed as $R = \frac{1}{N}\sum_{i=1}^{N}\left(\vec{X} - \vec{x}\right)^2$ where $\vec{X}$ is the center point of the slice trace and $\vec{x}$ is a point within the slice trace for N points in the trace. This is done for all EOMs and the muscle radius at each b-spline control point is coupled with the muscle during model export.

Finally, several biomechanical properties are computed per muscle. Table 3.3 depicts all properties stored and their default values.

Table 3.1: Extra ocular muscle insertion point distance relative to cornea, in millimeters.

	MR	**LR**	**SR**	**IR**	**IO**
Lateral	-10.85	12.45	0	0	19.5
Superior	0	0	12.65	-11.85	2.0

Table 3.2: Muscle and tendon lengths for a single subject dataset compared to expected ranges provided by literature [3]. All measurements in millimeters.

Muscle	**Muscle Length**	**Muscle Exp. Length (Range)**	**Tendon Length**	**Tendon Exp. Length (Range)**
MR	40.34	32 - 44.5	2.99	1 - 7
LR	36.08	24 - 42	7.20	4 - 11
IR	37.55	33 - 42.5	4.699	3 - 7
SR	37.47	31 - 45	4.3	2 - 7
IO	38.5	35 - 40	N/A	N/A
SO	44.65	38 - 47	16.23	13 - 22

Table 3.3: Muscle properties coupled during export. The final three are force-length curves specific to Orbit modeling.

Property	**Default Value**
Physiological Cross Section Area (PCSA)	$R^2_{muscle} * \pi$
Muscle Passive Force	$0.8036 * PCSA$
Muscle Active Force	$0.9878 * PCSA$
Tendon Passive Force	$80.36 * PCSA$
Muscle Damping	$2e^-3$
Tendon Damping	$2e^-3$
Initial Strain	-0.01
Active Force-Length Path	“data/forces/active.xml”
Passive Force-Length Path	“data/forces/passive.xml”
Tendon Force-Length Path	“data/forces/passive.xml”

Chapter 4: Results and Generated Models

The methodology defined in the previous chapters was implemented and used to generate 7 subject-specific geometric eye models using manual traces from MRI data. The pipeline was implemented as a plugin for Autodesk Maya and can be used with minimal modification to the 2020 release. Each model is packaged into a single file for export and later ingest into biomechanical simulators. Finally, a timing analysis is provided to illustrate the value of this pipeline.

4.1 Generated Models

Figure 4.1 depicts the four EOM b-splines generated using the implemented pipeline. Overall, the pipeline generalized well across the 7 subject datasets as seen in both the close approximation of the b-splines to the input traces and realism in areas where no traces are available (the most anterior portion of the SR muscle in Figure 4.1). In situations where the input traces were insufficient, b-splines were extended in Maya to an appropriate length designated by literature. This enabled realistic representation of the most posterior portions of the EOMs without requiring traces present. Complete modeling of the EOMs beyond MRI-based traces is a useful contribution of the pipeline.

The SO muscle, while much more complex, displayed similar attributes using the pipeline. Figure 4.2 shows resulting SO muscle b-spline and placement. Correct registration of the sagittal to coronal plane enabled accurate depiction of the SO muscle as it wraps around the globe. Additionally, the trajectory of the SO muscle as it wraps around the globe is different for each subject and eye. The natural wrapping of the globe is due to the unique SO fitting approach implemented in this pipeline. Finally, the pipeline performs well even

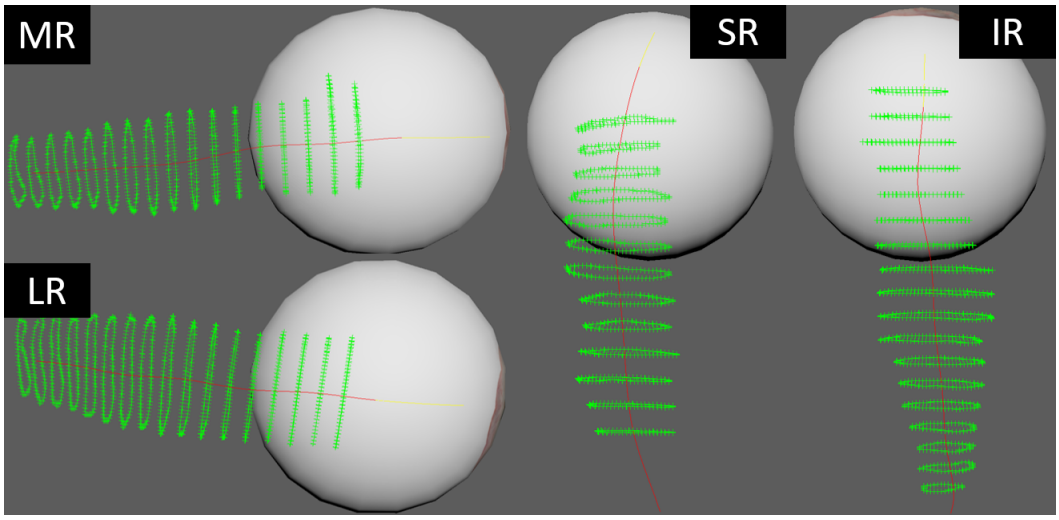

Figure 4.1: Four EOM generated b-splines presented, each from a different dataset to show diversity. Red line indicates muscle while the yellow line indicates tendon.

in situations where the sagittal (black) muscle traces are missing, which is the case demonstrated by the lack of SO sagittal traces in the OS eye in figure 4.2 (left). Additionally, the right dataset of figure 4.2 is significantly missing coronal data towards the posterior portion of OD. The b-spline nonetheless extends posteriorly, following the natural path provided by the few coronal (green) traces available. This demonstrates the effectiveness of the model to fill-in missing data in a natural and realistic way.

4.2 Computational Time

According to expert researchers with experience manually creating subject ocular models from traces, the average time to develop a realistic model suitable for biomechanical simulation is estimated to be 20 hours. Figure 4.3 shows the recorded time to generate 7 different subject models presented in the form of number of points vs. time (generally linear). As shown, the average run time across all subject sets was 129.6 seconds. An additional 15 minutes was spent fine-tuning the model after automated generation. This results in an average model generation time of 1,029.6 seconds, a dramatic improvement over the manual

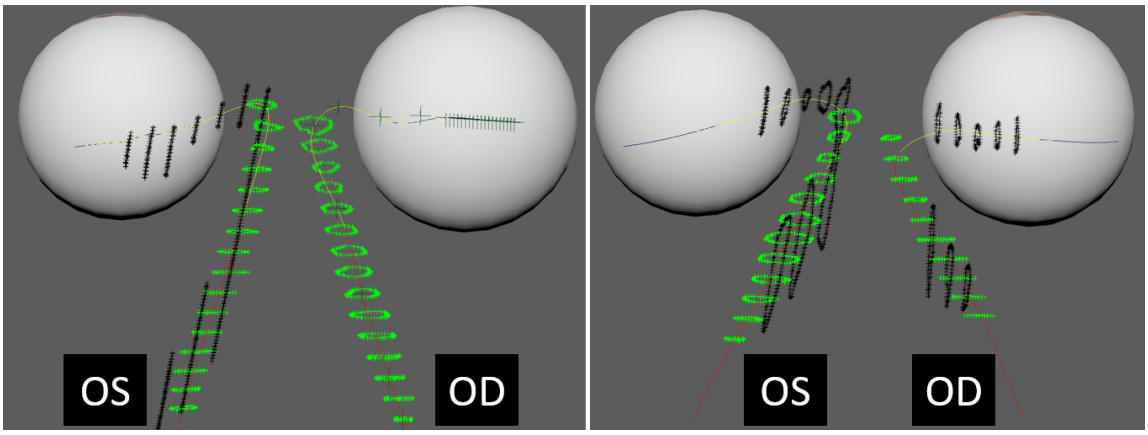

Figure 4.2: The superior oblique muscle shown in a top-down view in two patient sets. Black traces are those ingested in the sagittal plane and green traces are from the coronal plane.

time. This pipeline also involves minimal user interference. Users are able to fine-tune the final model to account for outliers if needed, but is seldom required. This enables high repeatability in addition to a standardized model generated regardless of the user.

4.3 Model Interface

The plugin was developed as a Python library to leverage the built-in Autodesk Maya 2020 Python 2.7 interpreter. One must first follow setup instructions provided to set up the environment for development. After the environment is setup, the library can be used as follows:

1. Open the Script Editor: Windows – General Editors – Script Editor.
2. Click on the "+" icon to add a Python scripts.
3. In the new Python script, import the plugin.

```
import registration_utils
```

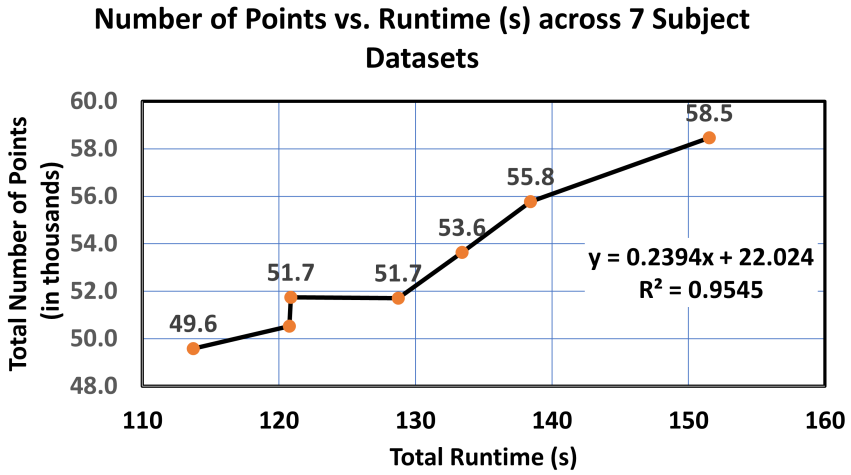

Figure 4.3: Model generation time for seven subjects presented in the form of number of points (in thousands) vs. time (s).

4. Then call the plugin entry point with a specific path to a subject dataset.

```
folder_path = r"full_path_to_folder"
muscles = registration_utils.registerEye(folder_path)
```

5. Run the script and wait for it to finish. Once done, you should see several statistics in the output window such as runtime and average RMS error.

6. After making modifications to the generated model in Maya (if any), simply run the export feature as follows. This will generate a JSON file of the encoded model at the provided file path.

```
from shape_utils import export
out_file_path = "path_to_out.json"
export(muscles, out_file_path)
```

4.4 Export

The exported format compresses all relevant modeling information to a single json file. All EOMs (SR, IR, LR, MR, SO, IO) and eye globes for each eye (left, right) are stored with their corresponding attributes and locations. All position information provided is with respect to the globe center as the origin. An example output file is provided in Figure 4.4. In addition to the attributes listed in table 3.3, the input slice radii and centroids as well as the generated b-spline control vertices are provided per muscle and tendon b-spline. For the eye globes, the parameterized spans (U, V) are provided in addition to the radius. All dimensions provided are in millimeters.

```
{
    "OD": {
        "muscles": {
            "SR": {
                "orbital_points": [
                "tendon_centroids": [
                "muscle_radii": [
                "muscleActive": 58577.25974161748,
                "musclePassive": 47654.065527803,
                "muscle_init_strain": 0.5,
                "tendon_radii": [
                "passiveFL": "data/forces/passive.xml",
                "initialStrain": -0.01,
                "name": "SR",
                "activeFL": "data/forces/active.xml",
                "add": true,
                "tendonDamping": 0.002,
                "muscle_bspline_cvs": [
                "tendon_bspline_cvs": [
                "tendonFL": "data/forces/passive.xml",
                "PCSA": 1.6863199206606042e-05,
                "muscleDamping": 0.002,
                "tendon_init_strain": 0.5,
                "tendonPassive": 4765406.5527803,
                "muscle_centroids": [
            },
            "IR": {
            "SO": {
            "IO": {
            "MR": {
            "LR": {
        },
        "Eyeball": {
            "eyeball_cvs": [
            "spansV": 10,
            "spansU": 19,
            "radius": 0.0116
        }
    },
    "OS": {
}
```


Figure 4.4: Sample exported JSON file for a given subject dataset. Horizontal lines indicate skipped numbers for illustrative purposes. Only one muscle's attributes are shown for demonstration.

Chapter 5: Conclusion

Biomechanical simulation of 3D ocular models is an important method to examine underlying causes of different ocular disorders and to computationally experiment with various surgical treatment options. Manual generation of subject-specific geometric ocular models is an incredibly exhausting and time consuming task, with an average manual turnaround time of 20 hours split across several days for a single researcher. Further more, manual model generation is subject to operator dependent variability affecting accuracy of the model and simulation. In this study, we introduce an end-to-end pipeline implemented as a plugin in Maya for automatic generation of subject-specific ocular models using traced ocular structure data from MR images. An ICP algorithm was implemented to register the different input imaging planes. Several sphere fitting algorithms were also assessed for accuracy and quality in generating a globe sphere using the input points. Using information from literature and the input traces, b-spline curves were computed using the input traces to optimally model the muscle and tendon paths. Finally, the models were packaged in an easy-to-use single file and properly loaded into simulation software for further analysis. Several models using patient sets were generated in this study and provided to an experienced clinician for evaluation with no negative feedback received. The source code and instructions will be made freely available to all via a github repository link.

Bibliography

[1] R. A. Clark, J. M. Miller, and J. L. Demer, "Three-dimensional location of human rectus pulleys by path inflections in secondary gaze positions," *Investigative Ophthalmology Visual Science*, vol. 41, no. 12, p. 3787–3797, Nov 2000.

[2] J. L. Demer and R. A. Clark, *Chapter 8 - Translation and eccentric rotation in ocular motor modeling*, ser. Mathematical Modelling in Motor Neuroscience: State of the Art and Translation to the Clinic. Ocular Motor Plant and Gaze Stabilization Mechanisms. Elsevier, Jan 2019, vol. 248, p. 117–126. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0079612319301128

[3] G. K. v. N. MD and E. C. C. MD, *Binocular Vision and Ocular Motility: Theory and Management of Strabismus*, 6th ed. St. Louis, Mo: Mosby, Sep 2001.

[4] J. L. Demer, *Mechanics of the Orbita*. Basel: KARGER, 2007, vol. 40, p. 132–157. [Online]. Available: https://www.karger.com/Article/FullText/100353

[5] "What is strabismus?" 2015. [Online]. Available: https://aapos.org/glossary/strabismus

[6] Z. Chaudhuri and J. L. Demer, "Long-term surgical outcomes in the sagging eye syndrome," *Strabismus*, vol. 26, no. 1, p. 6–10, Jan 2018.

[7] A. Kaufman, "Interactive tools for biomechanical modeling and realistic animation," Ph.D. dissertation, University of British Columbia, 2008. [Online]. Available: https://open.library.ubc.ca/cIRcle/collections/ubctheses/24/items/1.0051231

[8] S. Sueda, A. Kaufman, and D. K. Pai, "Musculotendon simulation for hand animation," *ACM Transactions on Graphics*, vol. 27, no. 3, p. 1–8, Aug 2008.

[9] Q. Wei, S. Sueda, and D. K. Pai, "Physically-based modeling and simulation of extraocular muscles," *Progress in biophysics and molecular biology*, vol. 103, no. 2–3, p. 273–283, Dec 2010.

[10] J. Schindelin, I. Arganda-Carreras, E. Frise, V. Kaynig, M. Longair, T. Pietzsch, S. Preibisch, C. Rueden, S. Saalfeld, B. Schmid, J.-Y. Tinevez, D. J. White, V. Hartenstein, K. Eliceiri, P. Tomancak, and A. Cardona, "Fiji: an open-source platform for biological-image analysis," *Nature Methods*, vol. 9, no. 7, p. 676–682, Jun 2012.

[11] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek, "The trimmed iterative closest point algorithm," in *2002 International Conference on Pattern Recognition*, vol. 3, Aug 2002, p. 545–548 vol.3.

[12] L. Li, M. Sung, A. Dubrovina, L. Yi, and L. J. Guibas, "Supervised fitting of geometric primitives to 3d point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[13] K. Dierkes, M. Kassner, and A. Bulling, "A novel approach to single camera, glint-free 3d eye model fitting including corneal refraction," in *Proceedings of the 2018 ACM Symposium on Eye Tracking Research Applications*, ser. ETRA '18. New York, NY, USA: Association for Computing Machinery, Jun 2018, p. 1–9. [Online]. Available: https://doi.org/10.1145/3204493.3204525

[14] A. Nguyen and B. Le, "3d point cloud segmentation: A survey," in *2013 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, Nov 2013, p. 225–230.

[15] A. Dimitrov, R. Gu, and M. Golparvar-Fard, "Non-uniform b-spline surface fitting from unordered 3d point clouds for as-built modeling," *Computer-Aided Civil and Infrastructure Engineering*, vol. 31, no. 7, p. 483–498, 2016.

[16] B. Kovac, J. Fehrenbach, L. Guillaume, and P. Weiss, "Fitellipsoid: a fast supervised ellipsoid segmentation plugin," *BMC Bioinformatics*, vol. 20, no. 1, p. 142, Mar 2019.

[17] C. E. Shannon and W. Weaver, *The Mathematical Theory of Communication*. Urbana: University of Illinois Press, 1949.

[18] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, May 2001, p. 145–152.

[19] G. Sharma, D. Liu, S. Maji, E. Kalogerakis, S. Chaudhuri, and R. Měch, "Parsenet: A parametric surface fitting network for 3d point clouds," in *Computer Vision – ECCV 2020*, ser. Lecture Notes in Computer Science, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020, p. 261–276.

[20] R. A. Clark and S. J. Isenberg, "The range of ocular movements decreases with aging," *Journal of American Association for Pediatric Ophthalmology and Strabismus*, vol. 5, no. 1, p. 26–30, Feb 2001.

[21] J. L. Demer, "Current concepts of mechanical and neural factors in ocular motility," *Current opinion in neurology*, vol. 19, no. 1, p. 4–13, Feb 2006.

[22] J. L. Demer and R. A. Clark, "Magnetic resonance imaging of human extraocular muscles during static ocular counter-rolling," *Journal of Neurophysiology*, vol. 94, no. 5, p. 3292–3302, Nov 2005.

[23] R. Kono, R. A. Clark, and J. L. Demer, "Active pulleys: Magnetic resonance imaging of rectus muscle paths in tertiary gazes," *Investigative Ophthalmology Visual Science*, vol. 43, no. 7, p. 2179–2188, Jul 2002.

[24] S. Sueda, G. L. Jones, D. I. W. Levin, and D. K. Pai, "Large-scale dynamic simulation of highly constrained strands," in *ACM SIGGRAPH 2011 papers*, ser. SIGGRAPH '11. New York, NY, USA: Association for Computing Machinery, Jul 2011, p. 1–10. [Online]. Available: https://doi.org/10.1145/1964921.1964934

[25] P. Sachdeva, S. Sueda, S. Bradley, M. Fain, and D. K. Pai, "Biomechanical simulation and control of hands and tendinous systems," *ACM Transactions on Graphics*, vol. 34, no. 4, pp. 42:1–42:10, Jul 2015.

[26] U. Karlsson, T. Kirby, W. Orrison, and M. Lionberger, "Ocular globe topography in radiotherapy," *International Journal of Radiation Oncology*Biology*Physics*, vol. 33, no. 3, p. 705–712, Oct 1995.

[27] N. D. Wycliffe and M. F. Mafee, "Magnetic resonance imaging in ocular pathology," *Topics in magnetic resonance imaging*, vol. 10, no. 6, p. 384–400, Dec 1999.

[28] D. Zhou, A. Abass, A. Eliasy, H. P. Studer, A. Movchan, N. Movchan, and A. Elsheikh, "Microstructure-based numerical simulation of the mechanical behaviour of ocular tissue," *Journal of The Royal Society Interface*, vol. 16, no. 154, p. 20180685, May 2019.

[29] D. Zhou, A. Eliasy, A. Abass, P. Markov, C. Whitford, C. Boote, A. Movchan, N. Movchan, and A. Elsheikh, "Analysis of x-ray scattering microstructure data for implementation in numerical simulations of ocular biomechanical behaviour," *PLOS ONE*, vol. 14, no. 4, p. e0214770, Apr 2019.

[30] M. A. Schill, S. F. F. Gibson, H. J. Bender, and R. Männer, "Biomechanical simulation of the vitreous humor in the eye using an enhanced chainmail algorithm," in *Medical Image Computing and Computer-Assisted Intervention — MICCAI'98*, ser. Lecture Notes in Computer Science, W. M. Wells, A. Colchester, and S. Delp, Eds. Berlin, Heidelberg: Springer, 1998, p. 679–687.

[31] Z. Yiyi, W. Guangzhi, D. Hui, and J. Yonghong, "Finite element modeling and biomechanical analysis of eyeball and extraocular muscles," in *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*, Jan 2005, p. 4978–4981.

[32] C. Fotsing, N. Menadjou, and C. Bobda, "Iterative closest point for accurate plane detection in unorganized point clouds," *Automation in Construction*, vol. 125, p. 103610, May 2021.

[33] Y. He, B. Liang, J. Yang, S. Li, and J. He, "An iterative closest points algorithm for registration of 3d laser scanner point clouds with geometric features," *Sensors*, vol. 17, no. 88, p. 1862, Aug 2017.

[34] R. Marani, V. Renò, M. Nitti, T. D'Orazio, and E. Stella, "A modified iterative closest point algorithm for 3d point cloud registration," *Computer-Aided Civil and Infrastructure Engineering*, vol. 31, no. 7, p. 515–534, 2016.

[35] J. D. Porter, R. S. Baker, R. J. Ragusa, and J. K. Brueckner, "Extraocular muscles: Basic and clinical aspects of structure and function," *Survey of Ophthalmology*, vol. 39, no. 6, p. 451–484, May 1995.

[36] R. F. Spencer and J. D. Porter, "Structural organization of the extraocular muscles," *Reviews of oculomotor research*, vol. 2, p. 33–79, Jan 1988.

Curriculum Vitae

Education

Master of Science, Computer Science. George Mason University. 2022.

Bachelors of Science, Bioengineering. George Mason University. 2019.

Professional Experience

Machine Learning Engineer. Booz Allen Hamilton. 2021 - Present

- Lead a software team in the development of a health software product to enable search, discovery, and analytics of FDA 510(k) premarket submission documents.
- Implemented NLP modules to extract predicate device and indications for use information from unstructured premarket document text using intelligent keyword search with over 95% accuracy.
- Developed several processing modules to extract semantic information from academic journal articles using RoBERTa question-and-answering natural language processing models.
- Leveraged entity recognition and fuzzy matching approaches to significantly reduce semantic search space from 1M articles to several thousand using Python and the HuggingFace library.

General Engineer. Department of Defense. 2019 - 2021

- Implemented and trained computer vision algorithms in PyTorch using thermal imagery. Evaluated algorithms against program requirements and produced comprehensive algorithm report deliverables.
- Developed a hardware benchmarking framework in CUDA to evaluate accelerator throughput on common deep-learning operations. Benchmarked deep-learning algorithms against multiple accelerators and provided decision aides to leadership.
- Developed a data container format using HJDF5 to store imagery, annotations, and associated meta data. Reduced storage costs by 20% and format was adopted lab-wide for AI/ML training and data sharing.
- Worked in a cross-disciplinary engineering team to develop AI/ML algorithms for aircraft sensor systems.