Shark Sim: A Procedural Method of Animating Leopard Sharks Based on Raw
Location Data

A Thesis

Presented to

the Faculty of California Polytechnic State University

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Katherine Blizzard

April 2013

© 2013

Katherine Blizard

ALL RIGHTS RESERVED

COMMITTEE MEMBERSHIP

TITLE: Shark Sim: A Procedural Method of Animating Leopard Sharks Based on Raw Location Data

AUTHOR: Katherine Blizzard

DATE SUBMITTED: April 2013

COMMITTEE CHAIR: Zoë Wood, Ph.D.

COMMITTEE MEMBER: Franz Kurfess, Ph.D.

COMMITTEE MEMBER: Chris Lupo, Ph.D.

Abstract

Shark Sim: A Procedural Method of Animating Leopard Sharks Based on Raw
Location Data

Katherine Blizzard

Fish such as the Leopard Shark (*Triakis semifasciata*) can be tagged on their fin, released back into the wild, and their location tracked through technologies such as autonomous robots. Timestamped location data about their target is stored. We present a way to procedurally generate an animated simulation of *T. semifasciata* using only these timestamped location points.

This simulation utilizes several components. Input timestamps dictate a monotonic time-space curve mapping the simulation clock to the space curve. The space curve connects all the location points as a spline without any sharp folds that are too implausible for shark traversal. We create a model leopard shark that has convincing kinematics that respond to the space curve. This is achieved through acquiring a skinned model and applying *T. semifasciata* motion kinematics that respond to velocity and turn commands. These kinematics affect the spine and all fins that control locomotion and direction. Kinematic-based procedural keyframes added onto a queue interpolate while the shark model traverses the path.

This simulation tool generates animation sequences that can be viewed in real-time. A user study of 27 individuals was deployed to measure the perceived realism of the sequences as judged by the user by contrasting 5 different film sequences. Results of the study show that on average, viewers perceive our simulation as more realistic than not.

Contents

1	Introduction	1
2	Background	3
2.1	Animation Background	4
2.1.1	Spline Paths	5
2.1.2	Keyframe Animation	10
2.1.3	Skeletal Hierarchical Models	11
2.2	Biology Background	13
2.2.1	Shark Anatomy	13
2.2.2	Types of Swimming Locomotion in Vertebrates	14
2.2.3	Relationship of Amplitude and Frequency of Tail Beat	15
2.2.4	Use of Fins in Swimming Motion	18
2.2.5	Shark Pitch During Movement	20
3	Related Works	25
3.1	Animal Depictions	26
3.2	Sea Life Simulations	27
3.3	Robotics	28
4	Methodology	29
4.1	The Data and the Simulator	31
4.1.1	Nature of the Data	31

4.1.2	Nature of the Simulator	33
4.2	Procedural Pathing	34
4.2.1	Time-Space Path Curvature	34
4.2.2	Space Path Curvature	35
4.2.3	Arc Length Parameterization	37
4.2.4	Analyzing Path	38
4.3	Shark Modeling	40
4.3.1	Hierarchy	41
4.3.2	Locomotion	41
4.3.3	Fin Movement	45
4.3.4	Shark Pitch	46
4.4	Procedural Keyframes	47
5	Results	49
5.1	Realism of Shark Motion	51
5.2	Performance	54
5.3	Future Work	54
5.3.1	Limitations	56
5.3.2	Conclusion	57
A	Sample Shark Data	58
B	Survey Result Table	59
	Bibliography	61

List of Figures

2.1	Comparison of an interpolated spline versus an approximated spline.	5
2.2	Illustration of undesired Hermite curve behaviors.	7
2.3	Fin names and location on model fish.	14
2.4	Theoretical amplitudes of harmonics at a constant velocity	17
2.5	Tail beat frequency verses velocity of <i>Triakis semifasciata</i>	17
2.6	Maximum lateral amplitude by % body length of the dorsal mid-line	18
2.7	Definition of Pectoral fin planes α and β	19
2.8	Pectoral fin angle changes correspondingly with vertical movement	21
2.9	The pectoral fins' two planes change configuration with chord, orientation, and camber	22
2.10	Shark pitch correlates negatively to speed	23
2.11	Shark pitch changes correspondingly with vertical movement	24
3.1	Shark tracking visualization from TOPP	26
4.1	Overview of simulation pipeline during runtime	30
4.2	Top down view of AUV data plotted on a map.	32
4.3	Path displayed on the screen	34
4.4	Tangent generation in low turning angles.	36
4.5	Tangent generation in high turning angles.	37
4.6	Shark turning angle derived from looking ahead.	39
4.7	Shark polygonal model and rig from dorsal and lateral views	40

4.8	Flow chart for creating swimming motion in a straight line.	42
4.9	A sample of shark model keyframes.	47
5.1	Screenshots of simulation during play through	50
5.2	Comparison of Catmull-Rom spline and our method	51
5.3	Bar graph of surveyed opinion on simulation realism	53

List of Tables

4.1	Variables to insert into Equation 4.4 depending on shark size (cm)	43
4.2	Pectoral fin configurations for rising and sinking	46
A.1	Example internal tracked data	58
B.1	Survey results for 30 clip viewers	60

Chapter 1

Introduction

Understanding the oceans and the animals that live in it is vital, if we want to have a share in the resources the planet offers us. Sharks are a key component in the ocean ecologies they inhabit. We can show our scientific research into sharks to the public by creating visualizations. Visualizations shown to the public educates them of scientific progress and raises awareness of the subject of research.

Past research efforts employ tagging to track leopard sharks (*Triakis semifasciata*) as they move about in the wild. Leopard sharks live off of the North American west coast. In an effort to study them, they can be picked up out of the water, tagged on their fin, and released back into the wild. While following the shark closely with a boat may change the shark's behavior, they can be observed at a distance by technologies such as autonomous underwater vehicles (AUV). A signal emits from the shark's tag. Its location is estimated and recorded along with a timestamp. Previously, this was done to study leopard sharks, guitarfish and other fish[3, 26].

The data are returned as a long list of numbers, indicating position in 2D

space (without water depth), and a timestamp. Points from these kinds of short term tracking missions tend to rest about a meter or two from the next data point. The raw location and timestamp data are difficult for a human reader to parse. Some form of visualization is necessary to interpret the data meaningfully.

Existing shark tracking visualizations show location data projected on a map. They use data gathered over multiple years and thousands of miles. The shark itself is reduced to a flat dot on a map.

We present an alternate method to visualize shark tracking data, more suitable for short term studies with denser data points, or for demonstrating the progress of shark research publicly, in an interesting manner. We demonstrate a method to procedurally animate a model of *Triakis semifasciata* based on the data returned from short term tracking missions. This provides a way to see an estimation of the shark's behavior close up, as if viewing the shark from a user controlled underwater camera nearby. It shows the shark's turns, speedups and slowdowns by displaying a path between data points. It then procedurally translates, orients, and animates a model of the shark according to the path's curvature.

We wish to create simulated animations of the shark's swimming behavior for educational purposes. Our goal was the creation of a real time system, with interactive viewing, which produced animations that would be perceived by a human viewer as 'realistic'. We measure our results via measured run-time performance and a user study, discussed in Chapter 5, Results.

Chapter 2

Background

A series of images flashed in rapid succession create the illusion of movement to a viewer. These images are called frames. Together, frames displayed on a screen form the basic principle of animation [17].

Modern computer animation has roots in the 1960's and 70's, at first by Ivan Sutherland in 1963 with the first interactive graphical program. Vector refresh displays repeatedly draw lines and arcs from instructions organized in a display list. Vector polygons, defined by their vertices, are rastered to the screen by drawing closely spaced horizontal lines in a process called scan conversion. Polygons can be connected together to make a form called a mesh, which then can represent any three dimensional object, be it solid, translucent, manifold or not. Polygonal meshes are one way of representing an object; other methods like volumes, B-splines, NURBS and parametric equations are options that can represent a variety of objects from perfect spheres to clouds [17].

Meshes are satisfactory for our purposes. There are several methods of animating them. Rigid body movement consists of rotations and translations

through space, which is performed by multiplying the vertices in the mesh by rotation and translation matrices. Deformations change the shape of the mesh to simulate concepts like soft bodies and changes in pose. A posable mesh can be represented by a skeletal model which simulates the interaction between a skeleton and its skin, where the mesh is the skin. Individual bones of the model are mapped to vertices in the mesh, such that rotating a bone at an endpoint moves all the vertices in the mesh controlled by that bone. These bones are organized in a hierarchical tree. Rotating one bone will move it and all of its child bones. This way, hierarchical modeling can animate poses for humans and animals, and also pose fantastic forms like dragons and animate objects [17].

A rig defines the relationship between the bones of a hierarchical model and the mesh skin. A simple hierarchical model assigns one bone per vertex, which creates a stiff, robotic skin that breaks when joints bend. Linear vertex blending assigns multiple bones with an influence weight to vertices, which allows smooth looking deformations on the mesh when the skeleton is posed [17].

2.1 Animation Background

The Computer animation techniques we use in our approach include spline paths, keyframes in a pose-to-pose animation system, and hierarchical modeling. Splines can be used to define a smooth path that an object can traverse over time. Hierarchical modeling provides the way to build and move a skeleton to create poses for objects, which the skeleton in turn deforms the outer skin of the model, made of polygons, in a process called skinning. Keyframes define a series of skeleton poses to be interpolated between over time.

Figure 2.1: Comparison of an interpolated spline versus an approximated spline.

2.1.1 Spline Paths

A spline is a series of curves connected together piecewise by control points, called knots. There are two ways to build curves from knots. Interpolation splines ensure that each knot is passed through by the spline. Approximation splines do not necessarily connect each point, but use them as a guideline for their direction, fitting the points as best as possible. Figure 2.1 shows the difference between approximation and interpolation. Splines can be represented with matrices [17].

We are concerned with interpolation splines in this case. Interpolation always includes all of the input points as part of the spline, thus preserving the spline accuracy. In animating an object along a path, the spline acts like a train track for the object to travel along, passing through knots [17].

Because curves are constructed piecewise from curve segments in between knots, we can describe them by their piecewise properties. C^0 continuity describes positional continuity, where all knots are connected together. C^0 continuity describes structures like polyline curves, which connect each knot with a line, with sudden sharp angles at each knot. C^1 continuity describes a spline with

tangential continuity and positional continuity. Tangential continuity is satisfied when the end tangent of one curve is the same as the beginning tangent on the next curve on the same knot. Likewise, curvature continuity, necessary for C^2 continuity, is satisfied when the curvature at the end of a curve is the same as the beginning curvature of the next curve. C^2 continuity requires positional, tangential, and curvature continuity. Computer animation rarely needs continuity beyond the second order, and first order continuity is sufficient to visually create an object's smooth path through space [17].

Hermite splines are interpolation splines that are generated from each point on the path, and its predefined tangent. Two points, p_i and p_{i+1} , and their tangents, p'_i and p'_{i+1} are chosen. To interpolate between them, a value between zero and one, called u in this case, is used to select a specific point in between them. A matrix multiplication involving a coefficient matrix for Hermite splines converts u into a point between p_i and p_{i+1} [17].

$$P(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_i \\ p_{i+1} \\ p'_i \\ p'_{i+1} \end{bmatrix} \quad (2.1)$$

Increasing u at discrete intervals between zero and one will create a curve at that resolution. Note that increasing u linearly will not necessarily move the interpolated point a linear distance. The u units and the distance traveled along the spline do not have a linear relationship. If distances along the curve need to be measured, such as the case with our simulator, the spline can be arc-length

Figure 2.2: Illustration of undesired Hermite curve behaviors. Shown are a loop (a), cusp (b), and fold (c) that can arise from different lengths of tangent magnitudes [8].

parameterized either numerically or analytically. One simple method to do this is to create a table mapping u values to their interpolated point, with a running tally of the Euclidean distance between each entry in the table [17].

Tangent selection is important to create the desired shape of the curve. The magnitude of the tangent controls how “taut” the spline appears, with near-zero length magnitudes recreating polyline splines. Larger tangent magnitudes decrease the rate at which the curve deviates from the tangent vector. Large enough tangents will force the curve to create local cusps or loops to meet both ends. Figure 2.2 demonstrates these local deformations [10]. Our simulator uses dense point data, often with samples taken several seconds apart. We assume that local cusps and loops would introduce too high a curvature, but loops that form over several knots are a reflection of loops in the data.

Tangents generated automatically, such as in a Catmull-Rom spline (a cardinal Hermite spline) for example, preserve C^2 continuity but often do not preserve monotonicity even when the input data is monotonic. That is, if the data has input values that only increase in value, such as the timestamps in our simulator, spline interpolation with badly chosen tangents can introduce areas that decrease in value. Depending on the needs of the spline, breaking monotonicity can create

undesired behavior, such as introducing negative changes in values (like time) where none should exist. Methods to correct non-monotonic splines include the Fritsch-Carlson method, which computes knot tangents by using a weighted harmonic mean of slopes. A tangent y'_i can be found for a 2D spline with coordinates (x_i, y_i) where $i = 1.....n$ by using Equation 2.4 below [4].

$$h_i = x_{i+1} - x_i \quad (2.2)$$

$$d_i = \frac{y_{i+1} - y_i}{h_i} \quad (2.3)$$

$$y'_i = \begin{cases} 3(h_{i-1} - h_i) \left(\frac{2h_i + h_{i-1}}{d_{i-1}} + \frac{h_i + 2h_{i-1}}{d_i} \right)^{-1} & \text{if sign } d_{i-1} = \text{sign } d_i \\ 0 & \text{if sign } d_{i-1} \neq \text{sign } d_i \end{cases} \quad (2.4)$$

Orienting to Paths

An object moving along a spline curve can be oriented to the curve so that it points in the direction it is traveling. This object would be rotated every time it advances down the spline with quaternions. Quaternions are defined by an axis of rotation and an angle, and can rotate an object any amount, any number of

times. We use quaternions to orient our shark model to the heading of its travel. Below we show the definition of a quaternion (Equation 2.5) from an angle θ and an axis of rotation denoted as a vector (x, y, z) [17].

$$\text{quatRot}(\theta, (x, y, z)) = q = [\cos(\theta/2), \sin(\theta/2)(x, y, z)] \quad (2.5)$$

$$q_1 q_2 = [\theta_1, v_1][\theta_2, v_2] = [\theta_1 \cdot \theta_2 - v_1 \cdot v_2, \theta_1 v_2 + \theta_2 v_1 + v_1 \times v_2] \quad (2.6)$$

$$v' = \text{rot}_q(v) = q v q^{-1} \quad (2.7)$$

$$M_q = \begin{bmatrix} 1 - 2y^2 - 2z^2 & 2xy - 2z\theta & 2xz + 2y\theta \\ 2xy + 2z\theta & 1 - 2x^2 - 2z^2 & 2yz - 2x\theta \\ 2xy - 2y\theta & 2yz + 2x\theta & 1 - 2x^2 - 2y^2 \end{bmatrix} \quad (2.8)$$

Quaternions can be utilized in several ways. Quaternion multiplication applied to a vector rotates the vector (Equation 2.7). Multiple rotations can be

computed by multiplying the quaternions together before applying them to the vector (Equation 2.6). Quaternion multiplication is not commutative. Additionally, quaternions can be converted into rotation matrices for use on the display pipeline (Equation 2.8) [17].

The angle and axis to orient an object on a path can be derived by making a Frenet frame. A Frenet frame is an orthogonal coordinate system (u, v, w) that moves along a curve. The curve's derivatives define the frame. This is where $w = s'$, $u = s'' \times s'$, and $v = u \times w$ [17].

To rotate,

$$q = [\arccos(v \cdot x_{axis}), v \times x_{axis}]$$

2.1.2 Keyframe Animation

One of the oldest methods of traditional animation, and one still in use today by traditional animators and computer animators alike, is keyframe based pose-to-pose animation. An animator uses keyframe animation by identifying the key poses of a movement: poses which identify where the motion will change direction or form. These key poses are keyframes. Interpolating any finite number of frames between two keyframes results in inbetween frames. This way, a simulator does not need to recalculate the shark model's pose in every frame during the simulation, but rather in only the keyframes. Calculation effort can be optimized by computing keyframes every few frames instead of every frame drawn to the screen. The simulator only needs to generate enough keyframes to keep their interpolated results accurate [17].

2.1.3 Skeletal Hierarchical Models

Posing a computer modeled character to make keyframes requires an underlying system to manipulate the geometry of the character’s skin. Where a geometrical mesh represents a character’s skin, invisible line segments can represent the bones of a skeleton. Posing the skeleton then poses the skin local to the manipulated bone [17].

Bones are organized in a tree to make a hierarchy, with a single root bone having other bones as children. Those nodes can have child bones on their own to create chains of bones from the root to a leaf node. All bone chains of an object will meet together at the hierarchy’s root. Changing the orientation of one bone will change the location of all of its child bones, in the same way that rotating one’s shoulder joint relocates the elbow, wrist and fingers on that arm [17].

A bone has a head point and a tail point that defines its rest pose. Adding rotation and translation information in the form of a matrix defines the bone’s transformed pose. In our simulation, a stack of matrices suffices as the method we use to track bone transformations. Each bone’s transformed pose is the product of its transformation matrix and the matrices of all of its parent bones up to the root. We organize each bone from the root downwards onto the stack, which multiplies each bone matrix together as it traverses the hierarchy depth-first. At a leaf-bone returning to its parent, these multiplied matrices are popped off of the stack until another untraversed branch is found. In this way, each bone inherits all of the transformations of its parent bones combined without being affected by its children [17].

Each joint’s rotation is set by some external element, either human or algorithm. While translation joints exist in the form of prismatic joints, they occur

rarely in nature and the translation matrix usually represents only the length of the bone. Translating down the bone length moves the head of the bone to the tail of its parent, which properly connects the bones together in chains. The rotation and translation matrix multiply together to make what we will refer to as the bone matrix [17].

When the skeleton is properly posed, each vertex of the mesh skin is moved to match the pose. To do this, we define a relationship between the mesh vertices and the bones that manipulate them. Often, a vertex will be influenced by multiple bones to provide smooth deformation. Proper deformation of the mesh is a complex topic, but a simple method that gives somewhat respectable results is linear blend skinning [17].

$$v' = \sum_b w_b M_b B_b^{-1} v \quad (2.9)$$

Each vertex is associated with one bone and a weight w between zero and one that controls how much the bone influences that vertex. All of the vertex weights sum to one, which prevents unwanted scaling. To transform a vertex, it must be moved from its location in skin space to joint space, where the head of the bone is at the origin. This translation matrix is B^{-1} . Then the bone's matrix M is applied to the vertex, applying the skeletal hierarchy to it. Transforming vertex v then looks like Equation 2.9 [17].

2.2 Biology Background

With animation technique established, we can discuss the biological specifics of leopard sharks and related undulating fish.

2.2.1 Shark Anatomy

Many terms we use were derived from anatomical terms acquired from the background works presented in this section. Some are listed here.

- anterior - The end of the body with the shark head.
- posterior - The tail end of the body
- lateral - The sides of the body, left and right
- dorsal - The top (back) side of the body, opposite of the ventral side.
- ventral - The bottom facing side with the stomach.
- axial - The direction bisecting the shark into anterior and posterior halves.
- undulation - The wavelike movement of the fish tail to propel itself through water.

Sharks have fins that act as hydrofoils, controlling the shark's orientation while it swims. Figure [2.3](#) shows fin names and their position on a shark body [\[16\]](#).

- Pectoral fins - the two large fins that extend from the sides, near the gills. These control movement.

Figure 2.3: Fin names and location on model fish, dorsal and lateral views.

- Dorsal fin - The first vertical fin jutting out of the shark's back.
- Second dorsal fin - Smaller than the dorsal fin and found further down the shark's back.
- Pelvic fins - two small fins that are located on each side, not far from the anal fin.
- Anal fin - a small fin on the underside of the shark, at the start of the tail.
- Caudal fin - The large fin at the tail of the shark. Important for propulsion.

2.2.2 Types of Swimming Locomotion in Vertebrates

Sharks and many other fish can be categorized mostly in four types of swimming locomotion patterns, called kinematics. Each kinematic represents a method of undulating lateral propulsion along the spine of the fish [22].

Anguilliform motion uses the whole body for propulsion. The fish's length

makes up at least one wave along the body, and the amplitude of movement is quite large. The flexibility of these fishes' bodies allows them to turn easily, though their velocity is less than the kinematics below. Eels, lampreys and tadpoles exhibit this type of motion [22, 16].

Subcarangiform motion uses the last two thirds of the fish body for propulsion, otherwise it is similar to anguilliform. The body is stiffer, which allows for higher velocities, but less maneuverability. Trout, cod, and the leopard shark propel themselves with this method [22, 6].

Carangiform motion uses the last third of the fish body for propulsion, and it uses less than half a wavelength at any given time. Salmon and mackerel are examples of this type of motion [22].

Thunniform swimmers like tuna push themselves forward with their caudal fin. Holding the rest of their body stiff allows these fish to swim long-distances quickly [22].

2.2.3 Relationship of Amplitude and Frequency of Tail Beat

Fishes that swim with lateral displacement, such as subcarangiform swimming leopard sharks, can have their kinematics modeled with a Fourier series. Points down the length of the fish, from anterior to posterior, have their position in space defined as the axial coordinate u and the lateral coordinate v . The (u, v) coordinates with respect to time t are found as follows [20]:

$$u_p = C_{p0} + C_{p1}t + \sum_i \psi_{pi}(\cos(\omega_{pi}t + \delta_{pi})) \text{ axial, u (2.10)}$$

$$v_p = D_{p0} + D_{p1}t + \sum_i \varphi_{pi}(\cos(\omega_{pi}t + \eta_{pi})) \text{ lateral, v (2.11)}$$

C_{p0} and D_{p0} represent average initial values for the p th point. C_{p1} and D_{p1} represent slopes that move the point away from a horizontal line, from the previous point's location. The summation periodically animates the point, expressing the series using a cosine function with amplitude (ψ and φ) and phase (δ and η). The fish's tail beat period T represents one full oscillation of the tail, which leads to the fundamental frequency being defined as $\omega_1 = 2\pi/T$. A point will trace the path of a shallow figure eight as the series oscillates. The series is truncated at the fourth harmonic as the fifth and higher harmonics are not realistic from the shark's work and energy viewpoint. Inclusion or exclusion of a harmonic in this system is based on the particular point p being examined, as shown in Figure 2.4 [20].

Shark tail beat frequency was documented by Graham et al. by observing leopard sharks in a swim tunnel. Separating sharks of differing lengths into three groups based on body length, three power functions shown in Figure 2.5 illustrate the relationships between the shark's velocity, measured as body lengths per second, verses their tail beat frequency. On the whole, Leopard sharks beat between .75 Hz and 2.25 Hz. Larger sharks tend to beat their tail at a slower frequency than shorter sharks do [7, 1].

Figure 2.4: Theoretical amplitudes of harmonics at a constant velocity for 30 points, as discussed by Root et al.. Dark circles indicate the amplitude of a point at a given harmonic. Inclusion or exclusion of a harmonic depends on the point being analyzed [20].

Figure 2.5: Tail beat frequency verses velocity of *Triakis semifasciata*, separated into three groups based on body length. 30-60 cm length group - top line; 61-90cm group - middle line; 91-121cm group - bottom line. Circles, triangles and squares show individual sharks used to generate power functions [7].

Figure 2.6: Maximum lateral amplitude by % body length of the dorsal mid-line as a function of body position. Values represent average amplitude for leopard sharks swimming at one body length per second [6].

The amplitude derives from Donley et al.'s measurement of leopard shark's lateral displacement. Swimming speed is at one body length per second. The amplitude was measured by an average peak-to-peak amplitude divided by two. Figure 2.6 shows the increasing displacement, and thereby amplitude, towards the shark's posterior [6]. Fish velocity increases with tail beat amplitude, and the amplitude increases along with frequency until the shark reaches about five beats per second [1].

The propulsion's wavelength is typically shorter than the body length of the leopard shark, indicating that the shark uses more than one wave at a time during undulation [6].

2.2.4 Use of Fins in Swimming Motion

Many of the fins on a shark do not articulate for locomotion, such as the two dorsal fins and the anal fin. These fins prevent the shark from rolling as it moves

Figure 2.7: Definition of Pectoral fin planes α and β from lateral view (B) and ventral view (C) [25].

through the water.

The paired pectoral fins control the shark's pitch and balance the movements of the tail. The shark uses them to ascend and descend vertically. The two smaller pelvic fins direct water flow from the pectoral fins. The pelvic fins are held parallel to the pectoral fins. The shark also uses the pectoral fins as brakes [5].

The pectoral fins can be conceptually divided into two flat planes forming a concave downward shape, and the shark can modify the angle between them to maneuver upwards, downwards or to roll. Figure 2.7 defines the anterior plane α connecting the three points marked as 14, 15 and 17 on the shark. The posterior plane, β , is made up of the points 15, 16 and 17.

At low speeds the whole shark holds its pectoral fins at about 23 degrees anedral (ventrally) to its body. When rising, the shark flips the posterior plane β downwards to make a 200 degree angle interiorly, and a 35 anedral angle to the shark body. The angle of attack on the anterior plane α is 14 degrees.

When sinking, the shark flips β upwards and α downwards, making an angle of attack of -22 degrees. The planes' interior angle is 185 and the anedral angle to the body is at 5 degrees. Figure 2.8 visually shows this relationship on an

actual leopard shark. It also shows a function showing the rate of change in the fin's internal angle as the shark's pitch changes with vertical movement, which is discussed in the next section.

Figure 2.9 demonstrates these angles visually in an abstract diagram, showing the two pectoral fin planes, their angles between themselves, their camber, and their attack angle. These figures are in relation to the rest of the shark and the water flow [25].

2.2.5 Shark Pitch During Movement

During swimming, the leopard shark will adjust the pitch of its entire body with relation to its velocity and vertical direction. Figure 2.10 shows the relationship between the shark's velocity and its upward tilt. As the shark's swimming speed increases, the pitch decreases and becomes level. Average varies from 11 degrees at .5 lengths per second, to .6 degrees at 2 lengths per second [25].

At a velocity one length per second, the leopard sharks' pitch amount was recorded during rising, holding, and sinking behaviors. Figure 2.11 shows the individual values taken during those behaviors. Rising, the sharks average a pitch of positive 22 degrees. Holding, the sharks average positive 8.3 degrees. Finally during sinking, the sharks average negative 11 degrees [25].

Figure 2.8: Pectoral fin angle changes correspondingly with vertical movement [25].

Figure 2.9: The pectoral fins' two planes change configuration with chord, orientation, and camber [25].

Figure 2.10: Shark pitch correlates negatively to speed, leveling out as velocity increases. A linear relationship could be inferred [25].

Figure 2.11: Shark pitch changes correspondingly with vertical movement. Plot of body angle versus behavior during propulsion at 1.0 l/s , where l is total body length. The graph shows angles for 6 individual sharks. Circles show holding behavior, triangles show rising behavior and squares reflect sinking behavior [25].

Chapter 3

Related Works

Shark tracking has been previously done with large predators such as the great white shark and salmon shark [12], megamouth shark [15] and sand tiger shark [13]. Visualizations of these studies show a data path over months to years, and thousands of miles. Several maps are available online, by the Census of Marine Life [12], the Guy Harvey Foundation and Nova Southeastern University [13], and by Ocearch [14].

Often, shark tracking studies are done by satellite tagging on a long term basis, such as the pursuit of whale sharks by Eckert et al. [2, 11]. Visualization is done with maps that emphasize the path the shark has taken, rather than close-up properties of the shark on smaller time scales. The sharks are usually represented as a dot or a flat, static picture. The TOPP project imposes shark paths on maps featuring oceanic temperature, presence of chlorophyll, and other oceanic properties that may give insight to shark migratory habits [12].

Figure 3.1: Shark tracking visualization from TOPP. Map shows the northern Pacific ocean and the paths taken by three salmon sharks [12].

3.1 Animal Depictions

Simulations of animals have a long history in recreating primitive behaviors as seen in flock, herds and schools. Reynold’s landmark work creates “boid” flocks by assigning basic behavior rules to the individual members of the crowd. Individual flock members were programmed to follow close to other flock members in the same direction, while avoiding collisions. This simple rule created realistic flight paths taken by the individual flock members. Flock members also have a sense of perception, where they gather insight about the environment through simulated senses. They use their senses to make decisions based on their needs. The use of perception creates a more natural looking simulation, as it prevents omniscience [19].

Realistic depictions of animals were spearheaded by film efforts, where they became realistic enough to replace traditional film effects. *Jurassic Park* (1994)

intermixed computer generated dinosaurs with live actors and puppets to convince viewers that they were observing real dinosaurs. *Jumanji* (1995) renders animals that viewers would be familiar with, such as elephants and rhinoceros. Familiar animals would be judged more critically when their realism is evaluated [17].

Since then, popular media has been filled with depictions of computer generated animals, appearing on every spot on the continuum between realism and artistic expression. Often, creatures in entertainment are artist driven if they are in the foreground, reserving simulation for those that are part of a crowd or scenery. A human animator defines the poses an in-focus creature holds from moment to moment in a process called pose-to-pose keyframing. The exact desired motion that the animator wants is created in this process without imprecise emergent behavior that simulation yields. These works notably include *Finding Nemo* (2003), which depicts anthropomorphic sea life, like sharks, in a realistic ocean environment [17].

3.2 Sea Life Simulations

Creating animation of fish has been done previously in other research. Previous models, like by Stephens et al. and Terzopoulos and Tu, exist to model fish unscripted. The fish actions are defined by their own artificial intelligence. They engage in natural behaviors such as predator and obstacle avoidance, schooling, feeding, hunting and other behaviors [22].

The Terzopoulos animation incorporates a spring-like model for muscle dynamics, rather than a hierarchical model or skeletal rig that many animation applications use. Points on the fish skin are connected in a polygonal mesh. Spring

edges contract and expand, bending the fish model. Periodic bending allows the fish to animate swimming. Then, these fishes engage in their programmed behaviors [24].

A different fish swimming animation model was created by Tan et al. It uses an evolutionary intelligence algorithm to recreate accurate aquatic movement through water. It does so by simulating a physically accurate fluid environment and then selecting for the most energy efficient kinematics propelling a model animal. As more trials are run, the more lifelike the animation becomes. Model sea creatures like sea turtles, rays, eels and clown fish have their motion evolved to match their real life counterparts. The same process worked for a fictional alien creature [23].

3.3 Robotics

Robot fishes are relevant in that they take observed fish behavior and recreate it mechanically, accounting for all the physical restrictions that manifest in reality that do not necessarily occur in a computer animation. The construction of robot fishes, such as the one by Nilas, serves as inspiration for the design of our method. Nilas creates a four segmented model fish that can swim in a multitude of kinematics in a straight line. When an obstacle is detected with infrared, the robot will autonomously swim around it by turning [16].

Chapter 4

Methodology

We are interested in a simulation that is semi-scripted, that includes a set of points to visit, but without needing any further input from a user about how these points are visited. We wish to do this without needing to run multiple trials as the set of input data points is potentially very large. This simulation will derive its technique from these several ideas: existing knowledge of motion kinematics from real leopard sharks and other fish, animating along Hermite splines, keyframe based animation, and hierarchical modeling.

Modeling the shark's path through the ocean combines the techniques we discussed in the previous chapters. The shark skin is modeled with the geometry of the shark and with a skeletal rig that allows it to be posed. The path the shark travels down, defined by the imported raw data, is modeled in a plausible manner for shark travel. This is all displayed to a computer screen with a procedural pose-to-pose keyframe based animation system.

We will briefly describe the simulation's overall process. Each of these components will be described in full later in this chapter.

Figure 4.1: Overview of simulation pipeline during runtime. The shark model’s movements ultimately derive from the path defined by the input data and the running time since the simulation started.

We begin with a data file representing a shark’s timestamped location path. While loading the data from the file, the simulator builds the splines used during runtime. There are two splines: one that maps the running time to the shark’s location, and one to represent the shark’s path through space. The time-space curve and the space curve’s tangents are computed. The space curve is also arc parameterized. Computing small batches of the splines can be done during runtime instead of calculating everything in loading time, but this would sacrifice the ability to zoom out and view the shark’s future path.

The simulation then loads the shark mesh and rig from a file. It builds a model representation of the shark rig in memory by mapping vertices to bones with their linear vertex weights. All the rigging work is done outside of the simulation.

Once the input files have been fully loaded and processed, the simulation enters its main graphical animation loop. The loop continues until the program is terminated. A clock keeps track of the total running time of the simulator, and becomes the single input into the simulation’s runtime pipeline, shown in [Figure 4.1](#).

The runtime pipeline consists of these five components:

1. The time-space path converts the clock into a location marker. The marker is the current knot's index and its progress toward the next knot, a value between 0 and 1.
2. The space path maps the location marker to the shark model's coordinates in three dimensional space. Velocity is derived by comparing this timestep location to a future one. The turning angle is derived by looking ahead at the path curvature.
3. Every few timesteps, the locomotion simulator takes the velocity and the turning angle and poses the shark rig according to biological kinematic principles. The simulation snapshots a pose and turns it into a keyframe.
4. The keyframe queue stores three keyframes, where the first two are interpolated between.
5. The display draws the current inbetweened frame to the screen using OpenGL.

4.1 The Data and the Simulator

Our simulator was built and tested for the data collected from real fish tracked. Here, we describe the circumstances of the data that was returned. We also describe the way that this simulator was built.

4.1.1 Nature of the Data

Raw data is stored in computer files, and datasets exist in a variety of file formats. Typically, short term tracking data are denser than satellite tracking data, measured on a scale of meters. The data do not contain sea depth information.

Figure 4.2: Top down view of AUV data plotted on a map. The pink line is the estimated shark position [3].

The path true to the data will lie in a plane. This makes some fin movements slightly unrealistic, as the tagged shark probably did not move in an exact plane. Our simulation adds a small amount of vertical variant for all points, moving them up or down by a value under one meter. This is done by adding the result of a sine function on successive integers to the y coordinate. This makes it easier for the user to see when points overlap and which direction the shark will take upon reaching those points. Spline tangents were calculated without this small vertical variant, leading the turns created to remain lateral.

Data points are typically on a magnitude of seconds apart, and not taken regularly. Figure 4.2 shows a top down path an AUV has generated. Appendix A is an example of what AUV data may look like.

The tracked data received are also not necessarily precise. Trackers like AUVs can cluster possible shark locations per sample and then record that cluster's centroid as each location point with associated timestamp. The data that robot records is an estimation with a degree of error and fewer significant figures. Data

points often lie on top of one another. In addition, sometimes a tracking system loses the location of the tag, introducing long gaps in the data [3].

4.1.2 Nature of the Simulator

The simulator is built using C++ with OpenGL. Zlib (from www.zlib.net) is used for uncompressing MATLAB files if they are encountered. The simulation's origin is set to the first point in the dataset. Rigging of the shark model was completed in Blender, and then exported to a custom file format for use in the simulator. Skinning of the shark model was done with the linear blend skinning algorithm in the simulator.

With the amount of knots that can appear in a single data set, coloring the spline becomes a useful indication of the shark's true path. This is important considering that the shark often revisits places and that organizing the points according to which ones the shark will travel to next helps to clarify the shark's next goal point.

View frustum culling is necessary to display the point data at any respectable real-time frame rate. View frustum culling only draws points located in the camera's field of view, the view frustum. By doing so, the processor has far fewer calculations to output, improving framerate [17].

Knots the shark has already passed through are colored red. The most upcoming points are white, and further away points fade to yellow and then green. Black points represent points far in the future. Figure 4.3 indicates the visual representation of the path in the simulator.

Figure 4.3: Path displayed on the screen. The path behind the shark is colored red.

4.2 Procedural Pathing

Datasets contain timestamped location points. The path connecting these points has to be inferred. Our method involves two splines, one to represent the shark's physical path through space, and one to interpolate the timestamps.

4.2.1 Time-Space Path Curvature

The data path contains timestamps that need to be interpolated. This method uses a Hermite curve that maps timestamps to knot indexes. Interpolating the time since the simulation started will yield the shark's location between knots. That value is immediately input into the space curve to get the shark's true position in space.

A time-space spline that is not monotonically increasing will cause slowdowns, stops, and even reversal of the shark movement. Backwards movement in a shark would necessitate animating the shark turning around, and this animation would have to keep up with the rate of the time-spline's interpolation. A different solu-

tion prevents the time spline from having reversals, which would involve choosing this path's tangents carefully. We will choose tangents to preserve monotonicity.

For the time-space curve, the tangents are chosen using the Fritsch-Carlson method as shown in Equation (2.4). Since the coordinates for the time-space curve are the timestamp and knot index, a 2D solution suffices. Both coordinates of this tangent are monotonically increasing and both can be found using the method [4].

4.2.2 Space Path Curvature

The space curve maps the path traveled in three dimensional space. The input to finding the shark's location is the location marker returned from the time-space curve. We use a Hermite spline, which allows tangents to be defined independently of the knots they belong to, and then stored in program memory. We shape our curves by defining the tangent's direction and magnitude. The space curve's tangents are chosen carefully to prevent local folds, loops and cusps. Figure 4.4 shows a simple case.

The turn angle is measured as the dot product from the vectors between three knots. One vector is made from a knot p and it's previous knot (p_{before} , to make vector $\vec{A}$) and the second vector is made between p and the next knot (p_{after} to make vector $\vec{B}$). In a low turning angle, the tangent $\vec{t}_p$ is small in magnitude. Its direction is the same as the vector connecting p_{before} and p_{after} .

If the turn angle is great enough, a new tangent is created instead. The minimum angle is arbitrarily chosen in this simulation to be a dot product of 0.7. Figure 4.5 shows this case. This tangent is found by creating a new vector $\vec{C}$ derived from $\vec{A} \times \vec{B}$, and then creating $\vec{D}$ which derives from $\vec{A} \times \vec{C}$. The tangent

Figure 4.4: Tangent generation in low turning angles. Blue lines are vectors created to measure the turning angle. The resulting tangent is orange and the spline is in yellow.

lies parallel to $\vec{D}$, and points toward the closest route to p_{after} , such that $\vec{D} \cdot \vec{B}$ is positive. This tangent will have a large magnitude, preventing the turn from becoming too sharp [10].

The 180 degree u-turn case is discontinuous. In this case, p_{after} is re-chosen from the next point forwards, until $\vec{A}$ and $\vec{B}$ are no longer parallel.

After the tangent vector is found, it is normalized so that a separate magnitude scalar can be multiplied onto it as shown in Equation 4.1. The tangent increases in magnitude the sharper the turn angle is. The sharpest turns are widened by larger tangents, allowing the shark model enough room to turn. Yet, the magnitude is less important for smaller turn angles. It becomes less important to keep turns wide and more important to simplify the path. Using large tangents when not needed increases the chances of introducing sharp turns on adjacent knots.

Figure 4.5: Tangent generation in high turning angles. $\vec{C}$ derives from $\vec{A} \times \vec{B}$ and $\vec{D}$ derives from $\vec{A} \times \vec{C}$. $\vec{t}_p$ is parallel to $\vec{D}$.

$$s' = s_{min} + (s_{max} - s_{min}) \frac{(\pi - \theta)^2}{\pi} \quad (4.1)$$

The maximum size magnitude takes up to half of the distance between the shorter of the turn angle vectors, and the minimum size is selected at a small, positive value. The maximum size prevents the curve from introducing sharp turns elsewhere on the path. The scalar is chosen as an interpolation between the min and max based on the turning angle.

4.2.3 Arc Length Parameterization

Arc length parameterizing the spline path allows the shark to look down the path for curvature information relative to itself. This allows the simulation to derive turning angle, future velocity, and global orientation, which we describe in Section 4.3.2. These values are analyzed by the shark locomotion simulator to create keyframes that respond to the path.

To define the path in space concretely, the spline path is sampled and recorded

in a table. The Euclidean distance from a sample point and the previous sample point are recorded. One hundred samples between knots serves adequately if the knots are 1-2 meters apart. Any further discrepancy between sample points can be linearly interpolated between.

4.2.4 Analyzing Path

We determine the motion for the shark model procedurally, based entirely on the path that it follows. The shark’s velocity and its turning angles are the two inputs to the shark model required to generate the model animation. Both of these values are derived from the space path the shark model follows.

The shark’s global orientation is decided by a modified Frenet frame. We measure the angle across two vectors. One vector is the difference between the root of the model at point r_i and a point ahead on the path, r_{i+1} . The distance from r_{i+1} to the root along the path is equal to distance between the shark’s root and nose. The first vector is defined as $r_{i+1} - r_i$. The other vector is the global z axis, where we have defined the y axis to represent upwards. The dot product of these two vectors gives the angle of rotation globally when rotated about the global y axis. This angle and axis can be saved into a quaternion and interpolated during simulation.

We also derive the shark’s turning angle based on the spline path. It is the difference between the shark’s current heading and its new heading. This turning angle is defined by a different two vectors. Two points are chosen, one point where the shark’s nose should be currently, a_1 , and another point a small distance (about the length between two spine bones) ahead of the first point, a_2 . Figure 4.6 shows their relationship. Using the root as the vertex, the turning

Figure 4.6: Shark turning angle derived from looking ahead. The turning angle θ is the angle between points a_1 and a_2 through the shark's root.

angle is derived through a two argument arctangent function.

We use the shark velocity to derive the shark locomotion. For accuracy, it is important that the shark model and data points are scaled to real-world proportions, where the shark model should match the size range being used to calculate the tail beat frequency and amplitude discussed in Section 4.3.2. A look-ahead simulation point is generated in the simulation a half second in front of the shark model's location. Basing the shark locomotion off of the look-ahead simulation point gives the shark a period of anticipation of its future action. The anticipation will make the shark appear as though it is purposefully following the path, rather than the path dragging a model shark along it unwillingly. The velocity is calculated on that lookahead-point by measuring its distance traveled over the time elapsed between steps in the simulation. This method does not prevent the spline from creating illegal velocities that exceed measured speeds in leopard sharks and seen in Figure 2.5, but such illegal velocities can be flagged [7].

We have gathered turn angle and velocity from the path we have created. We only needed the raw data points and simulation clock as input. We can now use

Figure 4.7: Shark polygonal model and rig from dorsal and lateral views. Indicated in blue are the model's bones that make up the hierarchical model. The bone visualizations are largest at their head and taper down to their tail, showing the direction of the hierarchy.

these values to animate the shark model.

4.3 Shark Modeling

The skeletal rig of the shark is posed according to the procedural rules outlined in following sections, built off of the rules for undulation, pitch and fin movement described in Chapter 2. Once posed, the model can be copied over to an animation keyframe, which becomes part of the animation and drawing system.

The shark model's spine is built down the x axis, where the head is closer to zero. This makes the spine bones easy to compare to unit vector $[1,0,0]$ to calculate the amount of rotation in degrees. This is found by first computing the difference vector between the head and tail points, and secondly by computing the dot product between this difference vector and the unit vector $[1,0,0]$. The rig has a total of 10 spine bones for simulating undulation, at least two bones for

each fin, with the pectoral fins having four in total, counting bones that connect the fins to the spine. Figure 4.7 is a picture of the model and rig.

4.3.1 Hierarchy

Deciding where to place the root of the hierarchical model affects the visual appearance of the shark’s locomotion. Placing it at the tip of the nose, for example, makes the swimming motion look more anguilliform rather than subcarangiform, and locks the nose in place. A more realistic location is close to the dorsal fin and the pectoral fins. This allows the head of the shark to swing around freely, allows the front of the body to better approximate subcarangiform motion, and allows the pectoral fins to point at the desired direction.

Following that, the shark’s locomotion is defined by the lateral movement of the spine, which stretches in two directions toward the head and tail. The pectoral, dorsal, lower caudal and anal fins make up branched leaves on this hierarchical tree structure. The pectoral fins have three bones to create the two planes α and β that control the shark’s pitch.

4.3.2 Locomotion

Shark locomotion is built upon several inputs: the turning angle from the curve path, the shark’s velocity and the length, in meters, of the shark. The velocity and shark length are the inputs to several functions that will eventually rotate each of the bones of the spine. Figure 4.8 displays a flow chart showing this relationship. After the straight line swimming angles are determined, the turning angle can be distributed over the spine bones to turn the shark. Our simulation defines our shark model to be 1.0 meters long, through this number

Figure 4.8: Flow chart for creating swimming motion in a straight line. The shark's velocity from the path and its length are used to rotate the spine bones periodically to undulate over time. The amplitude and frequency are placed in a wave function to generate rotation angles, and a phase offset is created to counteract discontinuous movement.

can be changed for other simulations without consequence.

By adapting the Fourier series from Equations 2.10 and 2.11 onto the shark model spine bones, the shark model will simulate undulation [20]. Since the shark is hierarchically modeled, the C_{p0} , C_{p1} , D_{p0} , and D_{p1} variables from those equations that were used to locate the point along the shark body are unnecessary and can be dropped, leaving the summation of the series to represent only the spine bone's oscillation.

$$u_p = \sum_i \psi_{pi}(\cos(\omega_{pi}t + \delta_{pi})) \quad (4.2)$$

$$v_p = \sum_i \varphi_{pi}(\cos(\omega_{pi}t + \eta_{pi})) \quad (4.3)$$

Inclusion and exclusion of harmonics on spine bones differs on the position

Shark Size	ω_{min}	ω_{max}	v_{min}	v_{max}
30-60cm	1.25	2.25	20	100
61-90cm	1.00	1.75	20	100
91-121cm	.75	1.20	20	100

Table 4.1: Variables to insert into Equation 4.4 depending on shark size (cm). ω values measured in Hz. Velocity v values measured in cm/s^{-1} [7].

of the spine bone on the body. The fundamental frequency is computed for the entire length of the shark, though dampened considerably in amplitude for the front third of the body to maintain a subcarangiform kinematic pattern. In the last third of the shark, harmonics above the fundamental are added one by one onto the computation.

The velocity and length of the shark affects the frequency (ω) of the tail beat. The frequency range comes from the power functions from Figure 2.6, corresponding to the length of the model shark [7].

$$\omega = \omega_{min} + (\omega_{max} - \omega_{min}) * \frac{v - v_{min}}{v_{max} - v_{min}} \quad (4.4)$$

Variable v in Equation 4.4 refers to the velocity of shark travel, measured in cm/s^{-1} rather than in body length like the other equations in this paper. The min and max versions of ω create the desired range of frequency (in Hertz) available for the size of shark.

The frequency is not expected to remain constant over the course of the simulation because the shark's velocity will not stay constant. Notice that changing the frequency of a wave while it is oscillating will not preserve the illusion of con-

tinuous movement. Shark oscillation will jump to a different angle every time the frequency changes, giving the shark model a jittery, rapid, unrealistic animation. The phase variables δ and η can be recalculated to counteract this phenomenon. They ensure the same angle is reached at the same t value for both functions before and after changing the frequency. The following equations represent the angle to be found by the cosine part of Equation 4.2 and they are the same for η .

$$\omega_{old}t + \delta_{old} = \omega_{new}t + \delta_{new} \quad (4.5)$$

Solving algebraically for δ_{new} leads to:

$$\delta = \omega_{old}t + \delta_{old} - \omega_{new}t \quad (4.6)$$

For amplitude variables ψ and φ , amplitude increases proportionally with frequency up to a maximum of one fifth of body length, at five tail beats per second [1]. The below equation also solves for ψ , except that the axial amplitude will be a small fraction of the lateral amplitude. Overall movement should be in a vertically flat figure 8 or oval shape.

$$\varphi_p = \varphi_{min} + (\varphi_{max} - \varphi_{min}) * \frac{\omega - \omega_{min}}{\omega_{max} - \omega_{min}} \quad (4.7)$$

Amplitude is reduced in the front third of the shark model to recreate subcarangiform motion. φ_{pi} for each harmonic is found by dividing the fundamental frequency's amplitude by the harmonic number [20].

Equations 4.2 and 4.3 - after plugging in the results of Equation 4.4, Equation 4.6 and Equation 4.7 - create the coordinates u and v . These coordinates create the shark model undulation for swimming in a straight line. The vectors are normalized and then dot product is taken with the x axis. This finds the amount of rotation off of that axis in degrees.

To make the shark bend for turning, a desired turning angle is passed in and added into the angle derived from v_p . The turning angle commands the shark to bend in a particular lateral direction relative to the shark model's local coordinate system, so that it appears to make a turn. When the turning angle is zero, the shark will swim in a straight line. Negative and positive values turn the shark left and right. Note that this process does not consider shark muscle bending limitations.

Two rotations are then performed on the spine bone, one axial and one lateral, with the amount of the final degrees found. With the amount of rotation known, the quaternion and rotation matrix information are created and saved onto the bone.

4.3.3 Fin Movement

Three cases are determined for the shark's movement of its pectoral fins. Rising, sinking, and level swimming animations incorporate the principles from Section 2.2.4. Typical values are used for rising, sinking and holding. Interpolation of these values allow the shark model to smoothly change fin position as

Behavior	α bone	center bone	β bone
Rising	5	-35	-9
Holding	-6	-23	-1
Sinking	-12	-5	10

Table 4.2: Pectoral fin configurations for rising and sinking, in degrees hedral to the shark body.

velocity and pitch change.

Changing pectoral fin position requires moving the three bones defined in the fins' rig, in order to simulate the behavior of the two planes α and β . The bone anterior will be referred to as the α bone. The bone posterior is named the β bone. The center bone separates the α and β planes.

4.3.4 Shark Pitch

The pitch angle of the shark can be adjusted by rotating the root bone on the skeleton around the shark's local z axis (recall that the shark model is aligned down the x axis). There are two rotations based on the facts from Section 2.2.5. One is based on horizontal velocity, interpolated from 11 degrees at .5 lengths per second, to .6 degrees at 2 lengths per second.

The other rotation is based on the vertical velocity. The sign of the y component of the velocity defines three states for vertical movement: rising, holding and sinking. The shark model interpolates between three values as it changes between the three states. Rising, the shark will rotate 22 degrees upward. Holding, the shark rotates to positive 8.3 degrees. Sinking, the shark rotates to negative 11 degrees.

From the path's velocity and turn angle, we have created a pose for the shark model with undulation, fin movement, and pitch changes. We can now create a

Figure 4.9: A sample of shark model keyframes. Keyframes are computed every few timesteps and inserted into a queue. The first two keyframes in the queue are interpolated on the display.

keyframe from this pose.

4.4 Procedural Keyframes

The shark model adjusts its pose procedurally in response to the path. The shark locomotion and fin movement determines what the finished pose of the skeleton is. After the skeleton poses the position of all of the fins and the spine, it then becomes a keyframe in an animated environment. Because each pose is the end effect of the shark's locomotion, fin movement and turning angle, keyframes are unlikely to be similar. Each keyframe is inbetweened by the computer to the next keyframe.

A keyframe copies the end pose information from the skeleton, allowing the skeleton to prepare the next pose while the keyframe is in use. The newest keyframe created is inserted as the third element into a queue. Some sample keyframes appear in [Figure 4.9](#).

The first and second elements in the keyframe queue are being interpolated as the simulation is running. When the interpolation between them finishes, the front of the queue is destroyed, and the simulation creates the next keyframe to insert. Linear or spherical linear interpolation between these frames are sufficient, as the keyframes tend to be close to each other in time, and the shark simulation does not demand the skeleton to make large steps between keyframes.

With interpolated keyframes being displayed to the screen, we have a complete running animation of the shark swimming down the path. Once the keyframe is finished interpolating, the next keyframe enters the queue. The simulation poses the shark model again and keeps repeating this process until the animation terminates.

Chapter 5

Results

We have created a computer generated animation of leopard shark motion using only timestamped location data as animation input. Location data is taken from a file with horizontal (no depth) coordinates in meters. The simulation's origin is set to the first point recorded.

Location data was used to create a space curve spline that the shark traverses across over time. The timestamps on each data point were used to create a time-space curve to correlate the running time of the simulator to the position of the shark model on the space curve. Both curves were created using Hermite splines. The space curve's tangents were modified to prevent overly sharp folds and cusps that would force the model shark to make an unrealistic turn, by using small tangents for low turn angles and larger, and perpendicular tangents for high turn angles.

The shark itself is a hierarchically modeled and skinned polygonal mesh. It's motions are based on scientific studies with real leopard sharks. Like real sharks, the shark model will angle its fins and pitch tilt its entire body according to any

Figure 5.1: Screenshots of simulation during play through. **A:** A zoomed out view of the spline path showing traversed path in red. **B:** The shark model approaching a 180 degree u-turn with tangent size increased to aid turning. **C:** The shark skin closeup with pectoral fins angled downward. **D:** The shark model skeleton with bones drawn as triangles.

rising and sinking behaviors, as well as the its velocity. Kinematics for lateral and axial undulation comprise a Fourier series with up to four harmonics. The undulation's amplitude and frequency derives from the model's velocity along the spline path and adjusts accordingly to that.

Several methods of distributing amplitude and harmonics of the shark undulation along the body were recorded and presented as a survey to judge human perception of realism in the simulation. The method we present in this paper is rated more realistic than not. In addition, the simulation runs at smooth interactive frame rates on desktop computers.

The purpose of the space-curve spline tangent algorithm was to prevent sharp turns that could not be traversed by the shark model without significant self

Figure 5.2: Comparison of Catmull-Rom spline (A) and our method (B). The Catmull-Rom spline can generate oversized tangents that can force the point before it to become a hairpin turn.

intersection. We compare our variant of Hermite spline with the Catmull-Rom spline, a similar simple automatic tangent generation method. Figure 5.2 shows a problem spot in Catmull-Rom that would be smoothed over by our Hermite spline.

The time-space curve for monotonicity were constrained by forcing the program to crash when time values stopped or reversed.

5.1 Realism of Shark Motion

It is difficult to evaluate the shark model's behavior without resorting to heuristics, since ultimately the purpose of the simulator is to produce a series of still frames to convince a viewer of movement. To measure the perception of realism, we created a survey.

Survey takers online at Amazon Mechanical Turk (<https://www.mturk.com>) watched a ten second video of the simulator, and three other 10 second videos of the simulator with minor adjustments (shown below) to the algorithm. An additional final video is made with our method replaced with a rotoscoped leopard shark filmed from a dorsal view. Rotoscoping is the act of copying animation

frames from live animals on film. It was animated using a program called Cal-Shark, developed by Greg Ostrowki. These clips were presented to survey takers labeled A, B, C, D and E without context. The clips in actuality have these changes:

- A** Our method, except four harmonics on the whole shark body in Equation 4.3.
- B** The method we describe in this paper as is, without adjustments. This is the gradual increase of the number of harmonics toward the tail.
- C** Like clip A, with amplitude not reduced across the harmonics to exaggerate movement.
- D** Our method, but only using the fundamental frequency in Equation 4.3.
- E** Rotoscoped live leopard shark used in simulator instead of our procedural keyframed model.

Multiple videos were created to make the survey viewers think more critically about the animation clips through comparing and contrasting them. Without these differences the viewers may have focused on aspects of the clips besides the motion. Variants of Equation 4.3 were chosen for the first four clips because this was an area where we did not have information specifically for leopard sharks versus other fish. This makes the first four videos very similar in appearance. By amount of changes, Clips A and D have the least amount of changes from the original, B. Clip C has one more change, the amplitude, that makes it further different than B. The rotoscoped clip E has leopard shark motion swimming straight. But, it does not respond well to the randomly generated path's requirement for changes in velocity, so it appears exaggerated. This is because a small,

Figure 5.3: Bar graph of surveyed opinion on simulation realism. 27 random viewers watched 5 film clips of the simulation and decided that the method described in this paper was most realistic.

finite amount of frames were rotoscoped. Clip E's presence is to be significantly different than the other four clips so that the viewers have a greater scope of motion styles for comparison.

The surveyed were then asked to rate how realistic the shark motion in each clip appeared, on a scale from 1 ("completely broken") to 10 ("lifelike"), without regard to the water, textures, or any other effects. In total, 30 people were surveyed, with 3 results thrown out for completing the survey in under 50 seconds, which is the time it takes to play all 5 clips. Figure 5.3 displays the average results for the remaining 27.

Our method, where we increase the number of harmonics used toward the tail of the shark, was rated the most realistic, 6.78 out of 10. Using four harmonics along the whole length of the shark was the next most realistic. The rotoscoped shark was rated the least realistic. Appendix B shows the survey question and the full results.

5.2 Performance

This method runs fast enough for real-time user interactivity. Performance was tested on a dataset of 18,900 points. The performance bottleneck is in view frustum culling those points and their curves. On a AMD Athlon II X4 640 Processor, 7.6 GiB memory, ATI Radeon HD 4250 GPU, the single threaded simulation reaches 56.3 frames per second average over 18 seconds of simulation. Shortening the culling process to consider only 600 points, the simulation runs at 68.5 frames/s average over 18 seconds of animation. This frame rate allows a user to rotate a camera around the shark, zoom in and out, and observe surrounding points as they wish. The program requires a load time (under 10 seconds) to read in the point data, build tangents, and arc-parameterize the curve path.

5.3 Future Work

Some improvements to the system can arise with more data. With data representing GPS coordinates rather than local distance, a map could be superimposed on the simulation. Way point markers illustrate the shark's relative position. Acquiring depth (y-axis) information, which is lacking in each dataset tested, can illustrate the shark's precise location clearer.

Alternatively, trackers like the AUV can be modeled and inserted into the simulation with its own path, if the tracker was in pursuit of the shark. This would illustrate the relationship, if any, between the tracker and the shark itself.

The caudal fin could be broken up, examined, and animated as a collection of smaller planes, such as we did with the pectoral fins and its two planes.

Shark behavioral models were not considered during implementation. Possible

behaviors include fleeing and hunting. Such consideration could enlighten us to the shark’s goals during simulation and may be incorporated into the animation.

Our method does not model water physics. As such, we have not incorporated a model to consider forces on different parts of the shark body.

An alternative method we tried involved rotoscoping top-down footage live leopard sharks. For this to work, we needed a human animator to create loops of every possible motion the shark model could use. While running, the simulator chose between the stored animation loops depending on the turn angle and velocity from the path. Leopard sharks, even in captivity, don’t generate a good loopable swimming animation as they turn whenever they want to. The required human involvement for this method was significantly higher than the method we wrote here. The path required far too many varieties of loops to make this method very flexible and feasible.

Our application is one of many methods available to animate a leopard shark. There are alternatives to the specific algorithms we used. There are other skinning algorithms besides linear blend skinning. Fritsch-Carlson is only one method of many to create a monotonically increasing spline. Of every variety of interpolating splines, we tried polyline curves, Catmull-Rom curves, Hermite curves, and our modified Hermite curve tangent method. Our shark model has 10 spine bones, and it is possible the simulation could be different if there were more. We did not focus on rendering the shark on the display, as we were more concerned with the motion kinematics. There could be richer shaders. Each component of our method should still produce good results if replaced with a different one. We have used some of the simplest methods to showcase the minimum requirements to create this simulator. With more advanced techniques, it should be simple to create higher quality simulations.

5.3.1 Limitations

Time-space curve analysis to prevent the shark model performing illegal velocities needs to be done. While the shark will stay true to the data presented, the data can have gaps caused by loss of contact between the tracker and the tag on the live shark. The simulation attempts to simulate over gaps, but the loss of information introduces places where the shark model may not travel at a reasonable speed. The simulation does not monitor velocity, nor does it prevent illegal velocities from happening.

Future work can also entail examining the velocity at which the shark model enters and exits turns. This should yield information that we could use to modify the turn animations to account for very fast turns. This subject was described by Kim et al. on work with artificial fish and Porter et al. on leopard sharks [9, 18]. Considering the maximum and minimum muscle length when turning can constrain the shark turning radius more realistically.

Similar spline velocity analysis is also needed to account for rapid starting, on boundaries where slow speed suddenly jumps to high speed. Shark bending models need to be created for this situation. These bending models can be incorporated into the rest of the simulation for calculating turning angle. This would aid with determining how sharp the shark should actually be able to turn. Turning of the shark model does not consider maximum or minimum muscle constraints.

Where *Triakis semifasciata* information was unavailable, studies conducted on related undulating vertebrates, primarily lamprey [20] and dace [1], were substituted. These studies claimed that their results are applicable for fish in general.

Rapid diving and rising behaviors were not accounted for. The data that

would represent accurate vertical movement was unavailable. Data points were artificially adjusted vertically up or down by a value under one meter. This is to aid visual discrimination between overlapping points.

5.3.2 Conclusion

We have presented a method of visualizing tracked sharks. Our simulation approximates the shark’s actions through animation, as accurately as possible to the path the shark has taken through the water. It runs at a real-time, interactive framerate of over 55 frames per second. That allows users to manipulate the scene camera and look around the shark as they wish. We have taken care to minimize the amount of human effort to simulate each set of data points. All our method requires are the output from the trackers, a rigged skeletal mesh of a shark, and the simulator.

We have created a satisfactory leap into alternative shark tracking visualization methods. Our simulation is a step forward for smaller scale shark tracking missions that differ from the continental scale tracking missions that exist today. This is a stepping stone for even more robust simulations in the future. We wish to inspire others to create different methods of visualizing scientific data. With this paper, we wish to further education, awareness, and interest of our ocean’s creatures and the ecosystem we share with them.

Appendix A

Sample Shark Data

Point No.	Location (x,y,z)	Time (seconds)
1	(0,0,0)	0.0
2	(0.927141, 0, 2.21834)	3.137
3	(2.78144, 0, 3.32751)	3.923
4	(3.70859, 0, 5.54584)	5.598
5	(5.56288, 0, 6.65501)	6.254

Table A.1: Example internal tracked data. Tracker data is unpacked and converted into a form such as the example points in this table. First point has been set to origin.

Appendix B

Survey Result Table

Survey Title: “SURVEY: Realism in animation”

Description: Watch five 10-second video clips of a computer animated shark swimming, and give feedback on how realistic they look.

Instructions: The following video contains five 10 second clips of a shark swimming, labeled A, B, C, D and E. Watch the swimming motion and decide how realistic each clip looks to you. Focus on the motion instead of the lighting, texture, water, lines, and other effects. Afterward, rate the realism of each clip on a scale from 1 - 10. A rating of 10 is lifelike. A rating of one is completely broken. Five is semi-realistic.

Worker No.	Seconds Worked	Clip A	Clip B	Clip C	Clip D	Clip E
1	123	6	2	2	1	1
2	196	3	4	5	5	7
3	241	6	7	2	5	8
4	140	7	5	3	2	2
5	89	10	8	10	9	5
6	106	3	5	2	2	3
7	464	8	3	5	7	2
8	89	4	8	4	6	3
9	248	4	8	4	4	1
10	841	7	6	7	4	5
11	141	5	2	7	1	2
12	268	10	7	5	5	4
13	180	6	8	5	5	4
14	281	5	8	3	2	2
15	395	10	7	5	5	4
16	69	7	8	2	3	2
17	38	8	6	8	9	2
18	174	3	7	5	9	2
19	157	7	5	10	8	1
20	87	4	7	6	6	5
21	197	6	5	4	6	5
22	476	5	8	6	4	7
23	25	3	4	7	5	6
24	943	7	7	6	6	4
25	96	8	6	7	7	4
26	152	4	8	4	4	8
27	391	4	7	3	2	3
28	44	5	1	2	7	8
29	136	3	5	3	6	3
30	73	5	9	4	5	2

Table B.1: Survey results for 30 clip viewers. Listed in seconds is how long viewers took to complete the survey. The successive columns describe their rating 1-10 for each clip.

Bibliography

- [1] R. Bainbridge. The speed of swimming of fish as related to size and to the frequency and amplitude of the tail beat. *Journal of Experimental Biology*, 35:109–133, June 1957.
- [2] S. A. Eckert and B. S. Stewart. Telemetry and satellite tracking of whale sharks, *Rhincodon typus*, in the sea of Cortez, Mexico, and the North Pacific Ocean. *Environmental Biology of Fishes*, 60(1-3):299–308, 2001.
- [3] C. Forney, E. Manii, M. Farris, M. Moline, C. Lowe, and C. Clark. Tracking of a tagged leopard shark with an AUV: Sensor calibration and state estimation. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 5315–5321, 2012.
- [4] F. Fritsch and R. Carlson. Monotone piecewise cubic interpolation. *SIAM Journal on Numerical Analysis*, 17(2):238–246, 1980.
- [5] J. E. Harris. The role of the fins in the equilibrium of the swimming fish. In *Journal of Experimental Biology*, volume 15, pages 32–47, March 1937.
- [6] Jeanine M. Donley and Robert E. Shadwick. Steady swimming muscle dynamics in the leopard shark *Triakis semifasciata*. *Journal of Experimental Biology*, 206:1117–1126, December 2002.

- [7] Jeffrey B. Graham, Heidi Dewar, N. Lai, William R. Lowell, and Steve M. Arce. Aspects of shark swimming performance determined using a large water tunnel. *Journal of Experimental Biology*, 151:175–192, March 1990.
- [8] Jun-Hai Yong and Fuhua (Frank) Cheng. Geometric Hermite curves with minimum strain energy. In *Computer Aided Geometric Design*, volume 21, pages 281–301, 2004.
- [9] C. Kim, M. Shin, S. Jeong, I. Kang, E. Kim, and B. Kim. Real-time motion simulation of artificial fish for virtual marine world. In *Advanced Language Processing and Web Information Technology, 2007. ALPIT 2007. Sixth International Conference on*, pages 253–260, 2007.
- [10] J. Manning. Continuity conditions for spline curves. *The Computer Journal*, 17(2):181–186, 1974.
- [11] Census of Marine Life. Electronic tagging of marine animals <http://www.coml.org/comlfiles/scor/scor-tagging.pdf>.
- [12] Census of Marine Life. Topp.org: Tagging of pacific predators <http://www.topp.org/>.
- [13] Nova Southeastern University Oceanographic Center, Guy Harvey Research Institute. Integrated tracking portal <http://www.nova.edu/ocean/ghri/tracking/>, 2013.
- [14] Ocearch. Global shark tracker <http://sharks-ocearch.verite.com/>, 2013.
- [15] D. R. Nelson, J. N. McKibben, J. Strong, Wesley R., C. G. Lowe, J. A. Sisneros, D. M. Schroeder, and R. J. Lavenberg. An acoustic tracking of a megamouth shark, *Megachasma pelagios*: a crepuscular vertical migrator. *Environmental Biology of Fishes*, 49(4):389–399, 1997.

- [16] P. Nilas. A prototypical multi-locomotive robotic fish parametric research and design. In *Proceedings of the World Congress on Engineering and Computer Science*, volume 1, October 2011.
- [17] R. Parent. *Computer Animation: Algorithms and Techniques*. Morgan Kaufmann Publishers, 2nd edition, 2008.
- [18] M. E. Porter, C. M. Roque, and J. H. Long Jr. Swimming fundamentals: turning performance of leopard sharks (*Triakis semifasciata*) is predicted by body shape and postural reconfiguration. *Zoology*, 114(6):348 – 359, 2011.
- [19] C. W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '87, pages 25–34, New York, NY, USA, 1987. ACM.
- [20] R. G. Root, H.-W. Courtland, C. A. Pell, B. Hobson, E. J. Twohig, R. B. Suter, W. R. Shepherd III, N. C. Boetticher, and J. H. Long Jr. Swimming fish and fish-like models: The harmonic structure of undulatory waves suggests that fish actively tune their bodies. In *11th International Symposium on Unmanned Untethered Submersible Technology*, pages 378–388, 1999.
- [21] J. Scharold, N. C. Lai, W. R. Lowell, and J. Graham. Metabolic rate, heart rate, and tailbeat frequency during sustained swimming in the leopard shark *Triakis semifasciata*. *Journal of Experimental Biology*, 48:223–230, 1989.
- [22] K. Stephens, B. Pham, and A. Wardhani. Modelling fish behaviour. In *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, GRAPHITE '03, pages 71–78, New York, NY, USA, 2003. ACM.

- [23] J. Tan, Y. Gu, G. Turk, and C. K. Liu. Articulated swimming creatures. In *ACM SIGGRAPH 2011 papers*, SIGGRAPH '11, pages 58:1–58:12, New York, NY, USA, 2011. ACM.
- [24] X. Tu and D. Terzopoulos. Artificial fishes: physics, locomotion, perception, behavior. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, SIGGRAPH '94, pages 43–50, New York, NY, USA, 1994. ACM.
- [25] C. Wilga and G. Lauder. Three-dimensional kinematics and wake structure of the pectoral fins during locomotion in leopard sharks *Triakis semifasciata*. *Journal of Experimental Biology*, 203:2261–2278, August 2000.
- [26] A. Xydes, M. Moline, Christopher G. Lowe, Thomas J. Farrugia, and C. Clark. Behavioral characterization and particle filter localization to improve temporal resolution and accuracy while tracking acoustically tagged fishes. In *Ocean Engineering*, volume 61, pages 1–11, 2013.