VISUALIZATION OF COMPLEX UNSTEADY 3D FLOW: FLOWING SEED POINTS AND DYNAMICALLY EVOLVING SEED CURVES WITH APPLICATIONS TO VORTEX VISUALIZATION IN CFD SIMULATIONS OF ULTRA LOW REYNOLDS NUMBER INSECT FLIGHT

A dissertation submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

By

CHRISTOPHER M. KOEHLER
B.S., 2004, The University of Akron

2010
Wright State University

WRIGHT STATE UNIVERSITY

SCHOOL OF GRADUATE STUDIES

November 17, 2010

I HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER MY SUPERVISION BY Christopher M. Koehler ENTITLED Visualization of Complex Unsteady 3D Flow: Flowing Seed Points and Dynamically Evolving Seed Curves with Applications to Vortex Visualization in CFD Simulations of Ultra Low Reynolds Number Insect Flight BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Doctor of Philosophy.

_____
Thomas Wischgoll, Ph.D.
Dissertation Director

_____
Arthur Goshtasby, Ph.D.
Director, Ph.D. Program of CS&E

_____
Andrew T. Hsu, Ph.D.
Dean, School of Graduate Studies

Committee on
Final Examination

_____
Thomas Wischgoll, Ph.D.

_____
Yong Pei, Ph.D.

_____
Arthur Goshtasby, Ph.D.

_____
Haibo Dong, Ph.D.

_____
Joerg Meyer, Ph.D.

Abstract

Koehler, Christopher M. Ph.D., Department of Computer Science and Engineering, Wright State University, 2010. Visualization of Complex Unsteady 3D Flow: Flowing Seed Points and Dynamically Evolving Seed Curves with Applications to Vortex Visualization in CFD Simulations of Ultra Low Reynolds Number Insect Flight.

Three dimensional integration-based geometric visualization is a very powerful tool for analyzing flow phenomena in time dependent vector fields. Streamlines in particular have many perceptual benefits due to their ability to provide a snapshot of the vectors near key features of complex 3D flows at any instant in time. However, streamlines do not lend themselves well to animation. Subtle changes in the vector field at each time step lead to increasingly large changes between streamlines with the same seed point the longer they are integrated. Path lines, which show particle trajectories over time suffer from similar problems when attempting to animate them.

Dynamic deformable objects in the flow domain also complicate the use of integration-based visualization. Current methods such as streamlines, path lines, streak lines, particle advection and their many conceptual and higher dimensional variants produce undesirable results for this kind of data when the most important flow phenomena occurs near and moves with the objects.

In this work I present methods to handle both of these problems. First, the flowing seed point algorithm is introduced, which visually captures the perceptual benefits of smoothly animated streamlines and path lines by generating a series of seed points that travel through space and time on streak lines and timelines. Next, a novel dynamic seeding strategy for both streamlines and generalized streak lines is introduced

to handle deformable moving objects in the flow domain in situations where static seeding objects fail for most time steps.

These two methods are then combined in order to visualize the instantaneous direction and orientation of a flow which results from flapping objects in a fluid. Initial tests are performed with a single rigid flapping disk. Further tests were performed on a more complex biologically inspired CFD simulation of the deformable flapping wings of a dragonfly as it takes off and begins to maneuver. For this test, seeds are automatically chosen such that the formation, evolution and breakdown of the leading edge vortex is highlighted as well as the wing wake interactions that occur between the forewings and hind wings.

Table of Contents

List of Figures

List of Tables

Acknowledgement

I would like to thank Zach Gaston and Hui Wan for helping me film the dragonfly that was reconstructed as part of this work, Haibo Dong whose CFD solver generated the data that was used to test my visualization algorithms, Zongxian Liang and Zach Gaston for making wing camber and force history plots that my visualizations were compared to, and Thomas Wischgoll for providing the computer that my work was done on.

Also, a special thanks goes to all the members of my dissertation committee, Thomas Wischgoll, Yong Pei, Arthur Goshtasby, Haibo Dong, and Joerg Meyer.

This dissertation is dedicated to my mommy and daddy for sending me lots of money :O)

1 Introduction

Visualization is concerned with interactively representing data with computer graphics in order to gain insight into the data that would be otherwise impossible or extremely difficult to gain. Insight acquired through visualization can be used to answer specific questions about a data set and it can also help one gain an understanding of processes that were previously unknown.

While it is considered to be a branch of computer science and engineering, visualization shares quite a few principles with mathematics, cognitive science, physics, perception science, and many other disciplines and has applications in an even wider range of fields. Visualization is also very closely related to other sub-fields of computer science and engineering such as computer graphics in that it employs many graphical techniques that were not originally intended for understanding data. There are many data modalities for which visualization is useful, however this work focuses on flow visualization.

1.1 Flow Visualization

Flow visualization is a sub-field of scientific visualization concerned with visualizing vector fields. Vector field data occurs in many different natural science and engineering applications, however this work focuses on the velocity vector fields generated through computational fluid dynamics (CFD) simulations. Velocity is an intrinsically continuous quantity, however these simulations output the data as discrete arrays of velocity vectors.

As the processing power of computers increases so does our ability to simulate larger and more complex fluid phenomena. However, our ability to generate large multi-phase unsteady fluid simulations through elaborate immersed dynamically deforming geometry is ahead of our ability to visualize and understand the subtleties of the resulting data. This is due to the fact that flow visualization is limited both by computational power as well as human understanding. One cannot simply apply flow visualization algorithms that work with small simple vector fields to extremely large vector fields representing complex flows and expect a good result. In most cases the resulting visualizations will be extremely busy and hard to interpret. Thus, rapidly increasing CFD dataset sizes impose not only computational difficulties but perceptual challenges as well.

1.1.1 Categorization

Flow visualization algorithms have been categorized as being direct, texture-based, feature-based, partition-based or geometric [1-5]. Direct flow visualization is the simplest of the four categories. It uses a low level of abstraction when turning velocity vector fields into visual representations. Texture-based flow visualization creates a dense visual representation of a vector field by smearing a noise image in the direction of the flow at each point. Feature-based flow visualization defines what features about the data are interesting, tries to detect those features and then marks them in the data. Partition-based flow visualization partitions the entire flow domain based on properties of the vector field and then uses the partitions as a basis for visualizations. Geometric flow visualization, also known as integration-based flow visualization, involves using discrete geometric objects, that are locally tangent to the vector field, whose shape, transparency,

and color attributes are based on characteristics of the underlying flow phenomena. Each category of flow visualization techniques has its own strengths, weaknesses and challenges so the ideal visualization method to use is highly dependent on the nature of the data.

1.1.2 Geometric Flow Visualization

The research presented in this document deals primarily with geometric flow visualization. The main constructs of geometric flow visualization are streamlines, path lines, timelines and streak lines. The theory behind these techniques is presented in detail in Section 3. Multiple categorizations of geometric flow visualization have been proposed based on the dimensionality of the stream object used in the resulting visualization, the spatial dimensionality of the data domain, the spatial dimensionality of the seeding object and the temporal dimensionality of the simulated flow [4].

I propose adding further classifications to this scheme based on the characteristics of any objects within the flow domain that the visualization method must work with. In particular visualization methods could be classified based on whether they handle no objects, static objects, rigid moving objects or dynamically deforming objects in the vector field. While little has been done to specifically target visualizing flows containing dynamic objects, the fact that many traditional visualization methods suffer when they are presented with such data justifies this classification. One of the main contributions of this work is a seeding method that can more effectively handle objects that move and dynamically deform inside of an unsteady 3D vector field. The remainder of this introduction presents some fundamental concepts that are important for understanding

this research effort as well as additional motivation for this work. Figure 1.1 shows an example of several streamlines, the most well known geometric visualization construct.

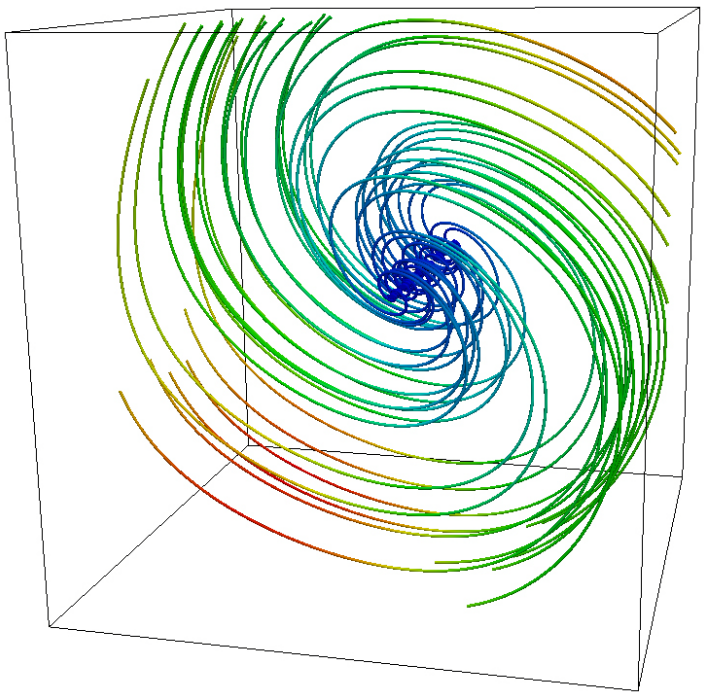

Figure 1.1: Several streamlines in a simple dataset containing one vortex.

1.2 Fluid Dynamics

The data visualized with the methods described in this document is based on fluid dynamics. Fluid dynamics is a sub-field of fluid mechanics that studies fluids in motion. Any substance that undergoes a constant deformation in the presence of shear stress is considered a fluid. Fluid dynamics has a wide range of applications, including weather prediction, micro air vehicles, movie special effects, wind turbines, micro fluidic devices, large aircraft and blood flow. The following section presents a brief overview of the theory of fluid dynamics and simulation because it is important that the visualization

methods used to understand fluid behavior be based in a firm understanding of the theory behind the original application.

1.2.1 Navier-Stokes Equations

The Navier-Stokes equations are a set of partial differential equations that describe the motion of incompressible fluids. They hold throughout a fluid and can be written as follows:

$$\frac{\delta \vec{u}}{\delta t} + \vec{u} \cdot \nabla \vec{u} + \frac{1}{\rho} \nabla p = \vec{g} + v \nabla \cdot \nabla \vec{u} \tag{1.1}$$

$$\nabla \cdot \vec{u} = 0 \tag{1.2}$$

Where $\vec{u}$ is the velocity of the fluid, ρ is the density of the fluid (m/V), p is the pressure that the fluid exerts on anything, $\vec{g}$ is the acceleration due to gravity, and v is the kinematic viscosity coefficient. The kinematic viscosity coefficient is the dynamic viscosity coefficient μ divided by the density. The Laplacian operator $\nabla \cdot \nabla$ is a measure of how far a quantity at a point is from the same quantity in the area around it, and ∇ is the gradient operator.

The first equation, which is known as the momentum equation or force equation, is essentially Newton's second law $\vec{F} = m\vec{a}$. The acceleration can be rewritten as:

$$\vec{a} \equiv \frac{D\vec{u}}{Dt} \tag{1.3}$$

With the capital D material derivative notation, so Newton's equation becomes:

$$\vec{F} = m\frac{D\vec{u}}{Dt} \tag{1.4}$$

The force $\vec{F}$ can then be replaced with the sum of all forces acting on a particle in a fluid. The pressure force, or body force can be measured as the negative gradient of pressure $-\nabla p$. The viscosity force can be written as the product of the dynamic viscosity coefficient and the measure of how far the velocity at a particle in the fluid is from the surrounding velocities $\mu\nabla \cdot \nabla\vec{u}$. The force due to gravity is written as $m\vec{g}$. The pressure and viscosity forces must be integrated over the whole volume of fluid, so they are multiplied by the volume V as an approximation.

All these forces can be combined as follows:

$$m\vec{g} + V\mu\nabla \cdot \nabla\vec{u} - \mathrm{V}\nabla p = m\frac{D\vec{u}}{Dt} \tag{1.5}$$

If we divide by the volume V and rearrange several terms we can almost get back to the original equation:

$$\rho\frac{D\vec{u}}{Dt} = \frac{m}{V}\vec{g} - \nabla p + \mu\nabla \cdot \nabla\vec{u} \tag{1.6}$$

$$\frac{D\vec{u}}{Dt} = \vec{g} - \frac{1}{\rho}\nabla p + \frac{\mu}{\rho}\nabla \cdot \nabla\vec{u} \tag{1.7}$$

$$\frac{D\vec{u}}{Dt} + \frac{1}{\rho}\nabla p = \vec{g} + \frac{\mu}{\rho}\nabla \cdot \nabla\vec{u} \tag{1.8}$$

The kinematic viscosity coefficient can then be substituted in:

$$\frac{D\vec{u}}{Dt} + \frac{1}{\rho}\nabla p = \vec{g} + v\nabla \cdot \nabla\vec{u} \tag{1.9}$$

The material derivative is then rewritten based on the chain rule. The result can be substituted back into the previous equation, which gets us back to the original momentum equation.

$$\frac{D\vec{u}}{Dt} = \begin{bmatrix} Du/Dt \\ Dv/Dt \\ Dw/Dt \end{bmatrix} = \begin{bmatrix} \delta u/\delta t + \vec{u} \cdot \nabla \mathrm{u} \\ \delta v/\delta t + \vec{u} \cdot \nabla \mathrm{v} \\ \delta w/\delta t + \vec{u} \cdot \nabla \mathrm{w} \end{bmatrix} = \frac{\delta\vec{u}}{\delta t} + \vec{u} \cdot \nabla\vec{u} \tag{1.10}$$

1.2.2 Computational Fluid Dynamics

The field of computational fluid dynamics studies how to use numerical methods to simulate fluid flow. Using a CFD solver, a model is built to represent some real world system that you want to study or visualize. The main benefit of using CFD is that it allows you to study things that are difficult to produce via experimentation. Its details are outside the scope of this work, but the data used to test the visualization algorithms in this document was generated with Dong et al.'s immersed boundary method CFD solver [6, 7].

1.3 Vector Data Modality

The data resulting from CFD simulations is vector based. An n-dimensional vector is a tuple of n scalar components $\boldsymbol{v} = (v1, v2, \ldots, v_n), \boldsymbol{v}_i \in \mathbb{R}^n$, however only 2D and 3D vectors are normally used in CFD simulations. A vector field $\boldsymbol{v}$ is a function $f: D \rightarrow \mathbb{R}^3$, where D is a subset of $\mathbb{R}^3$, and a vector dataset is a discrete sampling of a

vector field. Since CFD is able to simulate time-dependent flow phenomena of 3D compressible flows, the data typically consists of a series of vector fields output at a discrete number of time steps.

1.4 Divergence, Vorticity and Helicity

For each vector field output at each time step of a CFD simulation there are several metrics that are typically computed at each vector. Divergence is a scalar quantity used to measure the increase or loss of mass at any point in a vector field. If the divergence at a point in the vector field is positive that means the flow is spreading outward from that point, and if it is negative then the flow is being sucked into that point. Mathematically, the divergence of a vector $v = (v_x, v_y, v_z)$ is defined as follows:

$$divergence(v) = \frac{\delta v_x}{\delta x} + \frac{\delta v_y}{\delta y} + \frac{\delta v_z}{\delta z} \tag{1.11}$$

Vorticity is another important measurement that can be made at each point in a vector field. Vorticity is a vector quantity and is essentially a measure of how much a massless particle placed in the vector field will "spin". It characterizes the direction and speed of rotation at every point in the vector field. Vorticity will be of particular importance later when the application of visualizing air flow around a dragonfly wing is discussed. Vorticity can be used to calculate helicity, which is basically the degree to which a vector field exhibits a cork screw shaped motion. Mathematically, vorticity is defined as follows:

$$vorticity(v) = \left(\frac{\delta v_z}{\delta y} - \frac{\delta v_y}{\delta z}, \frac{\delta v_x}{\delta z} - \frac{\delta v_z}{\delta x}, \frac{\delta v_y}{\delta x} - \frac{\delta v_x}{\delta y}\right) \tag{1.12}$$

1.5 Motivation

There have been many derivatives of streamlines, path lines, timelines and streak lines attempting to enhance them or increase their dimensionality. However, there have been very few attempts at integrating them. Thus, the first goal of this work was to combine the benefits of multiple geometric flow line primitives into one method. Another issue that drove this work is the lack of flow visualization targeted at handling unsteady flows containing deformable immersed objects that are causing flow disturbances. I suspect this is both because of the additional difficulties it creates and also because such data is rare. The goal of applying this visualization to a flapping wing dragonfly CFD simulation is to understand how vortices behave in insect flight. Ultimately the hope is that the study of insect flight will lead to the creation of smaller and more efficient micro air vehicles.

The remainder of this manuscript is organized as follows: Section 2 presents the results of an exhaustive literature review of steady and unsteady geometric flow visualization methods for 2D and 3D data, applied flow visualization, vortex detection and flapping flight theory. Section 3 reviews several basic concepts of geometric flow visualization and then introduces the flowing seed point algorithm as well as several unique methods to generate dynamic seed curves. Section 4 presents data generation methods, implementation details, rendering techniques that can improve perception of geometric visualization, and a kinematics analysis of one of the data sets used to test the visualization methods. Section 5 presents the results of applying the new visualization methods to the series of vector fields from a quad wing dragonfly simulation. Section 6 reviews the main points of this work and proposes some future work.

2 Related Work

This section presents a comprehensive literature review of integration-based geometric flow visualization as well as several other areas of interest. The majority of the work can be categorized based on whether it is designed for either steady or unsteady flow. The dimensionality of the flow domain is used to further categorize the works. Methods that utilize programmable GPU's are grouped into a separate section. Vortex detection methods are also discussed because vortices are of particular interest in several of my data sets. Finally, several applications of geometric flow visualization on real world data sets are discussed. Since the goal of the proposed visualization methods is to improve the understanding of insect flight, a section is devoted specifically to previous work in flapping flight theory.

2.1 Steady Flow Visualization

A steady flow is independent of time. Thus, as time elapses the vector field representing the flow does not change. Due to this property, streamlines, streak lines and path lines are all identical in such flows, which simplifies things greatly. The research presented later in this document primarily deals with unsteady flows, however the vector fields at any instant in an unsteady flow can be handled the same way as a steady flow would be, so this is important background information.

2.1.1 2D Steady Flows

This section looks at previous work in visualizing 2D steady state vector fields with integration-based methods. Typically the goal of visualizing a 2D steady flow is to get an even streamline coverage that captures all the important features. Also the ability for the user to interactively control the streamline density is a convenient feature of such applications.

One of the first attempts at a streamline seeding algorithm for 2D steady flows was presented by Turk and Banks [8]. The algorithm starts with random initial seed points and then an image-guided energy function is used to incrementally improve the seeds until a desired streamline density is reached. Examples were presented on how this can be done for both streamlet seeding as well as for seeding longer streamlines. Prior to this work streamline seeding was done by regular grids, random sampling and user interaction only.

Another early attempt at visualizing 2D steady flow fields with evenly spaced streamlines was introduced by Jobard and Lefer [9]. The algorithm works by starting with a single streamline and then generating the next seed at a minimal distance from the first streamline. The new streamline is integrated in both directions until it either comes to close to another streamline or it leaves the vector field. All potential seed points surrounding one streamline are integrated before trying potential seeds around a new streamline. This process is repeated until there are no more potential seed points. The main benefit of this algorithm was that it requires only a single pass, unlike the iterative approach in [8]. Streamline density is user controlled by changing the separation distance

between adjacent streamlines. This streamline seeding algorithm has been combined with animated line integral convolution in order to animate steady 2D flows [10].

Another non-iterative seeding strategy for placing streamlines in 2D steady flows was developed by Verma et al. [11]. The main goal of this algorithm is to place streamlines near critical points in the flow while maintaining a relatively low streamline density. This is accomplished by segmenting the vector field into regions containing only one critical point. Each critical point region is seeded based on a predefined template. Several randomly placed streamlines are then added to help fill in sparse areas in the coverage. Streamlines are terminated based on a user controlled minimum distance that they are allowed to come towards other streamlines. It is mentioned that the algorithm is fast enough to perform interactively even in 3D, however all examples given are for 2D flows.

Lefer et al. presented a refined method for animating streamlines in order to visualize both velocity magnitude and flow direction of a steady 2D flow. Streamlines are densely seeded and colored so you can see a pattern on them. They are animated by making these patterns move down the streamlines in the direction of the flow. This is achieved by shifting color table entries cyclically, which yields a completely cyclic animation.

Mebarki et al. proposed another method for seeding streamlines in 2D steady flow vector fields [12]. They generate seeds for new streamlines at the farthest point in the vector field from all current streamlines, which fills gaps in the coverage and promotes longer streamlines. Results using this method give similar results to those achieved in [8], however their greedy algorithm is around 200 times faster and conceptually simpler.

Liu et al. presented an advanced evenly-spaced streamline placement algorithm that improves the quality of streamline placement in 2d steady state vector fields [13]. Double queuing is used to prioritize seed points in the neighborhood of critical points as well as seed points that will result in long streamlines. In order to increase speed, they used Hermite polynomial interpolation to decrease the necessary samples per streamline while still generating streamlines that are visually smooth. Also, a method for detecting spiraling streamlines is presented in order to reduce visual clutter.

More recently Li et al. developed an artistically inspired streamline seeding strategy to generate illustrative streamlines for 2D steady flow visualization [14]. The goal of this work was the capture flow patterns with the minimum set of streamlines while deemphasizing less essential and repetitive portions of the flow. The algorithm works by computing a local similarity measure among streamlines in the same region. The similarity measure is accumulated along each point of a streamline and a greedy algorithm is used to choose the next seed point that has the lowest degree of similarity to the current streamlines.

2.1.2 3D Steady Flows

In 3D vector fields, self occlusion of geometric visualization constructs is a major problem that was not an issue in 2D vector fields. Thus, even streamline coverage of the entire flow field is no longer a good goal because it will result in very busy visualizations where important flow features are occluded by the flow at the border of the vector field. The goal is typically to choose seed points that only capture important features of the flow so they will be easier to see. For the same reason, choosing a suitable streamline

integration time becomes more challenging. If streamlines are integrated to long they tend to leave the important areas and begin to contribute to visual clutter. Choosing a suitable rendering method is another challenge when displaying 3d geometric visualizations because it can potentially help alleviate depth of field ambiguities.

2.1.2.1 Surface Seeding in 3D Steady Flows

The extra dimension in 3D vector fields allows for more freedom when defining geometric objects upon which seeds can be generated. A method for rendering surface-particles in 3D flows that exploits this fact was presented by van Wijk [15]. Surface-particles are modeled as points in space along with a normal vector, which allows diffuse and specular lighting equations to be used during rendering. Several possible geometric sources are defined, from which the surface-particles are emitted either from random or regularly spaced positions. Some of the choices presented for seeding objects are points, lines, circles, ellipses, rectangles and spheres. They allowed for both pulsatile or continuous emission. In the case of continuous emission, the result is a streamline because the flow is steady.

2.1.2.2 2.5D Visualization of 3D Steady Flows

One way to visualize 3D vector fields that contain solid objects in the flow domain is to use 2.5D visualization. Essentially 2.5D visualization is the generation of stream objects on the surfaces of objects in the flow field. For instance if there was a simulation of a propeller, the streamlines would be drawn on the surface of the propeller instead of everywhere in the vector field.

Mao et al. developed a method for placing uniformly distributed streamlines on 3D parametric surfaces in curvilinear grids [16]. A curvilinear grid has the same structure as a regular grid, but the cells are quadrilaterals or cuboids instead of rectangles. The computational space of a curvilinear grid is the rectangular grid which defines its logical organization. The algorithm works by mapping vectors from the surface of the object into the computational space of the surrounding curvilinear grid. From there, the evenly spaced streamline algorithm developed by Turk and Banks [8] is used. A new energy function is added to minimize the mapping distortion caused by the uneven density of the curvilinear grid.

Spencer et al. introduced an image-based automatic streamline seed generation algorithm for vector fields defined on 3D surfaces [17]. A key feature of this approach is that the vector field is projected onto the image plane before integrating any streamlines. This guarantees that the streamlines will remain evenly space regardless of the mesh orientation and they will not occlude each other. The projection also makes this a very efficient algorithm because streamlines can be computed much faster than if the tracing was performed directly on the surface in model space and they don't have to waste time tracing streamlines on portions of the surface which will be occluded.

Rosanwo et al. proposed a streamline seeding method for 3D surfaces which is guided by dual streamlines that are orthogonal to the vector field [18]. Streamline seeds are only placed on the dual streamlines, which significantly reduces the search space for seed placement. This approach can be extended to curved surfaces within the flow domain without requiring a surface parameterization.

2.1.2.3 Streamline Integration in 3D Steady Flows

There are several sources of error when using numerical methods to calculate streamline trajectories. The numerical integration method used is one source of error. For example, low order Euler integration yields misleading results. Also the step size must be appropriate for the resolution of the grid you are working on. The cell interpolation scheme is another potential source of error. This is particularly evident in areas of high streamline curvature. The grid type used by the original CFD simulation also impacts the accuracy of the streamlines. The remainder of this section presents an overview of several published works on the computational aspects of tracing streamlines in steady flows as well as several methods for evaluating and comparing streamline accuracy.

A new method for tracing streamlines in steady 3D vector fields based on stream functions was developed by Kenwright and Mallinson [19]. The flow within each cell is represented by two stream functions. Another way of describing a streamline is that it is the intersection of two stream surfaces. Since stream surfaces are always tangential to the flow, a line of intersection between two of them must also be tangential to the flow. Calculating streamlines via tracking constant values of each stream function is mathematically equivalent to finding the intersection of two stream surfaces.

Ueng et al. described techniques for tracing streamlines in unstructured grids [20]. Tetrahedral cells of the unstructured grid are transformed to a so called canonical coordinate system in order to simplify calculations. Also, a Runge-Kutta based numerical integration implementation is proposed to speed up the streamline calculation

process in the canonical coordinate system. The methods were also applied to streamribbon and streamtube construction.

Lodha et al. presented methods for visualizing uncertainty in streamline visualizations [21]. As mentioned in the intro to this section, there are many possible sources or numerical uncertainty and errors when calculating streamlines. In order to be confident in one's results it is convenient to be able to visualize the relative errors of different approaches. In particular, this application focuses on the uncertainty caused by numerical integration. Uncertainty glyphs, flow envelops, priority sequences, animations, rakes and trace viewpoints were used to compare the effectiveness of various numerical integration algorithms at tracing streamlines.

An interactive flow visualization method designed to work on locally refined Cartesian grids was presented by Schulz et al. [22]. The grids local refinement is used in areas of the flow that have more interesting features. In particular a simulation of a vehicular body design is visualized. The portion of the grid corresponding to the area around the wheels, in front of the windshield and at the very front and back of the car was divided down to the smallest cell size.

Although the vehicle was a static object in this simulation, this is one of the few works that discusses some of the complications that arise when there are objects of any kind in the flow domain [22]. In simulations where a curvilinear grid is used the object could be handled automatically in that the grid border describes the vehicle surface, however with a Cartesian grid there must be an explicit representation of any objects in the flow domain. Due to numerical errors it is possible for streamlines near the object surface to actually intersect the surface. One option they propose is to just end the

streamline if it intersects an object surface due to numerical errors. An octree data structure was used to simplify collision detection between streamlines and the large triangular mesh that represented the vehicle in their study.

Comparitive visualization attempts to compare the content of two or more data sets. Verma and Pang outlined methods for comparing vector data sets [23][11]. They classify the potential comparative visualization approaches as being either on the image, data or feature level. Feature level comparison is essentially an extension of data level comparison except features that were derived from the data set are the objects being compared. One relevant application independent flow feature is the noble streamline. The metric they use to visually compare streamlines is the Euclidean distance between corresponding streamline points. These corresponding points are connected with lines, strips or spheres in order to visually convey the differences between the streamlines. Their methods work with streamlines that were generated with different numerical approaches and with different data sets.

Pugmire et al. evaluated two approaches to computing streamlines in parallel for huge datasets and also presented a new parallelization algorithm that is a hybrid of the two previous approaches [24]. Both previously presented methods, static data decomposition and load on demand, use straight forward parallelization strategies. Their novel approach is a hybrid in that it parallelizes over both the data and the set of streamlines to be integrated simultaneously by balancing I/O, computation, and communication. Tests showed their method provided better scalability with most datasets.

2.1.2.4 Seed Point Generation in 3D Steady Flows

Seed point generation is very important in 3D flows because it has a significant impact on what features are captured by the streamlines. There has been work using both automatic and user controlled seeding strategies. Laramee et al. presented a comparison of geometric, direct and texture-based visualization methods to analyze swirl and tumble flow in an automotive engine [25]. As part of this application they created an interactive streamline seeding plane tool. The seeding plane provides the user with six degrees of freedom for choosing seed points.

Ye et al. presented a strategy for streamline seeding that attempts to balance field coverage with clutter reduction [26]. The algorithm first searches the flow for critical points and then categorizes them based on the eigenvalues of their Jacobian matrices. Seeds are then placed in the vicinity of the critical points based on a series of seeding templates. In order to fill in areas without sufficient coverage Poisson seeding is used. Feature based filtering of the resulting streamlines is used to reduce clutter.

Li and Shen proposed an image-based method for streamline seeding in 3D vector fields [27]. Projections of streamlines into image space are used to control seed placement in order to avoid streamline overlap and visual clutter. The seeds generated in the 2D image plane are unprojected back into the flow domain so they can be integrated. This approach assures a minimum spacing between the projections of streamlines in the image plane.

Chen et al. developed a streamline seeding method for 2D and 3D steady flows that uses a similarity measure to grow streamlines from a dense set of potential seed points [28]. Seeds are randomly chosen from the set of candidates and integrated until

the similarity measure between the current streamline and any existing streamline falls below a threshold value. The similarity measure takes into account Euclidean distance as well as the shape and orientation differences between pairs of streamlines. A streamline evaluation method that attempts to reconstruct the vector field from a set of streamlines and compare it to the original is also presented.

2.1.2.5 Perceptual Streamline Enhancements

Several methods beyond just seed point placement have been proposed to address the perceptual challenges of streamlines in 3D vector fields. In particular these methods attempt to improve perception of depth, directional orientation and occlusion. When combined with the seeding methods mentioned in the previous section, streamlines are a very powerful tool for visualizing 3D steady flow.

Zöckler et al. introduced a technique for visualizing 3D vector fields with dense illuminated streamlines [29]. The traditional Phong shading model utilizes surface normal vectors to determine light intensity on mesh objects. Streamlines, which are represented by line primitives, have infinitely many normal vectors at any point. In order to apply the lighting equations to line primitives, they choose the normal vector that is coplanar with the light vector and the tangent vector. The resulting images increase rendering quality and make it easier to visualize small features in the vector field.

Mallo et al. presented an enhanced method of illuminating streamlines in order to increase realism [30]. The main contribution of this algorithm is a cylinder averaging technique that improves diffuse and specular reflections on bundles of streamlines. The

method supports multiple or infinite light sources as well as both orthographic and perspective projections.

Fuhrmann and Gröller introduced the concept of dashtubes for visualizing 3D steady flows [31]. Dashtubes are essentially cylinders extruded along streamlines with animated opacity mappings to help visualize the flow velocity and direction. The opacity mapping helps lessen occlusion while providing directional information. The dash length is kept independent of the underlying velocity because high velocity areas would result in long gaps.

Mattausch et al. used illuminated evenly-spaced stream tubes to explore the perceptual benefits of several other rendering options for visualizing 3D steady flows [32]. A slightly revised 3D version of Jobard and Lefer's seeding strategy was used [9]. They then tested several streamline enrichment methods such as end tapering, mapping scalar quantities to streamtube density, using opacity to show direction, color coded depth, streamline halos, magic volumes, region of interest placement and spotlight rendering.

In order to further address the perceptual problems inherent to 3D streamlines Laramee et al. explored several techniques such as oriented streamlines, streamlets and streamcomets[33]. Oriented streamlines use opacity to depict flow orientation in still images. Streamcomets work in a similar fashion but they offer more degrees of freedom to visually represent properties of the data. Tests were performed on several real world datasets to evaluate the effectiveness of the methods.

Weigle and Banks presented a study comparing the perceptual benefits of perspective projections and global illumination when rendering dense 3D streamtubes

[34]. Their tests involved human subjects who attempted to identify the closer of two streamtubes from a densely seeded area and flow domain shapes from varying densities of streamtubes. Their results show that physically-based illumination is a strong cue for accurate perception of 3D streamtubes.

Although intended for rendering molecular structures and not streamtubes, the rendering techniques of Tarini et al. have potential applications to rendering streamtubes and other geometric flow visualization constructs [35]. Their methods combine ambient occlusion, depth-aware halos, depth-revealing contour lines and intersection-revealing contour lines to create molecular renderings that are more informative and more capable of revealing shape.

Schussman and Ma developed a method of rendering extremely dense line data which can be used to improve perception of densely seeded streamlines in huge data sets [36]. Their method works by applying a lighting model to the lines and then sampling them into anisotropic voxels. These voxels are rendered with traditional volume rendering. The result is improved rendering efficiency as well as improved perception of structure and depth.

Salzbrunn and Scheuermann introduced streamline predicates in order to illustrate connections between streamlines and features as well as to answer questions about the overall structure of all streamlines in a flow [37]. Predicates are essentially functions that return Boolean values. Streamline predicates tell if a specific feature exists in a streamline. For example streamline predicates might tell if a streamline flows through a specific vortex or not. Salzbrunn et al. later extended this concept to path line predicates in unsteady flows [38].

2.1.2.6 Integral Surfaces in 3D Steady Flows

Hultquist presented the concept of a stream surface for visualizing steady 3D vector fields [39]. An integral or stream surface is the result of joining multiple streamlines that were seeded from the same line or curve into a single surface. This is typically accomplished by connecting all of the sample points on each streamline with a polygon mesh. Like other stream objects, the main challenge of flow visualization with integral surfaces is to minimize occlusion while capturing all important flow features.

From the implementation side stream surfaces introduce more complexity. Areas of divergence in the flow are particularly troublesome. For example, if the streamlines that originated on the same rake take drastically different paths through the vector field how should this be handled in the discrete representation of the stream surface? Similar situations happen if a portion of a stream surfaces converges to a critical point which it takes an infinite time to reach while other parts diverge into different areas of the flow. One way this can be handled is through tearing the stream surface, but this is logically complex and can have undesirable visual effects if handled incorrectly. This section details previous research on using integral surfaces to visualize steady state 3D vector fields.

Van Wijk presented a new method for generating stream surfaces [40]. In this method a stream surface is represented by the equation of an implicit surface. The initial curves of a stream surface can then be defined by this function at the boundary of the flow field. This approach can only generate stream surfaces that intersect the flow boundary.

Löffelmann et al. developed a technique to enhance stream surface visualizations called streamarrows [41]. Streamarrows are essentially arrow-shaped portions that have been removed from a stream surface via transparency mapping. This helps alleviate the occlusion problems inherent to large opaque stream surfaces. Streamarrows were implemented by mapping a texture of regularly shaped arrows onto a stream surface and then using it to control the surface's transparency.

Scheuermann et al. proposed a technique for generating stream surfaces on tetrahedral grids[42]. Within each tetrahedral cell, interpolation is based on barycentric coordinates. The stream surface is traced over one tetrahedron at a time. With this scheme, the portion of a stream surface inside any tetrahedral cell is also a ruled surface.

Garth et al. introduced a stream surface based method for visualizing vortices [43]. Their method for generating stream surfaces provides enhancements to Hultquist's method [39] so that it yields more accurate results in areas with very complicated flow structures. In particular their streamline integration is based on arc length instead of parameter length. They also explained a method to determine boundary surfaces of vortex cores to improve vortex visualization.

Laramee et al. presented a visualization method that combines stream surfaces with texture advection [44]. Texture advection is a flow visualization method that smears white noise in the direction of the flow. Stream surfaces alone have difficulty conveying flow direction. When texture advection is applied to stream surfaces it helps solve this problem. They demonstrate this method's effectiveness with an engine simulation dataset.

Peikert and Sadlo presented methods for visualizing topological features with stream surfaces [45]. They demonstrated that topology based stream surfaces can capture relevant flow features in the vicinity of periodic orbits and critical points while maintaining a relatively simple geometric representation.

2.1.2.7 Integral Volumes in 3D Steady Flows

Integral volumes are the result of further increasing the dimensionality of the seed object to a plane or surface. While this in and of itself does not imply a volumetric representation of the resulting object, the term integral volume is used to refer to all 3D stream objects for simplicity.

Schroeder et al. introduced the stream polygon concept for visualizing 3D steady flows [46]. Polygons placed normal to the flow are used as seed points for streamlines. The polygons are then either swept along streamlines to create tube structures or used to place new polygons at each streamline sample point. Scalar metrics of the underlying vector field such as normal strain, shear strain and rotation can then be visualized by varying the radius of the polygon and the shading of the resulting tube.

The concept of flow volumes, the volumetric equivalent of streamlines, was first introduced by Max et al. [47]. The idea was inspired by real world flow visualization experiments where smoke is injected into physical flows. They used volume rendering of semi-transparent flow volumes to simulate the results of these physical experiments. Becker et al. extended Max's concept of flow volumes [47] to work in unsteady flows by using streak lines [48].

Flow volumes suffer from information on the interior of the volume being lost. Implicit flow volumes were introduced in order to address the problem by Xue et al. [49]. Two methods are presented for rendering implicit stream flows. The first method is a slice-based 3D texture mapping renderer and the second method is based on a interval volume renderer.

2.2 Unsteady Flows

Unsteady flows evolve over many time steps, thus the sets of streamlines at different time steps are different For steady flows, you only need one vector field because it is the same at any point in time, however for unsteady flows you need a separate vector field for each time step of the simulation. This leads to extremely large data sets. Three dimensional simulations performed over many time steps on grids with a high resolution can easily grow into the terascale. Faster methods for generating integral curves in such data is a very active research area. Recently multi-core programmable GPUs have been applied in this area, however there is a bottleneck in that the GPU's onboard memory can only hold a few time steps. This section presents an overview of the previously published methods of visualizing unsteady flow data.

One key thing to note about this previous work is that there have been relatively few attempts at animating true streamlines and even fewer attempts to deal with dynamically deforming and moving objects within the flow field. Methods for solving these two problems will be addressed in detail later and represent a major contribution of the current work.

2.2.1 2D Unsteady Flows

Jobard and Lefer developed one of the first methods for visualizing 2D unsteady flow data by using animated streamlines [50]. Their approach to circumventing the inherent problems streamlines have with animation is to establish a correlation between streamlines at consecutive time steps in order to create smooth animations. The goal is to reduce motion in the animation that is not along a streamline. To this end they employ a feed forward algorithm where streamlines at any given time step are used to determine the streamlines at the next time step.

For each point on each candidate streamline at time t, a corresponding seed is generated at time $t + 1$. These corresponding streamlines are all integrated and compared to their respective candidate streamlines. The criterion used to measure correspondence with the candidate streamlines is the average distance between corresponding pairs of sample points. After corresponding streamlines have been established, the next step of their algorithm is to fill in sparse areas with additional streamlines. This step would probably not be included if a 3D version of the algorithm were developed due to occlusion issues.

Cyclical animated textures based on Lefer's previous work in [51] are applied to the resulting streamlines to give the impression that flow is moving in the direction of the streamline. As will be explained later in this document, the methods presented here do not rely on animating textures on the streamline because the end points of the individual streamlets serve the same purpose as all their seed points at a given instant travel along their respective generalized streak line as time elapses. Thus, the illusion that the

streamlines are moving in a direction that is instantaneously orthogonal to the flow is achieved via entirely geometric means.

The work of Jobard and Lefer [50] represents one of the few attempts at animating streamlines in unsteady vector fields. It's concept is novel and very useful, however more work needs to be done in this area. In particular, it has not been tested with 3D data and there is no guarantee that there will be streamlines in the most critical flow regions that have streamlines at consecutive time steps for which a good correspondence exists. The fact that correspondence between streamlines at consecutive time steps drops as the integration time of the streamlines increases is another drawback of the algorithm. Also, the fact that seed points of corresponding streamlines are not necessarily consistent between time steps could potentially lead to a slightly jerky effect in the animation as alternating portions of streamlines are more active between time steps.

2.2.1.1 Generalized Streak Lines

Wiebel et al. introduced the concept of generalized streak lines, which are essentially streak lines that allow a moving seed point [52]. They use this method to visualize vortices that develop and move along static objects in 4D flows simulations. A new method is also presented to track singularities over curved surfaces based on the shear stress field. As these singularities are tracked on object boundaries they are used as the particle emission points for the generalized streak lines.

I have only found two instances where generalized streak lines were used in the literature. The idea of a moving seed point is a simple concept however it adds additional

challenges to the seeding process but also has much potential for improved streak line based visualization methods. Clearly more work is needed to explore seeding strategies for generalized streak lines as well as their potential uses.

2.2.2 3D Unsteady Flows

Visualization of unsteady 3D flows is one of the most difficult areas of flow visualization. It has similar perceptual challenges to visualizing 3D steady flows and 2D unsteady flows but it is inherently slower due to the much higher disk and memory requirements for storing the data. For simplicity 3D unsteady flow can also be referred to as 4D flow due to the three spatial dimensions and one time dimension. This section outlines several previously published works on tracing particles through multiple steps of a time dependent flow, seed generation in unsteady flows and higher dimensional geometric visualization such as integral surfaces and volumes.

2.2.2.1 Line Integrals in 3D Unsteady Flows

Lane developed a method for visualizing large 3D unsteady flows by using streak lines [53]. The largest dataset, a Descending Delta Wing, consisted of a grid with 900,000 points simulated for 1,800 time steps resulting in 64.8 GB of flow data. Both the data sets in this work also involved moving grids, a feature rarely addressed in the literature, which further complicates visualization. Streak lines were generated from fixed seed points in the data sets. For the Delta Wing dataset, 340 seed points were used and for the other dataset, a V-22 tilrotor aircraft, 400 seed points were used. In this work the seeds were chosen manually. In order to handle the large data size, they limited the

number of time steps that could be interactively examined so that all time steps could fit in the available memory.

Lane also developed UFAT, a system for tracing particles through unsteady flow fields [54]. The two main benefits of UFAT over other systems available at the time was its ability to handle a very large number of time steps and the fact that it could handle moving grids. Methods are presented for particle integration in both physical and computational coordinate space. Streak lines at each time step are computed in a preprocessing step and stored on disk so they can be viewed later without having to integrate them again, thus speeding up the application. The system also visualizes scalar metrics of the flow data such as temperature, density, pressure and mach number via color mapping. The data from several simulations based on real-world problems were used to test the system.

In the context of this work, it is important to note that while streak lines have been used effectively for data with dynamic objects in the flow field [53, 54] they still have limitations for this kind of data. One such limitation is that they are not tangent to the underlying vector field at any given moment, which could potentially mislead a user. Also, most streak line implementations are limited by the fact that the source from which particles is emitted over time is fixed. If a simulation occurs over many time steps there is no guarantee that particles emitted from the same source will interact with the objects moving through the flow. Another weakness of streak lines is the fact that they can potentially intersect moving objects in the flow field.

A method for efficient tracing of particles and streak lines in large unsteady vector fields resulting from aeronautical simulations was developed by Kenwright and

Lane [55]. In the case of curvilinear grids, particle tracing is much slower than in regular grids, however accuracy is sacrificed if the grid is transformed to a more convenient computational space. Their method performs integration, step size control and interpolation on curvilinear grids by using tetrahedral decomposition to speed up point and velocity interpolation.

Teitzel et al. present an analysis of the relationship between the errors caused by interpolation and those caused by numerical integration in order to develop a robust adaptive step size integration scheme that offers improved efficiency without sacrificing performance [56]. When dealing with discrete samplings of time dependent vector fields there are inherent errors due to interpolation of velocity vectors within cells and interpolation between entire time steps. For such rough data, higher order numerical integration algorithms are potentially more accurate than necessary. They propose an adaptive 3rd order Runge-Kutta scheme and prove that it is accurate enough while much faster than higher order schemes.

Sparse grids are a method that can be used to reduce the total number of data points in a vector field. They have been proposed as a method of data compression for visualizing the huge data sets resulting from time dependent unsteady 3D flow simulation by Teitzel et al. [57]. Sparse grids have their own set of challenges though because not all particle tracing algorithms can handle them. Teitzel and Ertl proposed new methods for effectively tracing particles in unsteady flows that are represented by sparse grids [58]. They also introduce a method for particle tracing on sparse curvilinear grids.

Interactive integration-based flow visualization has been the goal of much of the research presented in this section. Schirski et al. developed a software framework which

combines much of this work into a single unified package [59]. The end result is a system that allows multiple interactive visualization options for the exploration of unsteady flow fields in a virtual environment.

2.2.2.2 Seeding in 3D Unsteady Flows

One of the first attempts at visualizing unsteady 3D flow was developed by Bryson and Levit in their virtual windtunnel [60]. A user wears a six degree of freedom head mounted stereo CRT monitor to see inside the virtual windtunnel. A glove controller is used to interactively choose seed points for streamlines, streak lines and path lines.

Wiebel and Scheuermann presented methods for creating 3D visualizations of 4D flows [61]. Their goal is to address some of the drawbacks of animation like the lack of a transient impression at any given time step. They generate a so called eyelet point and use it to define bundles of streak lines and path lines that pass through it at different times. Particles are selected such that they pass the eyelet at equidistant and evenly distributed points in a time interval. Then a method to use these bundles of lines to create surfaces is explained. In addition, iso-surfaces are places at areas of the vector fields at which local metrics vary heavily over time. These areas of high activity are also used for choosing more informative eyelets because the path lines passing through an eyelet only change if there are changes in the flow at that point. All geometry in the unsteady flow data that was used to test this method was static. Dynamic objects could potentially cause problems due to the fact that the eyelet locations are fixed in space.

Helgeland and Elboth presented a texture-based method for visualizing 3D unsteady flow [62]. The first step of the algorithm is to inject evenly spaced particles into the flow domain and calculate their path lines. The initial particle seeding is based on Jobard and Lefer's algorithm [9]. At any given time step, these particles are essentially a sparse input texture. Sparse input textures don't suffer from the same occlusion issues that dense texture methods do when used on 3D datasets. Field line generation is then done at each time step with an LIC based method. The particle advection and field line generation steps are both separated from the volume rendering of the result in order to speed up the rendering process and allow interactive visualization.

2.2.2.3 Visualization Enhancments in 3D Unsteady Flows

Jänicke and Scheuermann presented a new way of visualizing unsteady vector fields by using ε-machines, which show a compressed representation of the data [63]. An ε-machine is essentially a finite state machine that is visualized with directed graphs. The ε-machine compresses the data down to its essentials in order to highlight important phenomena. A use can select a subset of the compressed version of the data in order to visualize the corresponding portion of the original data with more traditional methods such as flow lines and surfaces.

2.2.2.4 Integral Surfaces in 3D Unsteady Flows

Integral surfaces have also been used to visualize 3D unsteady flows. Schafhitzel et al. introduced a point-based method for generating and rendering stream and path surfaces [64]. The point-based approach relies on the massive integration of particles in

parallel on the GPU, thus avoiding the slow triangulation step used by other stream surface algorithms. Also, the point-based surface representation lends itself well to efficient rendering via splatting. They also demonstrated how texture-based methods can be applied to both stream and path surfaces in order to visualize the flow structure on the interior of the surfaces.

Garth et al. developed an algorithm for generating integral surfaces that is based on successive timeline approximation [65]. It is applicable on both steady and time dependent vector fields. They identify that many problems with previously proposed stream surface generation algorithms is the lack of separation between the surface approximation and the generation of a corresponding mesh that is suitable for rendering.

Von Funck et al. presented a method for interactively generating streak surfaces by skipping the typical adaptive remeshing step and hiding the artifacts with smoke rendering [66]. Unlike stream surfaces, all locations on a streak surface are updated every time step which means the entire resulting polygon mesh must be regenerated each time step as well. This is very computationally intense. Without adaptive remeshing, the mesh representing the streak surface will keep a relatively low triangle count, however some of the triangles will have a poor aspect ratio. These unfortunate triangles are hidden with opacity mapping and smoke rendering. They present comparative visualizations of smoke surfaces with significantly different triangle counts to show that the same smoke structures are visible.

McLoughlin et al. introduced an algorithm for generating stream and path surfaces that is fast and simple enough to use in practical visualization applications [67]. The authors speculate that the reason stream surfaces have not been more widely adopted

outside of visualization research applications is due to implementation complexity. In this algorithm, the stream or path surface is represented by a quad-based mesh. The quad mesh lends itself to being stored in a 2D array, which greatly simplifies the implementation. Their method handles convergence, divergence and rotation in the flow without distorting the stream surface.

Krishnan et al. proposed an efficient method of generating time and streak surfaces in large 4D vector fields [68], which allows for interactive exploration of the evolving surface. Their method is based on a decoupling of particle trajectory integration and surface construction, which allows for increased parallelism in the surface advection stage. They also presented an extension of generalized streak lines [52] to generalized streak surfaces.

Bürger et al. introduced the first algorithm for real-time adaptively refined streak surface integration and rendering [69]. Two approaches for accomplishing this are presented. The first approach is patch-based and the second is particle-based. Both schemes run entirely on the GPU using 3D texture maps to store the Cartesian grids of the vector field.

2.3 Vortex Detection in Vector Fields

Topological or feature-based visualization relies on detection and tracking of features in vector fields. For example, in flow visualization some common features include vortices, shock waves, recirculation, flow separation and flow attachment. While this research is centered on geometric integration-based visualization, feature detection is still useful for seed point generation. Of particular interest in my test data is vortex

detection and tracking. Vortex detection algorithms can be categorized in several ways. On categorization is based on whether they attempt to find regions in the flow that contain vortices or if they attempt to find vortex cores only. Another possible categorization is to group methods that work based on measurements at single points in the vector field and methods that work by measuring geometric constructs like streamlines or path lines in the vector field.

The simplest method of finding regions that contain vortices is to use metrics that can be applied to any point in the flow. For example one can look for areas with vorticity magnitude above a threshold value, however Zabusky showed that areas of high vorticity are not guaranteed to contain a vortex [70]. For example, shear flows can exhibit high vorticity at every point without containing a vortex. Villasenor and Vincent developed an algorithm that uses this approach to construct vortex tubes based on the average vorticity vectors found within cylinders [71]. Levy et al. used a similar approach except instead of vorticity they used the helicity of a flow to detect regions that contain vortices [72]. Another similar approach proposed by Robinson is to use areas of low pressure to locate vortices [73]. It is possible to have areas of low pressure that don't contain vortices however. Most of these methods work best with relatively simple flow data, however they can be useful in more complex flows when combined with other methods to help narrow the search area.

In order to identify the core of a vortex Banks and Singer presented a predictor-corrector method [74]. Seed points with high vorticity and low pressure are chosen. Streamlines which run along the vortex core are then traced by integrating along the vorticity field. Roth and Peikert proposed a method for vortex core detection that works

by identifying points where vorticity is parallel to velocity [75]. Sujudi and Haimes presented an algorithm for finding vortex cores by determining where the Jacobian matrix has one real eigenvector which is parallel to the velocity at that point [76]. Kenwright and Haimes used the eigenvector method to detect vortex cores, vortex bursts, spiral vortex breakdowns and vortex diffusion [77]. Roth and Peikert presented a novel vortex detection scheme for 3D vector fields. It is based on the eigenvector method and uses higher-order derivatives in order detect vortices with cores that are not straight [78].

Jiang et al. developed an algorithm to detect vortices that works by testing each grid cell to see if it belongs to a vortex core by examining the vectors at neighboring cells [79]. More recently, Jankun-Kelly et al. presented a feature-based vortex detection method for large vector fields represented by unstructured meshes [80]. They exploit the fact that local extrema in certain scalar fields coincide with vortex cores. They also use k-means clustering in order to handle complex vortex topologies like those that occur when two vortices merge.

Streamlines have also been used as a method of detecting vortices. Sadarjoen and Post presented two such methods which are based on measuring properties of 2D streamlines [81, 82]. This approach differs from the majority of previously presented vortex detection methods in that does not depend on point-based measurements taken on the vector field. Their first approach is based on measuring the curvature center of the oscillating circle on a streamline at many sample points. The second approach is to measure the winding-angle of streamlines in order to determine which streamlines are part of a vortex. This method is more powerful than point based approaches because it can find weak vortices. I believe there is more work to be done in this area by testing

other geometric stream objects and more global measurements on them in order to identify vortices.

2.4 Applications of Geometric Flow Visualization

One cannot fully determine the quality of a visualization method if it is only tested on very simple synthetic data. This is the motivation for testing my algorithms on the simulated dragonfly data set. There have been several past publications presenting the results of testing flow visualization methods on more complex real world inspired CFD simulations.

Laramee et al. presented a visual exploration of fluid flow through a cooling jacket [83]. They used a broad range of direct, geometric and texture-based visualization methods as well as automatic, semi-automatic and interactive feature extraction. Advantages and disadvantages of the various methods are compared in light of the application. Bauer et al. presented a case study of visualization in a time dependent 3D flow resulting from an industrial application [84]. The application was the rotating helix structure that builds in the draft tube of a wind turbine, known as a vortex rope. They employed a particle based approach to animated visualization such that particles are only visible in the region of interest and their density represents the density of the simulation medium.

You et al. used visualization to study vortex shedding and motion in a flow containing a flexible plate [85]. They present a series of iso-line visualizations arranged with respect to time or according to simulation settings. While this work did not utilize integration-based stream objects, the application domain was one of the few examples

that used a deformable plate in the flow field and has a lot in common with my dragonfly simulation data sets.

Flow visualization studies have also been done in several biomedical applications. Soni et al. performed a visual exploration of air flow in the small bronchial tubes [86]. Secondary flows within the bronchial tubes are dominated by vortices, which influence particle deposition. Streamline based visualization of the flow through the bronchial tubes was combined with dense visualizations on slices perpendicular to the tubes in order to show secondary flow at the same time. In a similar application He et al. used streamlines to visualize blood flow in a hemodynamic simulation through a cerebral artery [87]. A patient-specific model of a cerebral artery with an aneurysm was constructed. Visualizations were done with vector glyphs and stream tubes with velocity mapped to the surface color in order to show how flow slowed in the aneurysm.

2.5 Insect Flight Simulation

The generalized streak line seeding and flowing seed point algorithms explained later in this document are ultimately intended to improve visualization of insect flight simulations. Insect flight seems impossible based on traditional notions of aerodynamics and physicists, aerospace engineers, zoologists and biologists have been puzzled for years over how these tiny creatures can fly. Insect wings are essentially deformable flapping airfoils, which bend and twist during flight allowing them to make a variety of quick flight maneuvers. For instance, Wootton showed that insects can accelerate a precise mass of air directly downward in order to hover, or obliquely to fly in any given direction [88].

Little attention has been placed on understanding the flight mechanisms employed by airborne objects that operate at ultra-low Reynolds numbers in the 100 to 10,000 range. It is in this ultra-low Reynolds number range where insects have evolved into the undisputed masters of flapping wing flight in the micro-aerial world. For instance, dragonflies operate at a Reynolds number near 10,000, where viscous effects are dominant and cannot be ignored like they can when studying objects that operate at high Reynolds numbers like airplanes. Insects have incorporated hovering, maneuvering and gliding skills into their flight repertoire, but it is still not completely clear how they can fly at all, but perhaps the use of new flow visualization methods can help clarify it.

2.5.1 Flapping Flight Theory

Insects flap their wings to generate lift and drive. In order to fly, the thrust produced must overcome the drag generated from moving through the air and the lift produced must overcome the insect's weight. An extremely unsteady flow field is the result of the constantly flapping insect wings and it is not entirely clear how the process works. Several theories as to how insects generate the required lift and propulsion to fly have been proposed with the most notable being the clap-and-fling [89], quasi-steady aerodynamics [90] and the leading edge vortex hypothesis [91].

In the clap-and-fling mechanism proposed by Weis-Fogh [89], the wings clap together above the insect's body and then fling apart. During the fling air gets sucked in and creates a vortex over each wing. These vortices move across the wings and act as the starting vortices for the opposite wings during the next clap. Thus circulation occurs and lift is increased enough to explain the lift that certain insects generate, however Marden

showed that most insects do not use the clap [92] so it clearly cannot explain the high light coefficients that all insects produce.

Quasi-steady aerodynamic theory was first proposed by Ellington [90] to reduce the complex unsteady nature of flapping flight to a simplified model. The basic theory is that instantaneous forces on the wing are equivalent to steady state forces at the same angle of attack and instantaneous velocity. In other words, the air flow over the wing at any given time is the same as how the flow would be over a non-flapping, steady-state wing. The potential effects the fluid from previous points in time has on the next static position is ignored. However, multiple scientists have demonstrated that quasi-steady aerodynamic theory alone cannot account for enough lift force to counter an insect's weight [90, 93-96]. This means that some unsteady phenomena is providing additional aerodynamic forces, which is why unsteady flow visualization is essential to this application.

2.5.1.1 The Leading Edge Vortex

Due to the fact that clap-and-flip and quasi-steady aerodynamic theories were unable to correctly account for lift in flapping flight, attention has shifted to the leading edge vortex (LEV). The theory is that fluid separates from the sharp leading edge of an airfoil and is then drawn into a vortex, which creates a low pressure region above the wing resulting in higher lift production. The total lift force develops normal to the surface of the wing and consists of a vortex lift term and a potential flow term.

For blunt airfoils at an angle of attack, fluid moves around the leading edge which produces a suction force parallel to the airfoil’s chord. Lift is calculated by adding the

suction force to the normal force. At higher angles of attack the LEV grows to the point where it can no longer remain attached to the foil. A trailing edge vortex begins to form and moves towards the leading edge resulting in vortex shedding from the leading and trailing edges. This vortex shedding causes a loss of lift, however the presence of a large LEV just before it separates results in a temporary high lift coefficient. Many researchers suspect that insects take advantage of this phenomenon to generate such high lift forces. Maxworthy demonstrated that the leading edge vortex could be a source of lift in flapping flight by means of a mechanical flapping model [97].

Studies of the LEV have produced different results in 2D and 3D. Dickinson and Götz showed that in 2D simulations of an airfoil moving with a high angle of attack, the LEV grows until flow reattachment can no longer occur. A trailing edge vortex forms and is shed into the wake followed by the leading edge vortex, at which time a new LEV begins to form [98]. Sane demonstrated that lift production in 2D air foil studies is time dependent with the maximum lift occurring just before the LEV is shed [99]. The angle of attack used in this study was well below the maximum utilized by insects.

Three dimensional studies of flapping wings differ in that the LEV grows to a certain point and then remains attached during the down stroke [100]. In this experiment Ellington et al. observed 3D smoke trails flowing around the flapping wings of a hawkmoth tethered to a stand in a wind tunnel. The LEV could be observed in the smoke streak lines as it increased in size while moving towards the wing tips via axial flow during the down stroke. This study found that the lift force accounted for 1.5 times the weight of the hawkmoth. Berg and Ellington's experiments on mechanical flapping foils [101] and Willmott's work with tethered hawkmoths [96] have also identified the

dynamic stall that occurs when the angle of attack is rapidly changed as a way to generate the kind of lift needed for insect flight.

While it is clear that the LEV is capable of generating the necessary lift for insect flight, it is not clear why it grows to a certain size and then stabilizes in the 3D case but not the 2D case. A similar experiment involving a fruit fly wing contradicted the results of the hawkmoth experiment. The fruit fly study or Birch and Dickinson [102] was performed with a Reynolds number of 160 and showed no signs of a helical vortex. Additional experiments showed that lift is generated from the LEV itself and not from its stable attachment to the wing throughout the stroke.

In addition to the contributions of the LEV, there are additional lift sources present in flapping wing insect flight. For example Dickinson showed that the wing rotation during the down stroke to up stroke transition provides additional lift [103]. In addition Dickinson et al. concluded that delayed stall, rotational circulation as well as wake capture all contributed to lift production [104]. Wake capture refers to the phenomena where vortices are shed into the flow field and then later encountered by the same or a different wing after the stroke reversal. Wake capture allows insects to recover energy that was lost during the previous stroke and it is of particular interest in the study of quad wing insects.

2.5.2 Visualization Implications of the LEV

Continuing research is still being conducted on the formation and stabilization of the leading edge vortex. The formation and attachment of the LEV is thought to be governed by the Reynolds number, because spiral flow along the wing span was observed

by Birch et al. only at high Reynolds numbers [105]. Yet, another experimental study by Adrian et al. suggested that span wise flow is not necessary for the stabilization of the LEV, and the growth, formation, and stabilization of the LEV is associated with the rapid increase in angle of attack and its shedding is caused by a decrease in the angle of attack [106].

All these variations in results suggest that this area of research needs more attention to accurately define and understand the mechanisms responsible for the high lift production via the leading edge vortex. Although researchers in the bio-fluid field [107-109] agree with the LEV being the main source of lift production for insects, understanding the unsteady nature of flapping airfoils is the first step in understanding the nature of insect flight [110]. Thus, the main goal of visualizing insect flight simulations should be to highlight how the wing kinematics affect the formation of the LEV. If a visualization either occludes the behavior of the LEV or does not properly emphasize it relative to other flow features it will decrease the visualization quality. At the same time it is also desierable for visualizations to capture any wing wake interactions, trailing edge vortex formation and rotational circulation at stroke reversal.

3 Method

This section presents a theoretical background and overview of several novel visualization techniques. First, the theory behind streamlines, path lines, timelines and streak lines is introduced because this forms a basis for the new methods. Next, the flowing seed point algorithm is introduced. Visual examples of how flowing seeds work with a variety of flow lines are provided. Then, the dynamic seed curve concept is introduced. Several methods of generating dynamic seed curves based on both the underlying vector fields and on the moving objects in the flow field are presented. The majority of the examples in this section use the flapping disk data set. More information about the creation of this data set can be found in section 4, and the results of applying the techniques described in this section to a more complex data set can be found in section 5.

3.1 Streamlines

Streamline based geometric visualization constructs were chosen as the basis for the flow visualization research in this document. Streamlines are essentially trajectories in a time independent vector field that are computed over time intervals. They are tangent to the vector field at all points and are computed by integrating over a vector field $\boldsymbol{v}$ for some time interval when starting at a seed point p_0. In a steady flow, the time t represents integration time, not physical time, thus it is constrained only by speed and numerical precision. The following differential equation describes a streamline $\boldsymbol{S}$ and Figure 3.1 shows an example of several simple streamlines.

$$\frac{d\boldsymbol{S}_{t_0,p_0}(t)}{dt} = \boldsymbol{v}(\boldsymbol{S}_{t_0,p_0}(t)) \tag{3.1}$$

$$\boldsymbol{S}_{t_0,p_0}(t_0) = p_0 \tag{3.2}$$

The streamline is then obtained by solving the differential equation step by step:

$$\boldsymbol{S}_{t_0,p_0}(t+\Delta t) = \boldsymbol{S}_{t_0,p_0}(t) + \int_t^{t+\Delta t} \boldsymbol{v}(\boldsymbol{S}_{t_0,p_0}(t))dt \tag{3.3}$$

The fourth order Runge-Kutta method is used to find a numerical solution as follows:

$$\Delta p_0 = \Delta t \cdot \boldsymbol{v}\left(\boldsymbol{S}_{t_0,p_0}(t)\right) \tag{3.4}$$

$$\Delta p_1 = \Delta t \cdot \boldsymbol{v}(\boldsymbol{S}_{t_0,p_0}(t) + \Delta p_0/2) \tag{3.5}$$

$$\Delta p_2 = \Delta t \cdot \boldsymbol{v}(\boldsymbol{S}_{t_0,p_0}(t) + \Delta p_1/2) \tag{3.6}$$

$$\Delta p_3 = \Delta t \cdot \boldsymbol{v}(\boldsymbol{S}_{t_0,p_0}(t) + \Delta p_2) \tag{3.7}$$

$$\boldsymbol{S}_{t_0,p_0}(t+\Delta t) = \boldsymbol{S}_{t_0,p_0}(t) + \frac{\Delta p_0}{6} + \frac{\Delta p_1}{3} + \frac{\Delta p_2}{3} + \frac{\Delta p_3}{6} \tag{3.8}$$

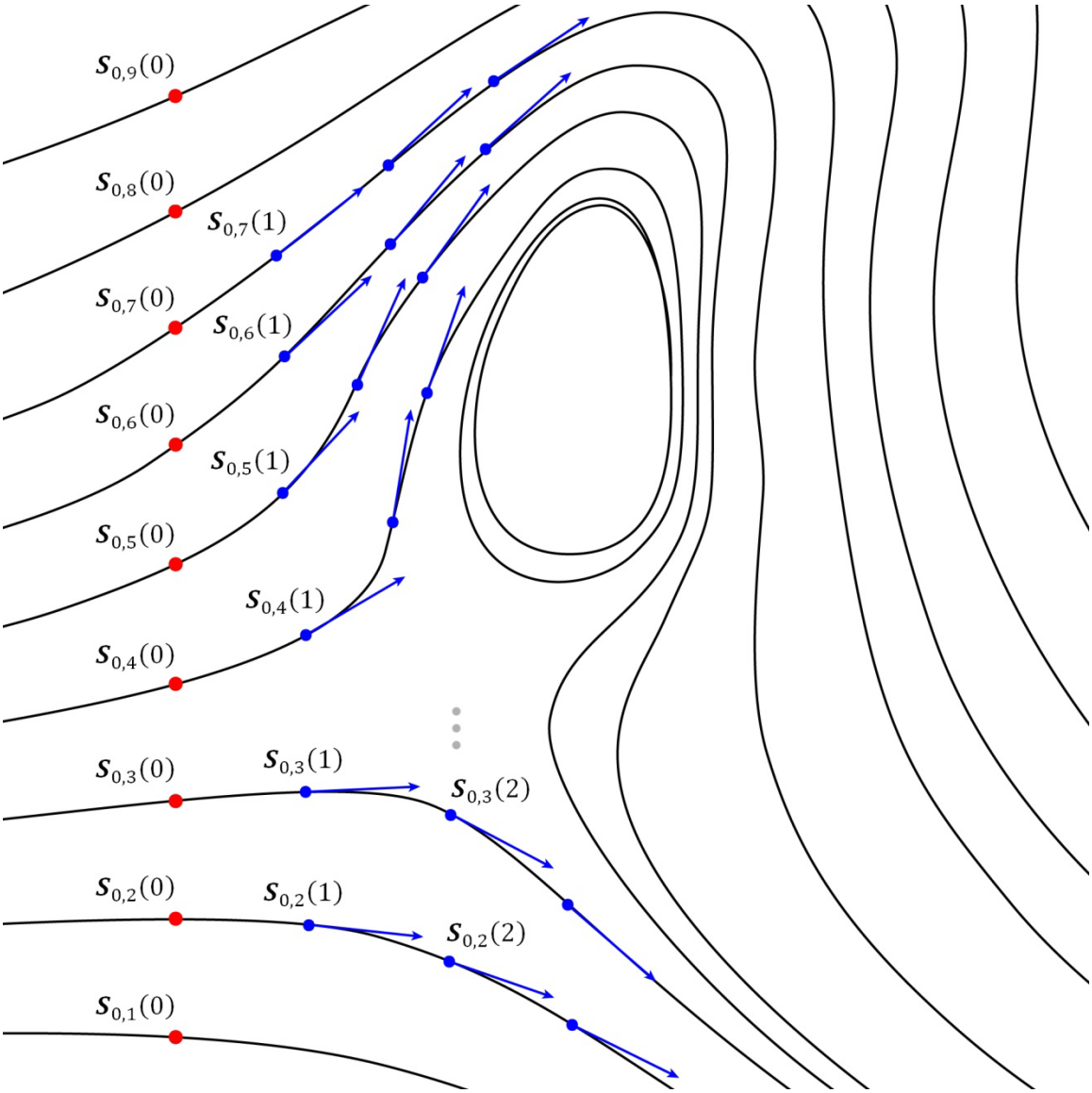

Figure 3.1: Several streamlines in a 2D vector field that contains one vortex.

3.2 Path Lines, Timelines and Streak Lines

Path lines are logically similar to streamlines, but they are used to view unsteady flows by tracing the path of massless particles over multiple time steps. Since the flow is time-dependent, one must integrate over multiple time steps of the vector field to compute a path line. The trajectory of a path line $\boldsymbol{P}_{t_0,p_0}(t)$, where t_0 is the seed time, p_0

is the seed point and t is now both the simulation time and integration time, can be determined by solving the following differential equation. This concept is illustrated in Figure 3.2.

$$\frac{d\boldsymbol{P}_{t_0,p_0}(t)}{dt} = \boldsymbol{v}(\boldsymbol{P}_{t_0,p_0}(t), t) \tag{3.9}$$

$$\boldsymbol{P}_{t_0,p_0}(t_0) = p_0 \tag{3.10}$$

Integrating yields:

$$\boldsymbol{P}_{t_0,p_0}(t+\Delta t) = \boldsymbol{P}_{t_0,p_0}(t) + \int_t^{t+\Delta t} \boldsymbol{v}(\boldsymbol{P}_{t_0,p_0}(t), t)dt \tag{3.11}$$

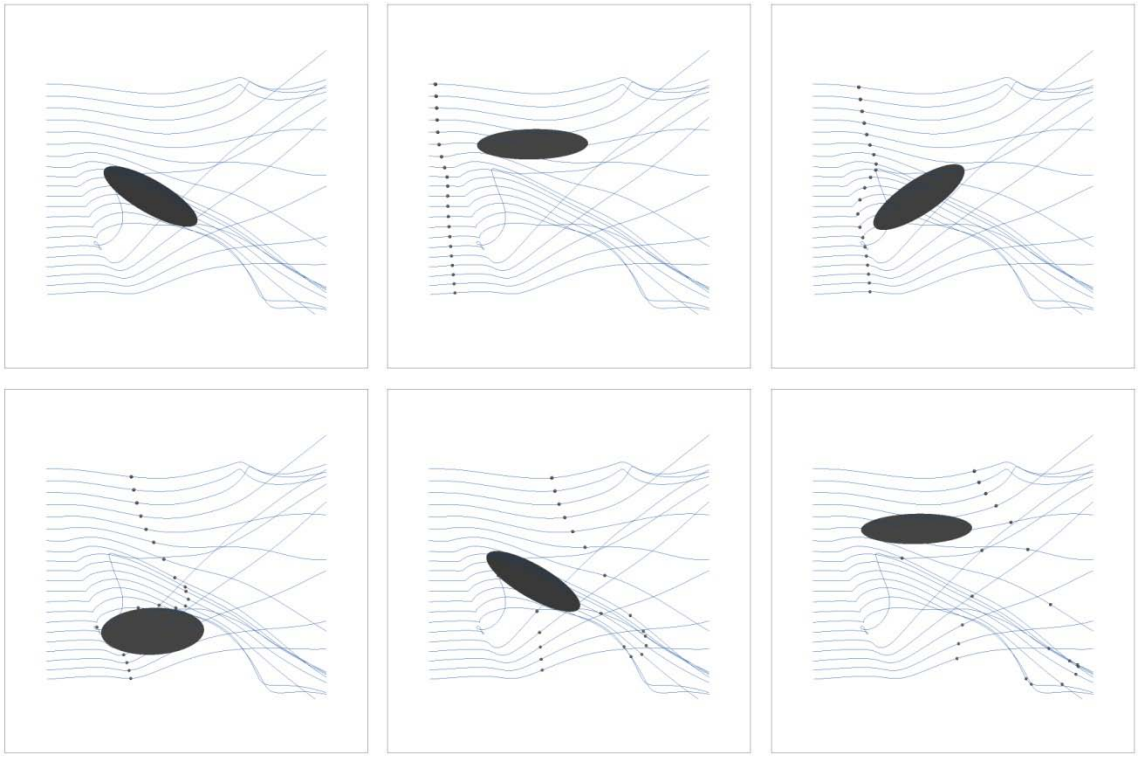

Figure 3.2: Several path lines and the corresponding particles at time steps 0, 200, 400, 600, 800 and 1000 of the flapping disk data set.

Streak lines and timelines are easy to define based on the previous definition of a path line. A streak line is generated by connecting a series of particles that were released into the flow from the same seed point (Figure 3.3). In real world applications this is done by injecting dye or smoke into a flow. A streak line $\boldsymbol{K}_{t_0,p_0}(t)$ is defined based on the definition of a path line, as shown in the following equation.

$$\boldsymbol{K}_{t,p_0}(t_0) = \boldsymbol{P}_{t_0,p_0}(t) \tag{3.12}$$

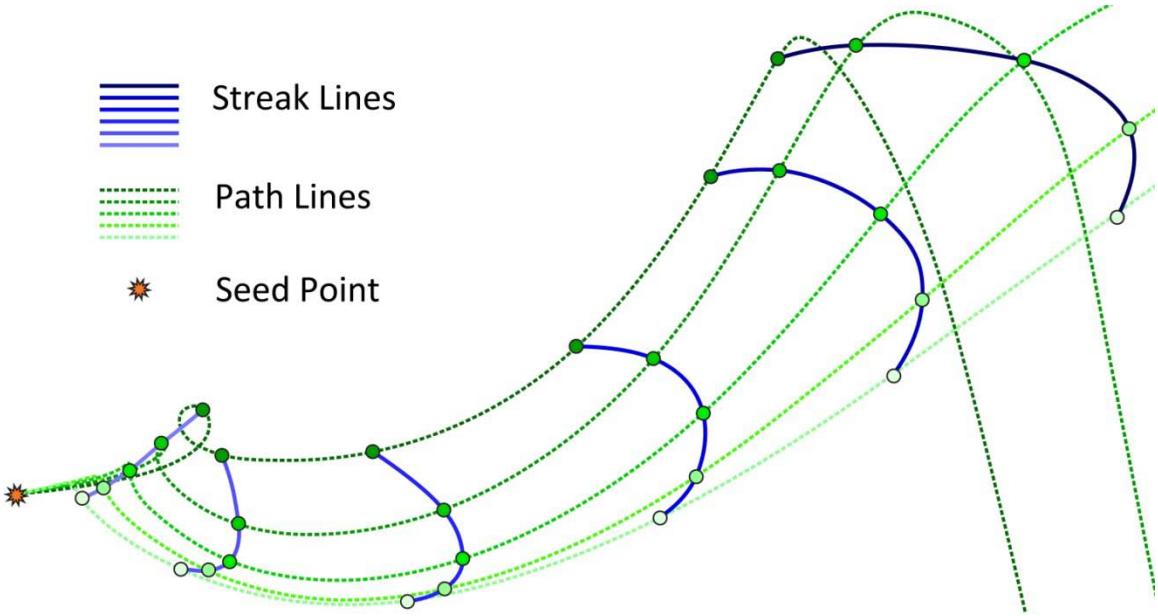

Figure 3.3: Evolution of a streak line at 6 time steps (blue) and the corresponding path lines (green).

Wiebel et al. proposed the concept of a generalized streak line [52]. A generalized streak line $\boldsymbol{G}_{t,p_{ij}}(t_0)$ is a line connecting all particles which were released at consecutive time steps from a seed location $p_{ij}(t_0)$ as it moves along a seed curve p_{ij}. Generalized streak lines will be used later in this document to keep particles near the leading edge vortex of a flapping wing as it moves throughout the flow domain. Figure

3.4 shows a conceptual description of generalized streak lines and Figure 3.5 shows several time steps of a generalized streak line generated with the flapping disk data set.

$$\boldsymbol{G}_{t,p_{ij}}(t_0) = \boldsymbol{P}_{t_0,p_{ij}(t_0)}(t) \tag{3.13}$$

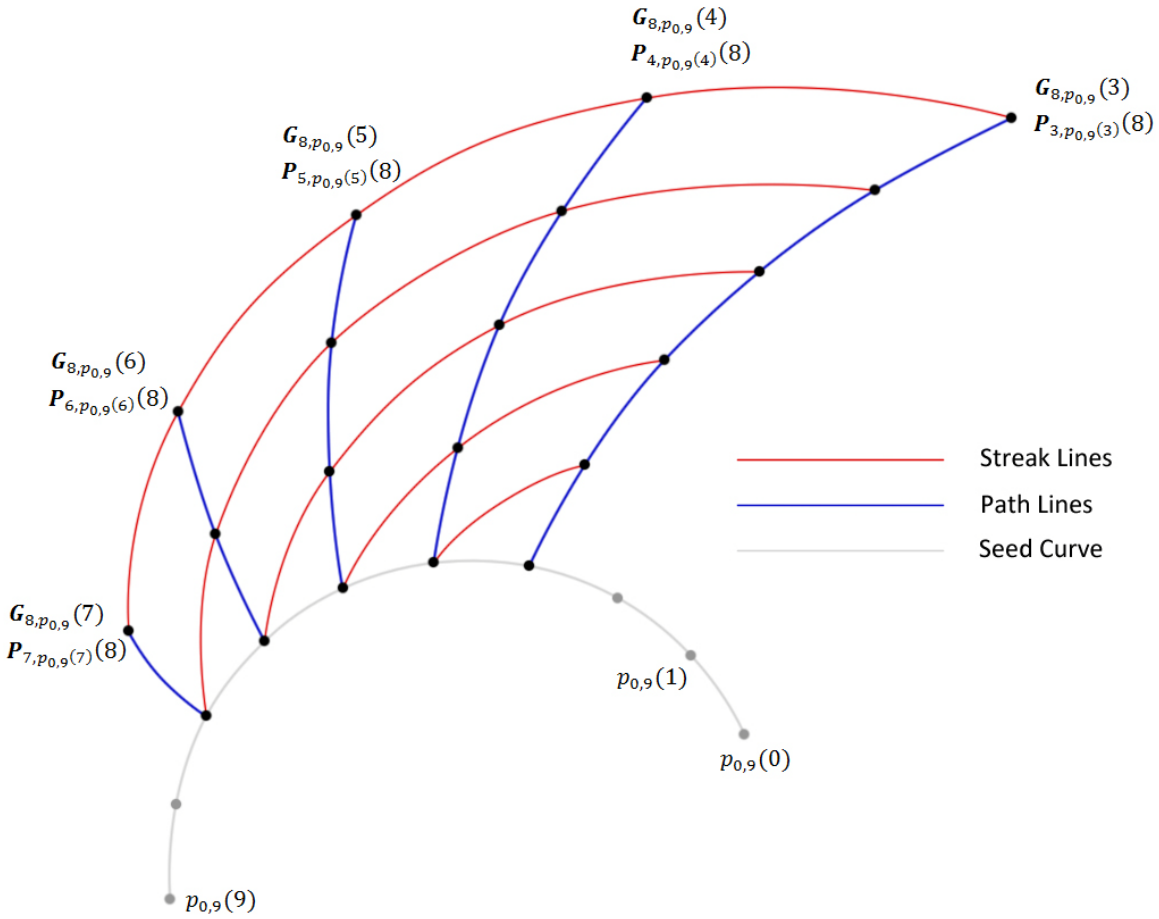

Figure 3.4: Conceptual illustration of a series of generalized streak lines.

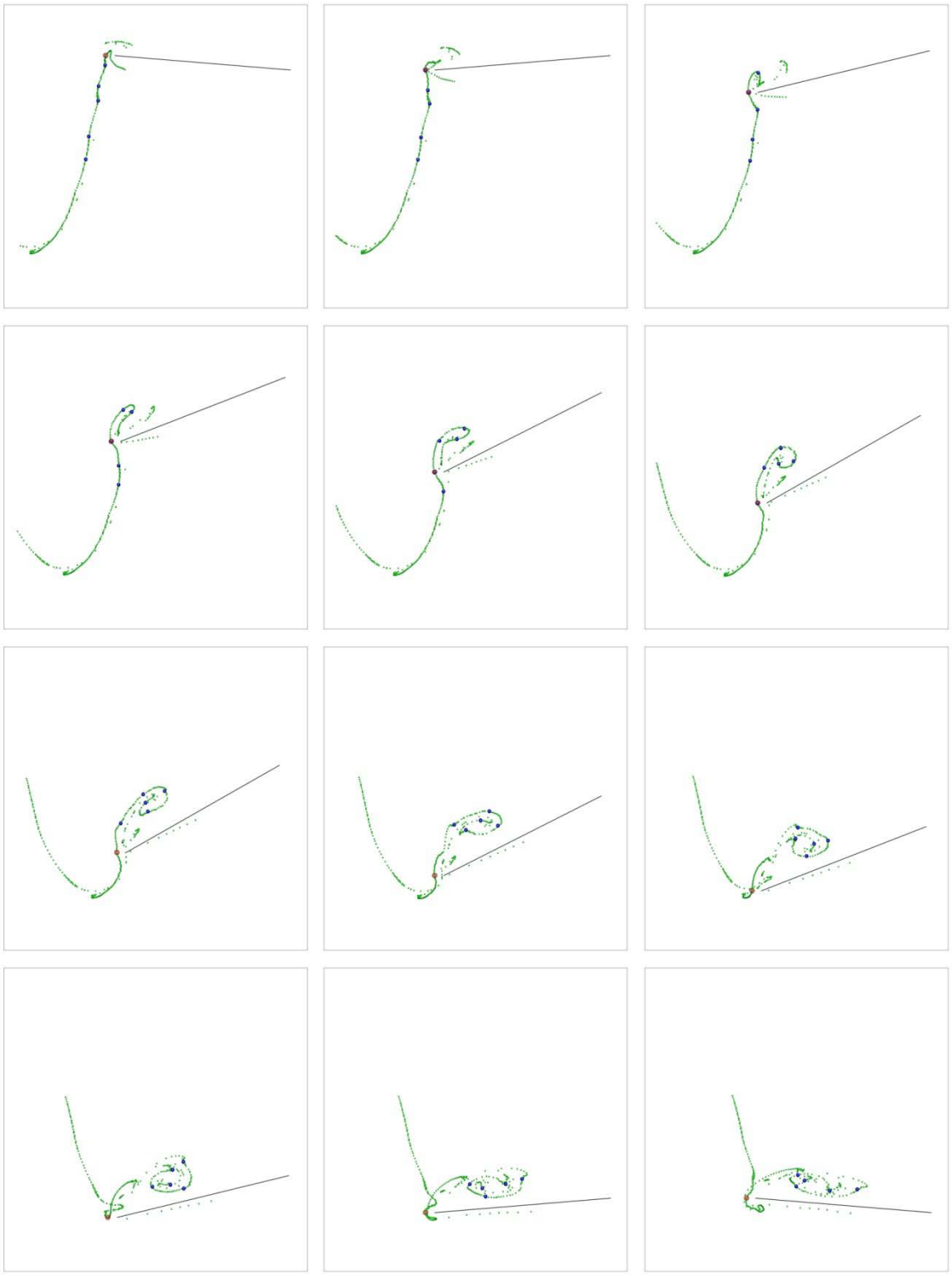

Figure 3.5: A generalized streak line in the flapping disk data set at 12 time steps starting at time 180 and proceeding at 40 time step increments. The red dot is the moving seed point which all particles pass through. The five blue particles pass the seed at time steps 220, 260, 300, 340 and 380.

Conceptually timelines are lines connecting a series of particles that were all released into the flow at the same time. The particles are aligned along a seed curve p when emitted into the flow. The seed curve is typically a straight line, but this is not essential. A definition of a timeline $\boldsymbol{T}_{t_0,p}(t)$ is provided in the following equation by relating it to the path line definition and a visual example is provided in Figure 3.6. Throughout the remainder of this document the term "flow lines" will be used to refer to streamlines, path lines, timelines, streak lines and generalized streak lines simultaneously.

$$\boldsymbol{T}_{t_0,p}(t) = \boldsymbol{P}_{t_0,p(t_0)}(t) \tag{3.14}$$

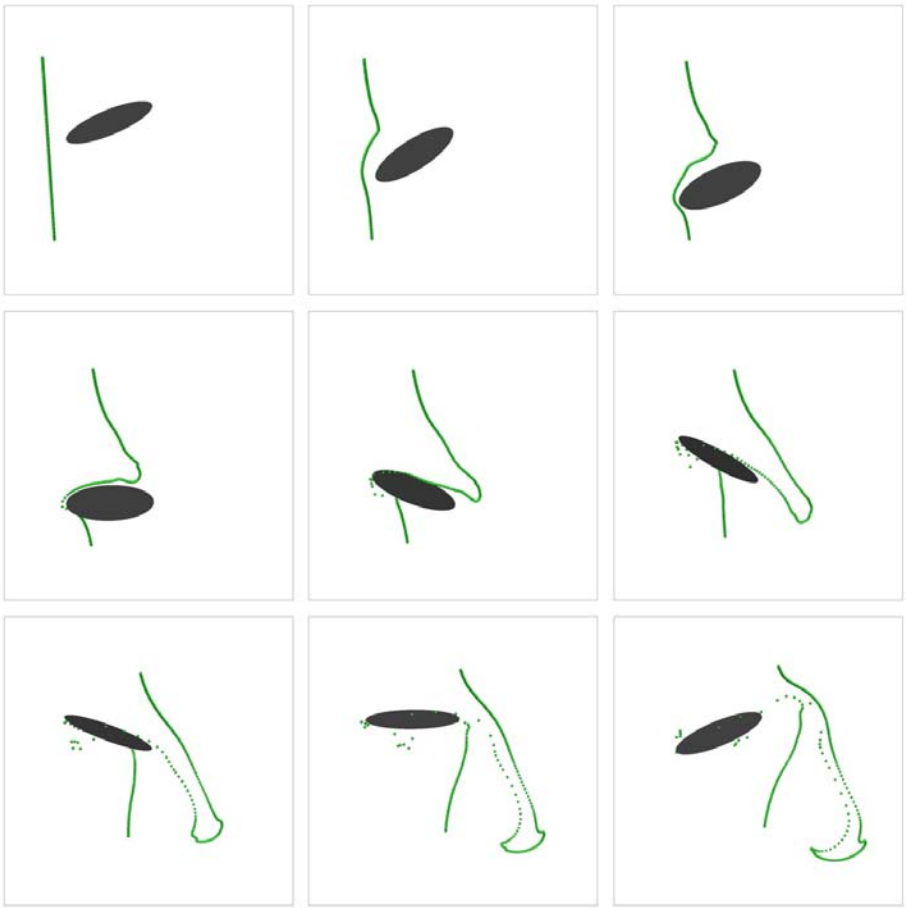

Figure 3.6: A single timeline moving through the flapping disk data set at time steps 300, 400, 500, 600, 700, 800, 900, 1000 and 1100.

3.3 Flowing Seed Points

Clearly streak lines and timelines lend themselves to animation whereas streamlines and path lines do not. In the case of streamlines, slight changes in the vector field between time steps can lead to very large changes in the streamline trajectory, resulting in a choppy animation. Despite this problem, it has been shown that for many data sets, short streamlines or streamlets can be animated relatively smoothly [62].

Path lines with constant seeds on the other hand have temporal data built in so it does not make sense for them to move over time. One possible way to animate path lines is to gradually move the seed point in either space, time or both, but this suffers from the same choppy animation problems encountered when animating streamlines. Another more effective way of animating path lines is to only show a small portion of the path line in the immediate vicinity of the corresponding massless particle as it moves in time.

On the other hand, while they are easier to animate, streak lines and timelines do not posses all of the perceptual benefits inherent to streamlines and path lines. Neither is instantaneously tangent to the flow, and they offer little contextual information about individual particle trajectories. Also, streak lines can potentially be perceptually misleading in that the human brain may think that the flow is moving tangent to the streak line, when this is rarely the case. Both streak line and time line animations also suffer if the animation is paused.

This section describes a novel method, known as flowing seed points, which combines multiple flow line variants into one smoothly animated visualization such that the benefits of each are achieved simultaneously. The flowing seed point visualization

approach has the following advantages when compared to traditional geometric visualization methods:

- Improved visualization of vortex formation and breakdown
- Temporally smooth animation of streamline and path line evolution
- Visualization of instantaneous and temporal divergence
- Ability to handle moving objects in the flow field
- Visualization of instantaneous and time varying velocity
- Comparative visualization of vector fields at neighboring time steps

3.3.1 Flow Lines as Seed Curves

In the previous definitions of timelines and generalized streak lines, the concept of a seed curve was used to describe groups of particles released into the flow at the same time as well as particles emitted from one moving seed point at different times. These seed curves can be either user defined or automatically generated.

There is no reason why seed curves can't also be used to seed inherently static flow lines like streamlines and path lines. There is also no reason why a seed curve cannot dynamically evolve over time. When the seed curve used to place streamlines and path lines evolves over time, the result is an animation. This is susceptible to all of the previously mentioned animation problems, which is why only very short streamlets and pathlets are used. Multiple seed curves are discretized into many distinct seed points to maintain a high level of coverage despite the low number of integration steps used in each streamlet or pathlet. While any static or dynamic seed curve could be used to seed a

series of streamlets and pathlets, there are many perceptual benefits from using either a timeline, streak line or generalized streak line as the seed curve. This concept, known as flowing seed points, is illustrated in Figure 3.7.

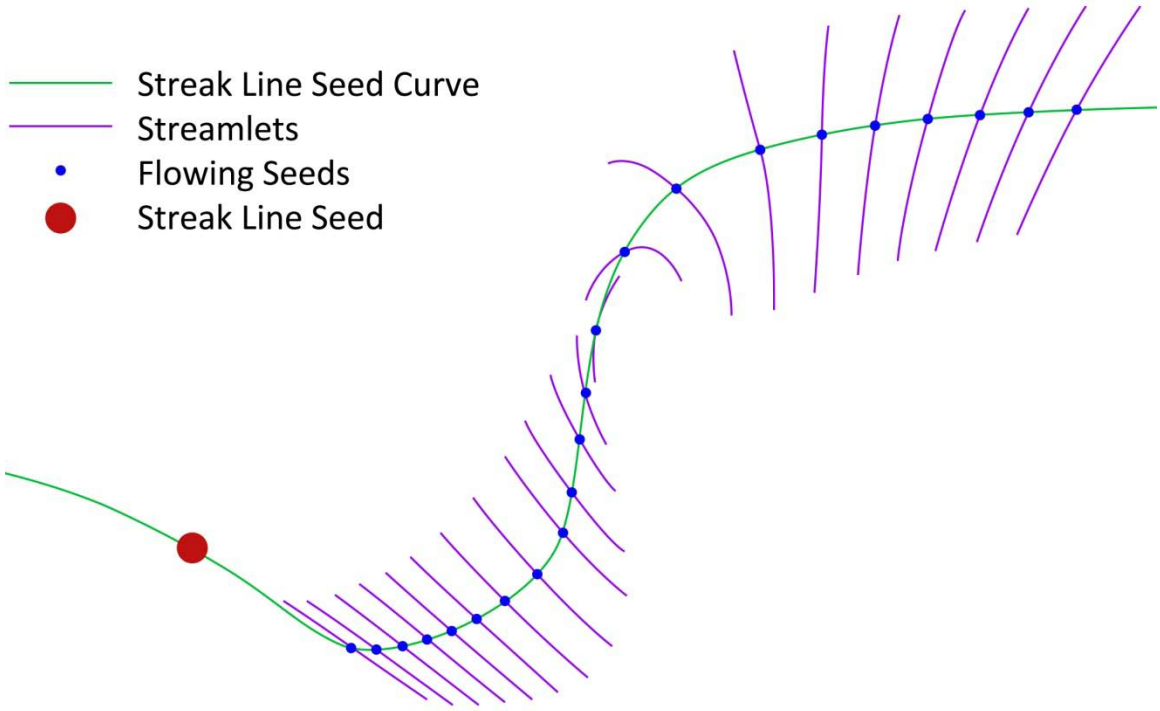

Figure 3.7: Flowing seed point method used to generate streamlets along a streak line.

3.3.1.1 Streamlets, Pathlets and Flowing Seeds

Streamlets offer many perceptual benefits when combined with flowing seed points. The main benefit is that one can see the evolution of the instantaneous state of the vector field within the context of the seed curve while at the same time visualizing the massless particle trajectories. This allows for a good streamline coverage while maintaining spatial coherence in the animation while at the same time maintaining all the benefits of the underlying seed curve. Pathlets on the other hand make it easier to

visualize how the particles within the flowing seed curve are diverging or converging. The animation can be paused for a more detailed look at some feature and the pathlets still maintain some contextual information about the neighboring time steps. Figure 3.8 shows the results of both streamlets and pathlets with flowing seed points near a vortex in the flapping disk data set.

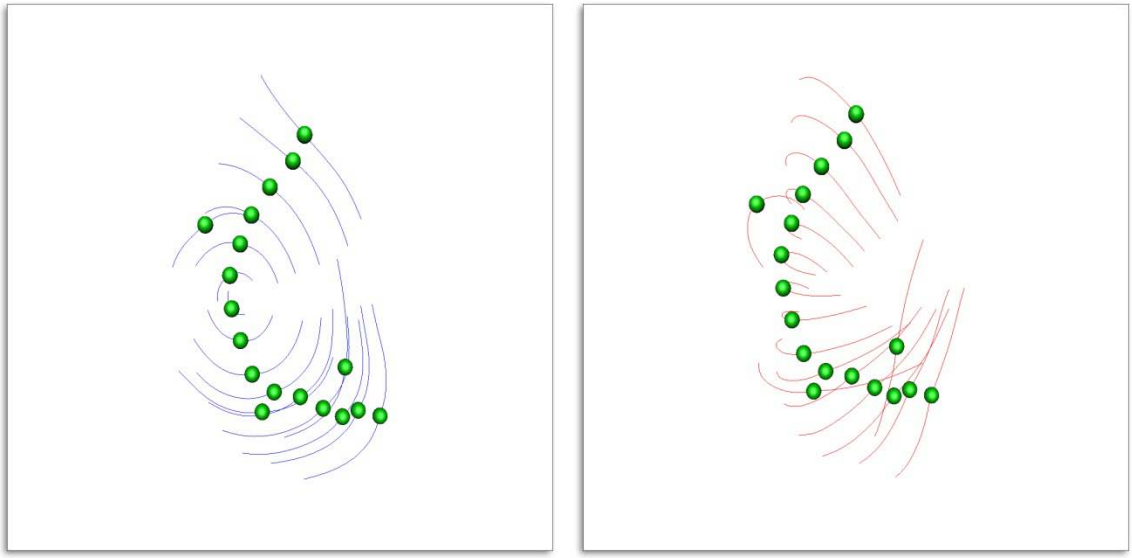

Figure 3.8: Comparison of streamlets and pathlets with the same flowing seed curve: (a) streamlets, (b) pathlets.

3.3.2 Time Seeds

Timelines are one option to use as a flowing seed curve. This concept is illustrated at several time steps in Figure 3.9. The shape of the streamlets at the earlier time steps allows a user to predict the timeline shape at later time steps. Figure 3.10 shows a comparison of the same timeline with and without flowing seeds. It is easy to

see how the lengths of the streamlets seeded along the timeline tell you things about the velocity of the vector field at that instant, which cannot be seen without them.

Figure 3.9: Time seeds at several time steps in the flapping disk data set.

Timelines are typically more useful with simulations that contain an inlet flow. Also, they aren't particularly well suited to the flapping wing data sets that this research is focused on visualizing. For these reasons the majority of the flowing seed point tests performed throughout the remainder of this document use streak lines or generalized streak lines for the seed curve.

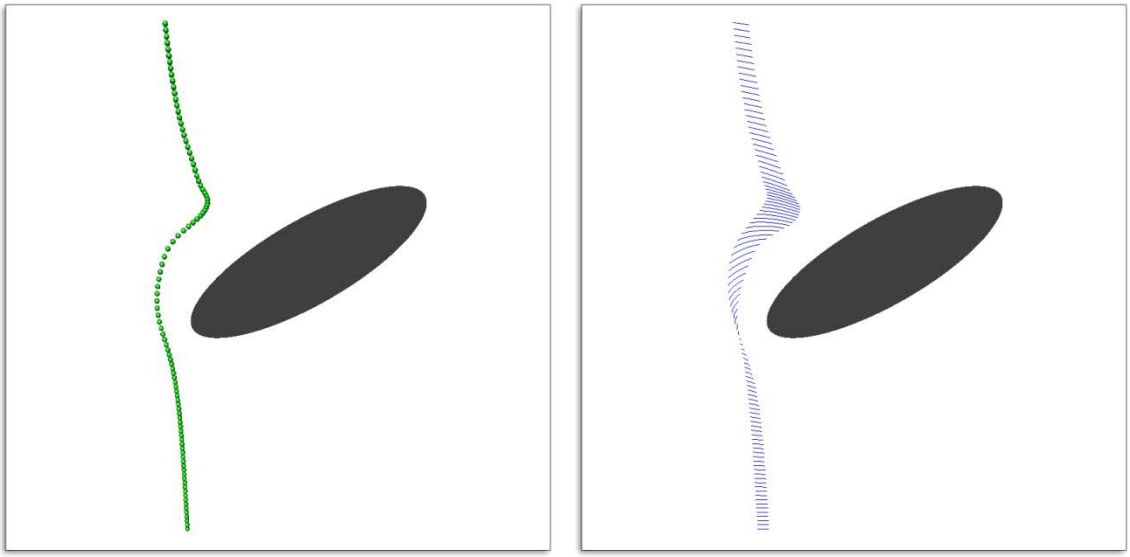

Figure 3.10: Comparison of a timeline with and without flowing seeds: (a) particles, (b) flowing seeds.

3.3.3 Streak Seeds

Streak lines are another option to use as a flowing seed curve. Figure 3.11 shows an example of this. One perceptual problem with streak lines is that a user may believe they are instantaneously tangent to the vector field, when this is rarely the case. Clearly the streamlets seeded along the streak line visually convey when the instantaneous vector field is approximately tangent to the streak line and when it is not. It shows that in many cases the streak line is closer to being perpendicular to the instantaneous vector field trajectory.

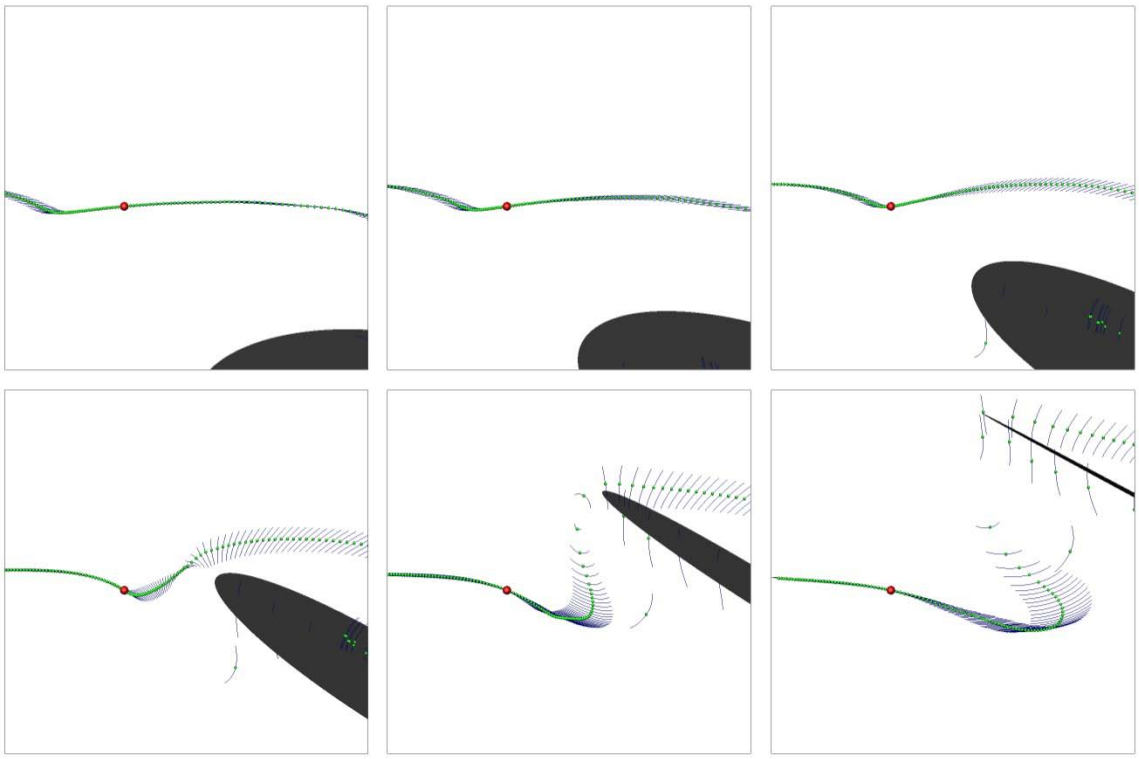

Figure 3.11: Streak seeds at several time steps in the flapping disk data set.

While streak seeds make more sense than time seeds to study the flapping wing data sets, they are still not ideal. This is due to the fact that streak lines have static seed points themselves. This means more seed curves are needed to capture moving vortices. This can be seen in Figure 3.12 where six streak line seed curves are used. In this figure the flowing seeds begin to capture the leading edge vortex as the disk moves downward, but there are many wasted seeds in less important areas. It has already been demonstrated that generalized streak lines are most effective at getting the majority of the particles in the vicinity of the flapping wing induced flow phenomena (Figure 3.5), so that is the flowing seed curve the remainder of this research will focus on.

Figure 3.12: Six streak line seed curves begin to capture the leading edge vortex with many unnecessary particles.

3.3.4 Generalized Streak Seeds

When using the flowing seed point method with flapping wing data sets, generalized streak lines are the ideal seed curve because they allow the particle emission point to move with the key flow features, which in turn moves with the flapping wings. In the case of flapping flight simulations, the particle emission point can move with the leading edge of the wing. When all flowing seeds are emitted in close proximity of the leading edge vortex core and not traced very far in time before being removed, coverage is maximized while self occlusion is held to a minimum. An example of this can be seen in Figure 3.13.

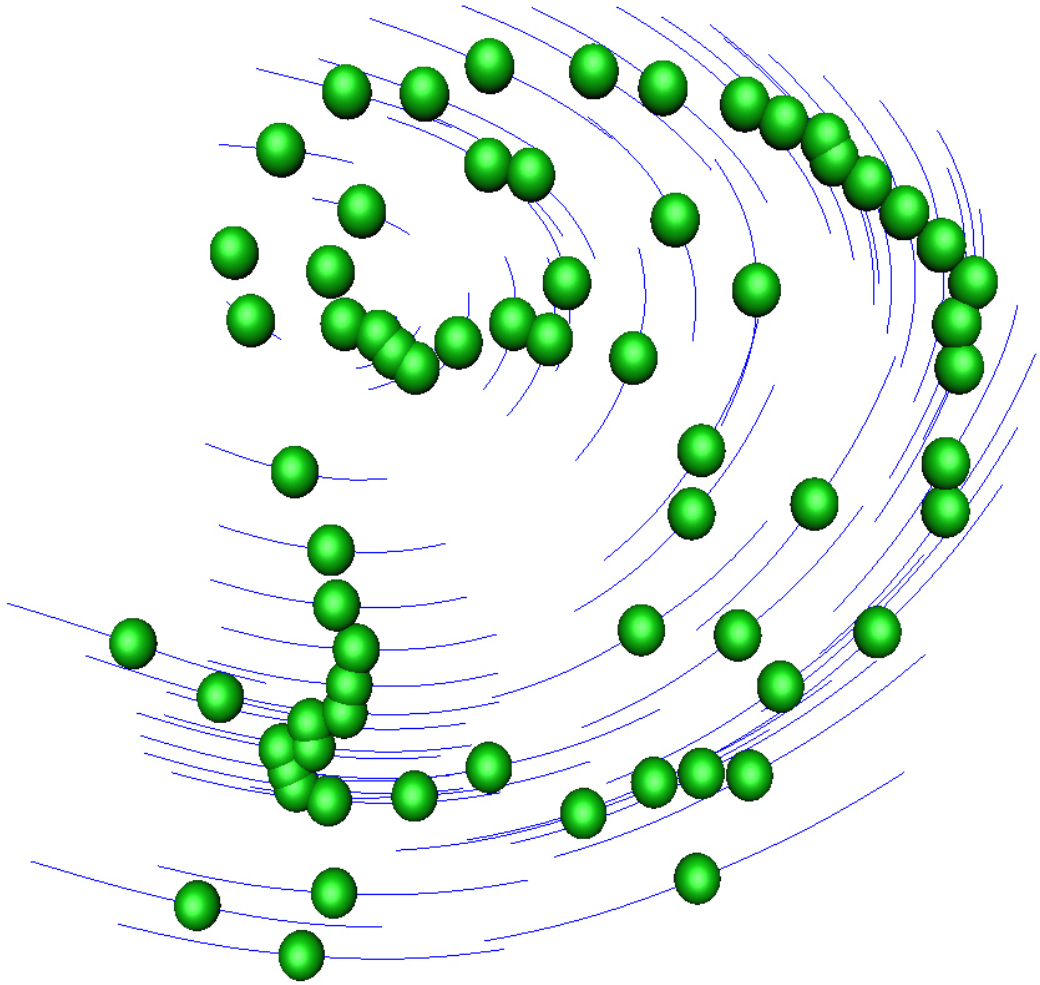

Figure 3.13: Several particles with the same moving seed point used to seed streamlets in a vortex.

Generalized streak lines and flowing seeds are particularly useful for visualizing vortices that are only present in a flow for a very short period of time. This is illustrated in Figure 3.14. The top row of images in this figure shows the state of the streamlets right before the formation of the vortex, during the existence of the vortex and right as the vortex begins the break down. When looking at the particles alone it is unclear exactly when the vortex forms and how long it lasts.

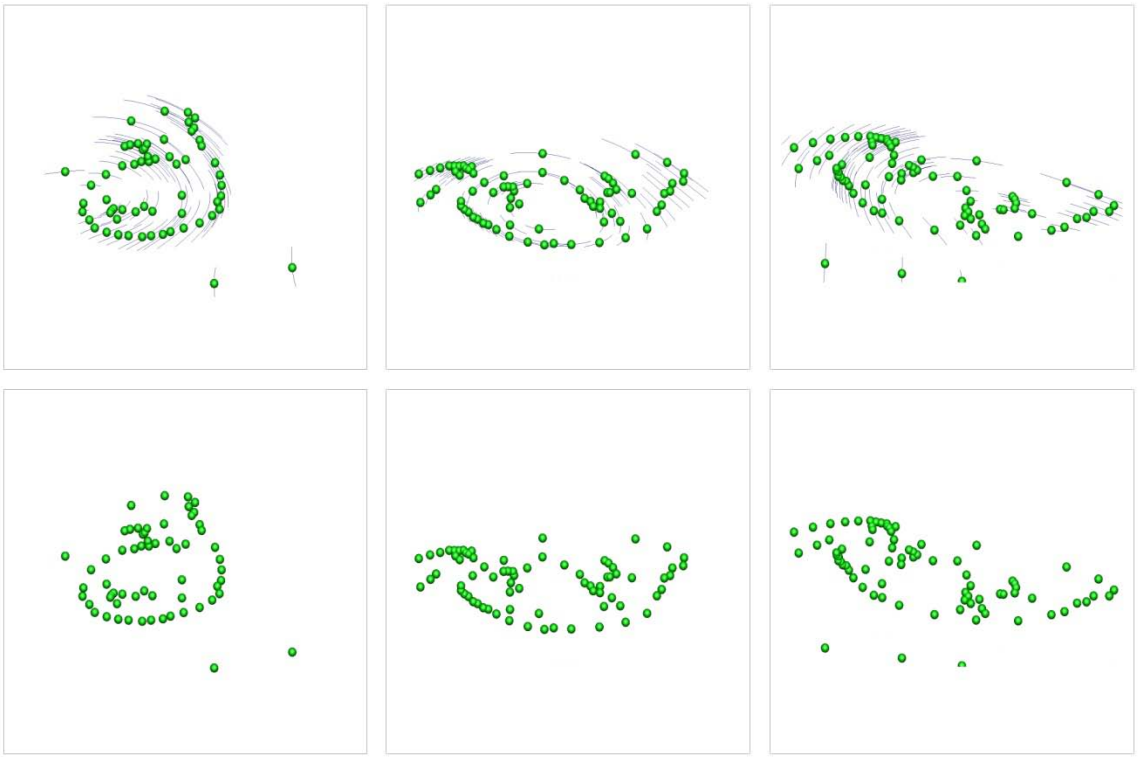

Figure 3.14: Vortex formation and breakdown captured with generalized streak seeds (first row) and the corresponding particles without streamlets (second row).

My method yields still images that reveal the instantaneous flow structure instead of just approximating it. Thus, each time step of the animation can be easily interpreted compared to single time steps of typical 3D time dependent flow animation techniques like streak lines or particle advection along path lines. The flowing seed point method can be further enhanced by using streamlet tubes and color mapping scalar quantities like vorticity (Figure 3.15).

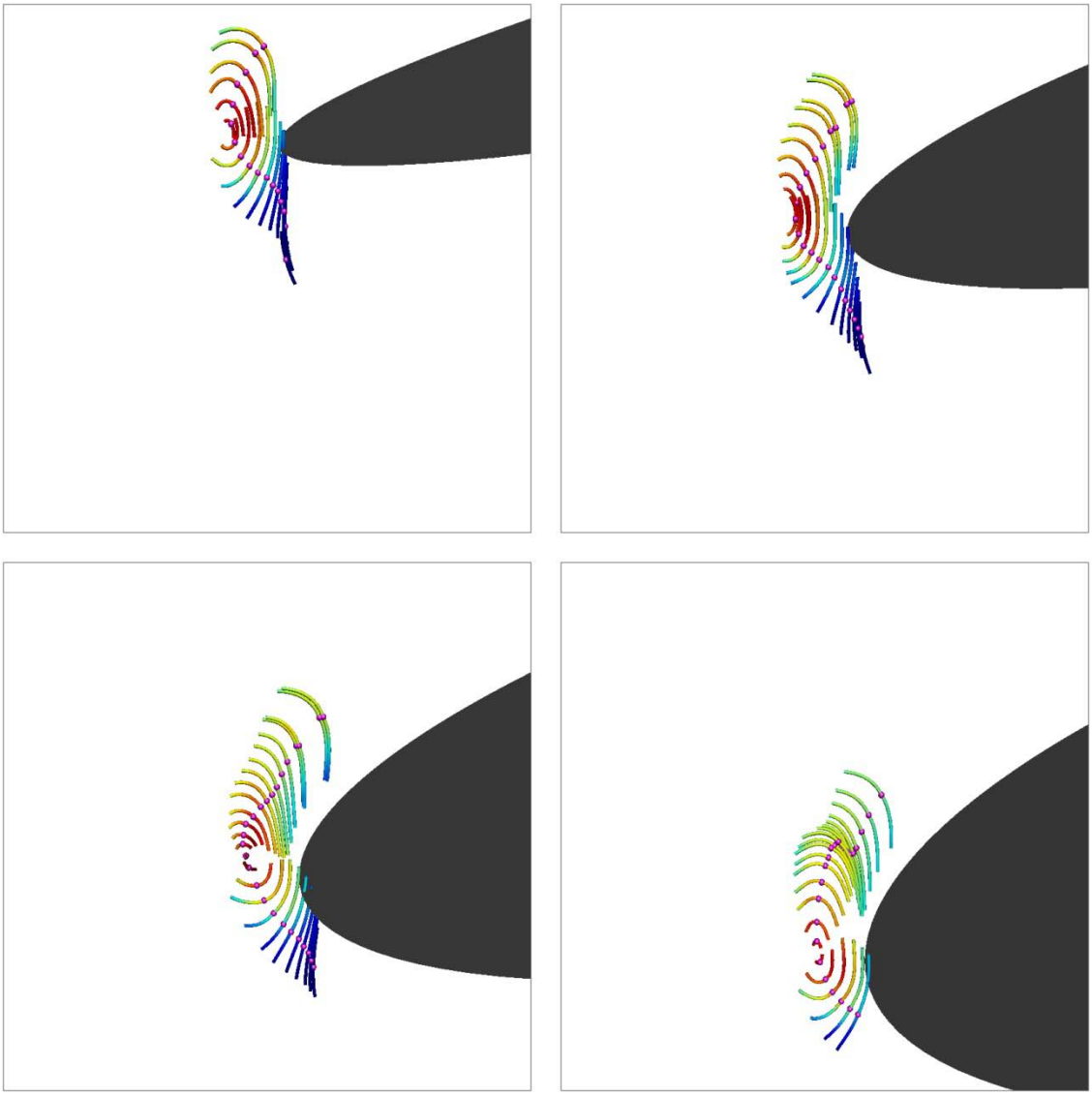

Figure 3.15: Tube representation of flowing streamlets following a vortex with vorticity magnitude mapped to color.

3.4 Dynamic Seed Curves

Flowing seed points is one approach to dynamically evolving seed curves however there are other options. Dynamic seed curves and surfaces can be created by using some feature of the underlying flow such as vorticity. Without restricting the search in some way this can be extremely time consuming. One way to speed up the

process is to generate iso-surfaces on a series of planes based on vorticity magnitude. These surfaces are then used to place flow line seeds at a series of time steps. This is also a rudimentary method of vortex core detection, but it is quite effective in situations where you know approximately where the most important vortices are.

Objects in the flow field can also be used to create dynamic seed curves and surfaces. By exploiting 3D modeling operations such as edge loop selection, surface patch selection, vertex normals and path extrusion a user can easily define seed objects at a single time step which will evolve over time as the base object moves in the flow domain. When dealing with complex data sets this is much more effective than static linear seed objects (Figure 3.16) which are typical in most commercial visualization packages.

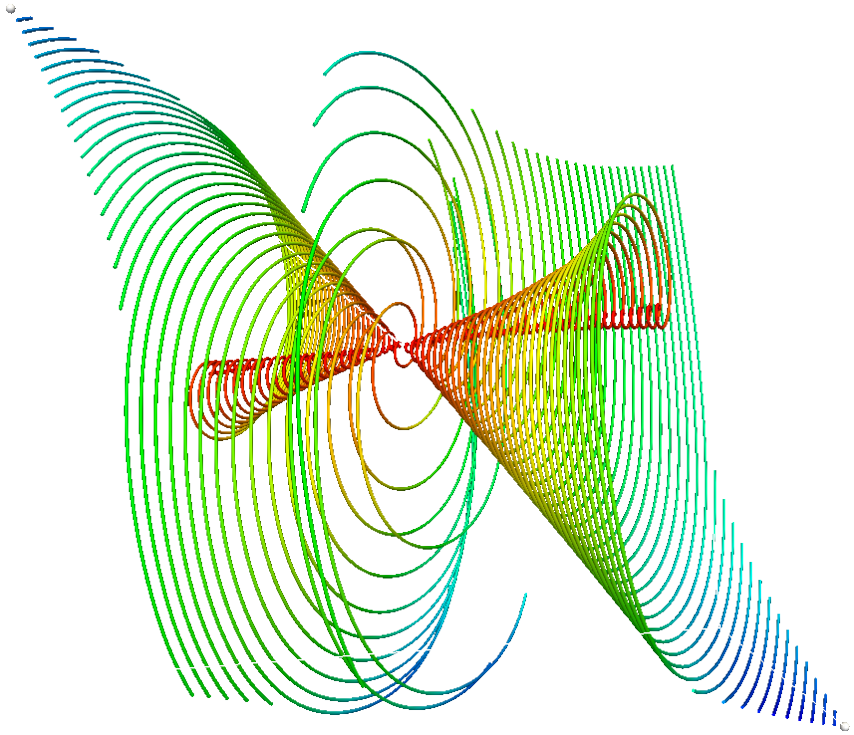

Figure 3.16: Result of a static linear seed object in a very simple data set.

3.4.1 Iso-Seed Planes

Using a global iso-value threshold alone is not an effective seeding method in the flapping wing data sets due to the fact that the points of maximum vorticity magnitude are generally all clustered together. It is also extremely computationally expensive to use a seeding method that involves reading all the scalar values from a large block of 3D vector field data. Tests have shown it is much more effective to partition the data set in some way and then seed based on the maximum vorticity in each section. One way to both restrict the search area and partition the data is to use a series of iso-planes. This works particularly well with flapping wing data sets because the planes can be bound to the leading edge, which is where the most interesting flow occurs. An example of using iso-planes for seed placement and vortex core detection in the flapping disk data set is shown in Figure 3.17.

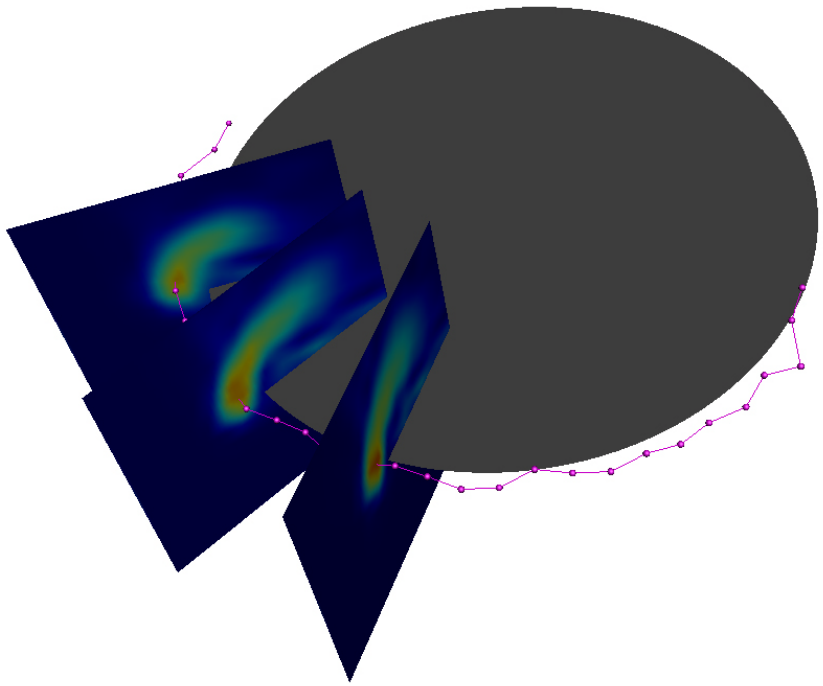

Figure 3.17: A series of color mapped iso-planes placed perpendicular to the leading edge vortex core.

One of the main strengths of the iso-seed plane method is that it works just as well with much more complicated data sets. This can be seen in the next example with the dragonfly data set. Iso-planes are placed perpendicular to the leading edge of each wing at a user defined number of points (Figure 3.18). The point on each plane with the maximum vorticity magnitude is chosen and connected into a vortex core line for each wing, which is then discretized into a series of seed points, as seen in Figure 3.19. The vortex core detection is limited to the area around the planes, however this can be desirable since we know approximately where the main vortex is.

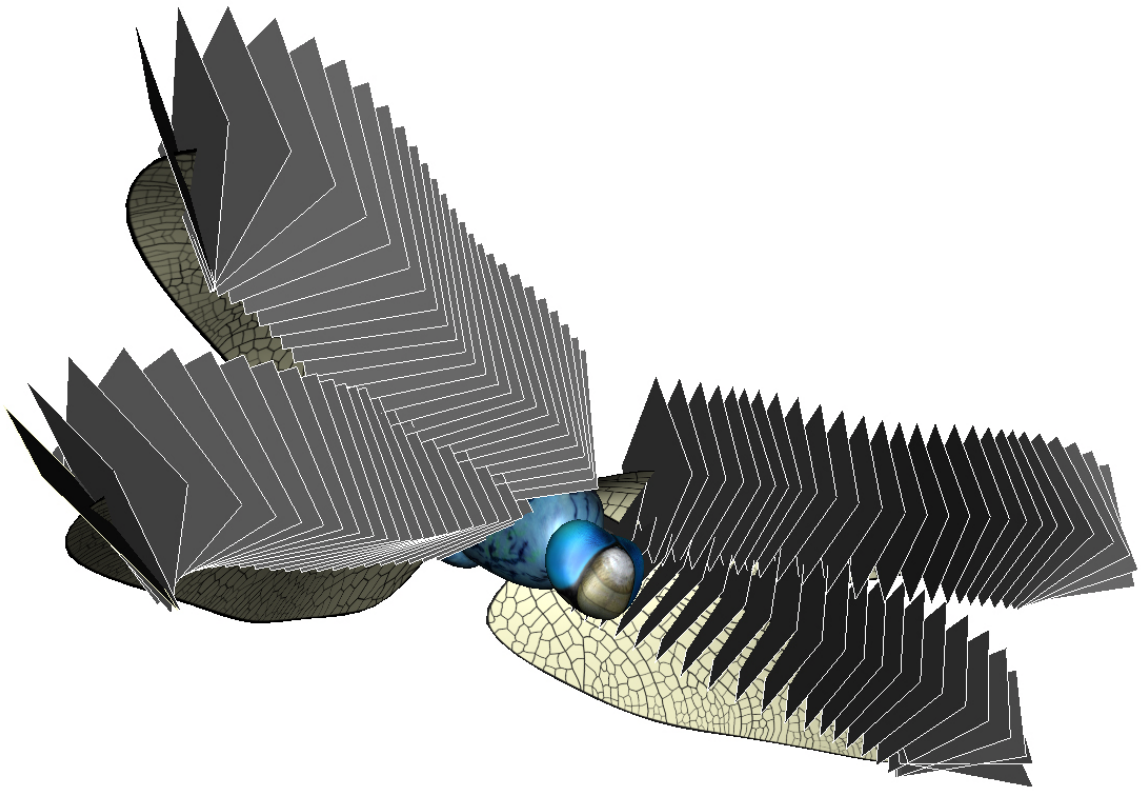

Figure 3.18: Dragonfly with iso-seed planes positioned along the leading edge of each wing.

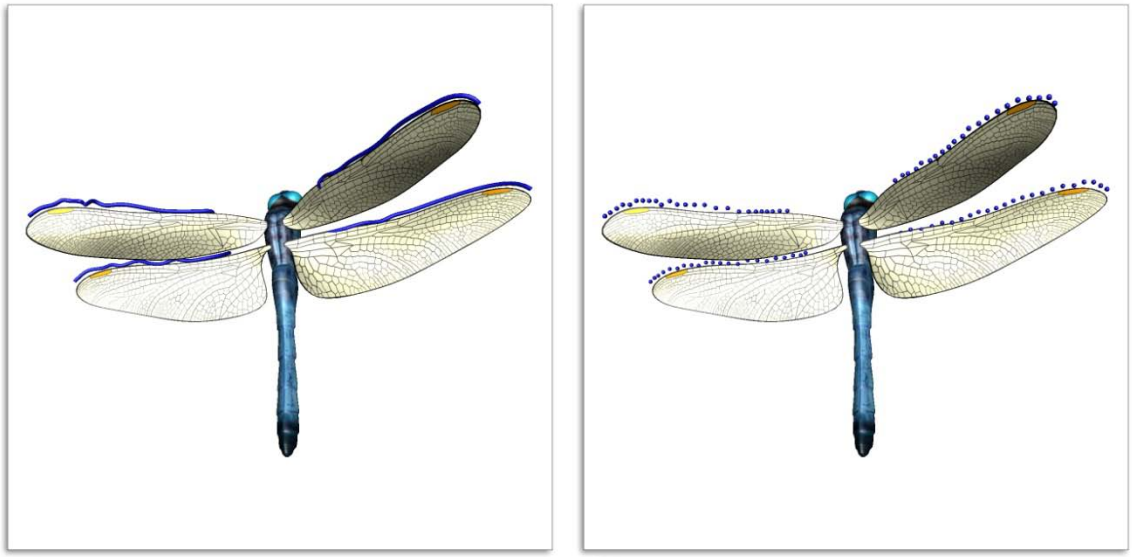

Figure 3.19: Iso-seed plane results at a single time step: (a) vortex core detection, (b) seed point generation.

3.4.2 Vertex Normal Seeds

Another approach to generating dynamic seeding objects in complex flows is the vertex normal seeding approach. This method works in flows that contain one or more mesh based objects moving within the flow domain. It completely ignores the underlying vector field and assumes that the mesh objects moving in the flow are causing the most interesting flow phenomena. Thus, seeds are bound near the surface of the objects in order to capture the neighboring flows features. Specifically, seeds are placed along the vertex normals of user selected vertices on objects in the flow domain. Vertex normals are calculated by averaging the surrounding face normals (Figure 3.20). Figure 3.21 shows how a user can control the distance from the original mesh vertex that a seed is placed at.

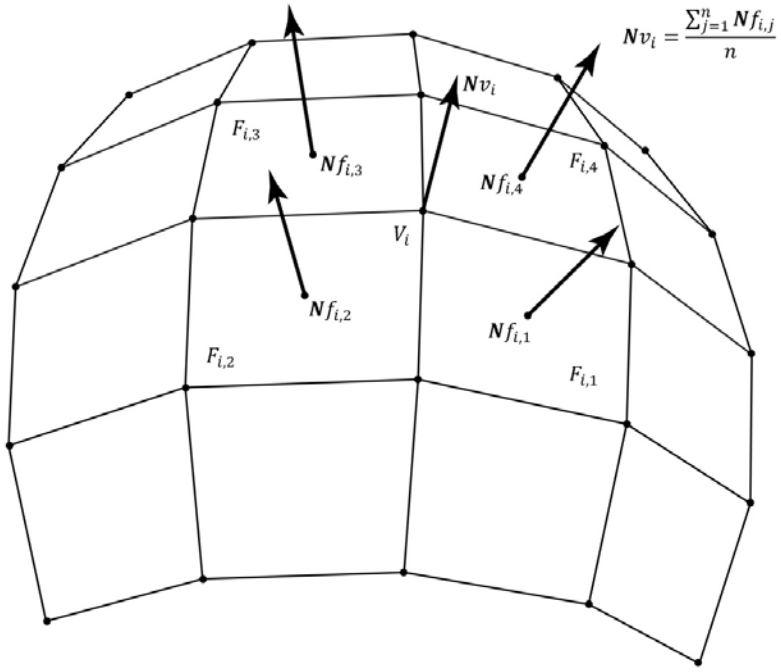

Figure 3.20: Calculation of vertex normals.

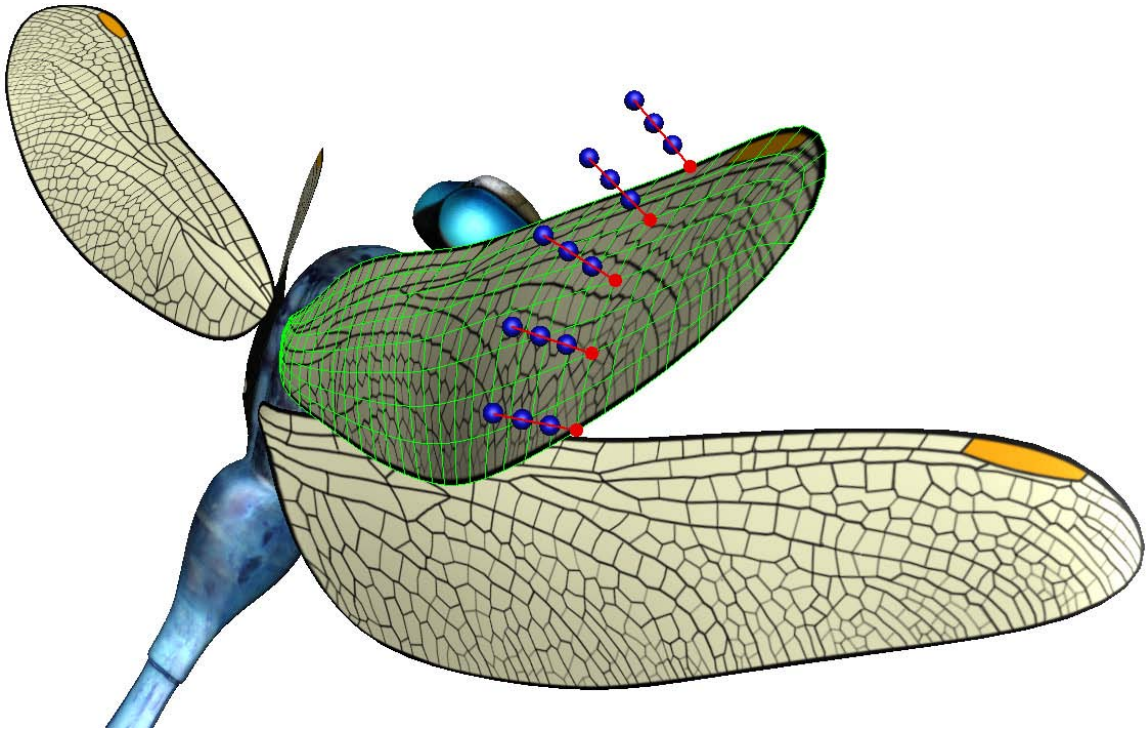

Figure 3.21: Multiple seed points placed along the normals of 5 vertices on the original immersed object.

In order to use vertex normal seeds effectively, a user must be able to quickly select vertices in the areas where seed points are desired. To accomplish this, several 3D polygon modeling operations were incorporated into the software. In particular, edge loop selection can identify a series of connected border edges by counting the number of faces adjacent to each edge connected the original selected edge. Edge loop selection was used to place seeds along the leading edge vortex of the dragonfly wing. Path extrusion can be used to increase the seed density along an edge loop. Surface patch selection and subdivision refinement can be used to control the density of seeds in a selected area.

There are several key areas in the dragonfly data set where vertex normal seeding is useful. Obviously the leading edge is important due to the leading edge vortex. The wing tips are also of interest in order to visualize vortex shedding. The wing roots are another important area if you are looking to visualize span-wise flow. Results from generating vertex normal seeds at the wing tip, wing root and leading edge are shown in Figure 3.22. All of these seeds were generated at a single time step with only a few mouse clicks, but they move dynamically with the wings as the vertices and their normals move. Results from testing iso-seed planes, vertex seeds and flowing seeds with a variety of flow lines are presented in detail in section 5.

Figure 3.22: Vertex normal seeds placed at the wing tips (first column), wing root (second column) and leading edge (third column) shown at three different time steps.

4 Implementation

The main novel seeding and perceptual contributions to visualization presented in this work are theoretical in nature, however their accuracy depends on the underlying implementation. For instance, flow lines are susceptible to numerical errors at many stages of their calculation and these errors tend to compound, resulting in images that may be visually appealing but not accurate. To demonstrate the validity of my results and the relevance of the CFD data to which they are applied, this section details several aspects of data generation, numerical integration, vector interpolation, time step size and rendering that were used. An explanation of how flow line computation is decoupled from visualization to allow for interactive exploration of the data is also presented.

4.1 Test Data Generation

In order to test the visualization methods proposed in this document prior to trying them on any more complex data, simulations of the flow around a single rigid flapping disk were performed. The simulations were run with an immersed boundary solver capable of simulating flows around complex moving objects in fixed Cartesian grids [6]. This grid structure is convenient because it is not necessary to convert the data to a less accurate computational space, however all of the concepts presented here will work with any grid structure upon which particle trajectories can be computed. A much more complex insect flight data set was also generated.

4.1.1 Flow Around a Flapping Disk

The flapping disk test data sets are motivated both by the need to test the visualization methods in unsteady flows containing dynamically moving objects as well as the desire eventually capture the vortices that form in insect flight and micro air vehicle simulations. Two versions of the simulation were done with both membrane as well as solid ellipsoidal flapping foils.

The term 'flapping' refers to the oscillatory pitch and heave of the foils. While these data sets contain only one foil which does not undergo any of the active and passive nonlinear deformations present in insect wings, they are a good starting point to evaluate how well a visualization technique will capture the vortices formed in a more complex insect flight simulation.

The meshes used in the simulations are defined by their axes a_x, a_y and a_z. In this study a_x and a_z were set to 1 which yields a circular shape for the solid and membrane foils. The thickness ratio a_y/a_x of the solid foil was set to 0.12. The mesh surfaces are composed of triangular faces. Figure 4.1 shows both the solid and membrane foils.

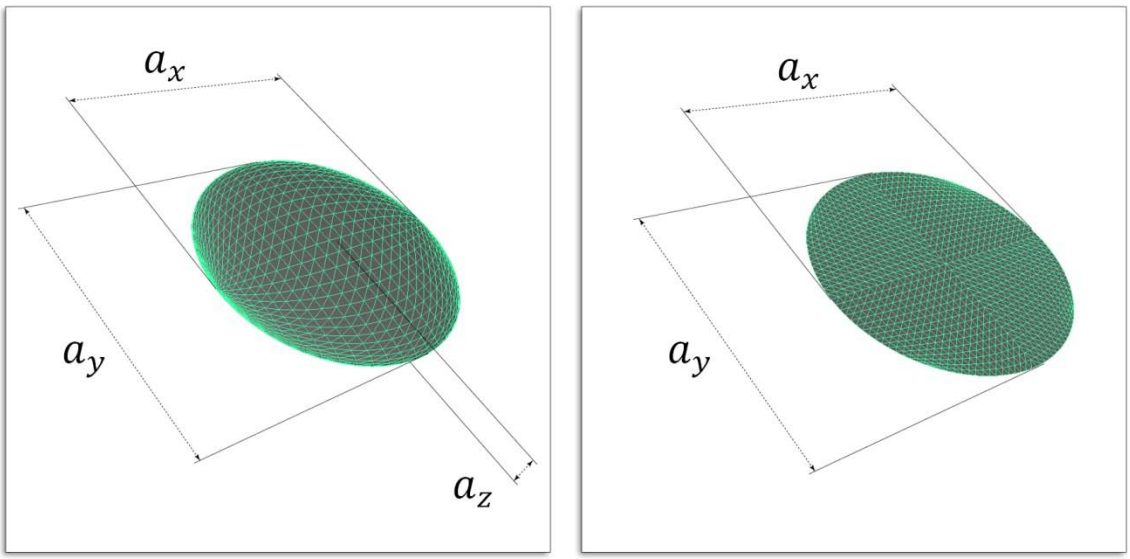

Figure 4.1: Foil meshes: (a) solid disk mesh, (b) flat plate mesh.

The flow moves over the disks as they undergo a pitch and heave motion. The pitch motion of the foils can be described as follows:

$$\theta(t) = A_\theta \cos(2\pi f t) \tag{4.1}$$

and the heave motion in the y-direction is described as follows:

$$y(t) = A_y \sin(2\pi f t) \tag{4.2}$$

where A_y is the heave amplitude, A_θ is the pitch amplitude, θ is the pitch angle with respect to the inlet flow, f is the flapping frequency and t is time. For the purposes of this work, A_y is set to 0.5 and A_θ is set to 30°. For each period of oscillation 800 time steps are exported, hence f is set to 0.00125. A total of six oscillations were simulated for each disk, hence t ranges from 1 to 4800. Also, a Reynolds number of 200 was used for both simulations. Figure 4.2 shows several time steps of one period of the flapping

disk at 100 frame intervals, and Figure 4.3 shows an image of the 2800th time step of the simulation visualized with a simple vorticity magnitude based iso-surface. The iso-surface visualizations show how the vortex circling the leading edge of the disk has the highest vorticity, however the direction the flow is moving is completely missed with this method.

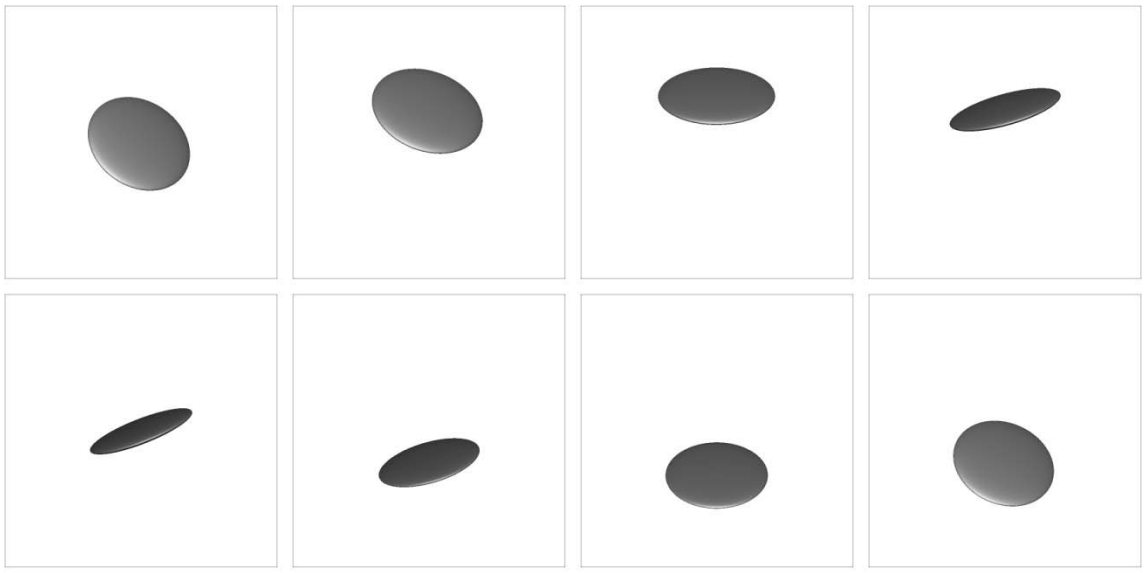

Figure 4.2: Eight snapshots of the solid flapping disk taken at 100 frame intervals.

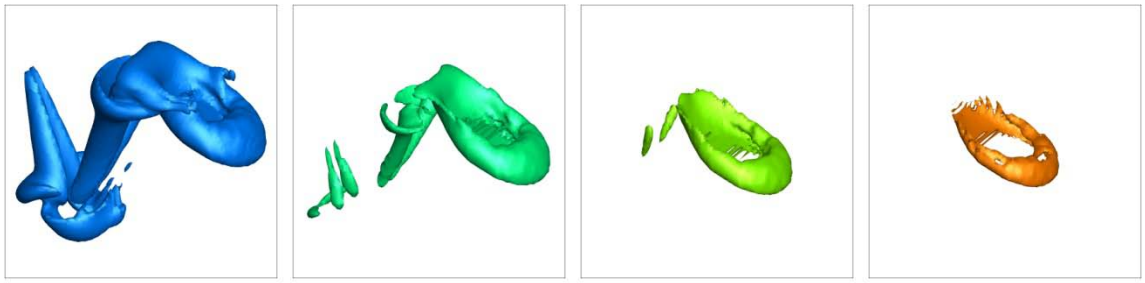

Figure 4.3: Vorticity magnitude iso-surfaces at one time step of the flow around a flapping disk simulation visualized with four iso-values.

4.1.2 Insect Flight Data Generation

The primary data set used for testing the flowing seed point and dynamic seed curve methods is from the simulation of a quad wing dragonfly. While it is clear that insect wings are not rigid it is difficult to capture and reconstruct the exact motions of the wings as they deform because they are moving extremely fast. For this reason most past CFD simulations of insect flight have been based on theoretical rigid wing models. This section details the techniques used to generate a more accurate reconstruction of a dragonfly's wings as it takes off, maneuvers and begins to hover. The resulting 3D deformable wing reconstruction was used in a CFD simulation and the results were then used for further test my geometric flow visualization methods.

4.1.2.1 Camera Setup

In order to capture the details of the wing kinematics several dragonflies were photographed from three angles with a high speed photogrammetric system, as shown in Figure 4.4. The cameras are each capable of generating 1000 frames per second of 1024 by 1024 black and white images. Very low exposure times were used in order to avoid motion blur on the fast moving wings. A synchronized end trigger system was used in conjunction with the cameras in order to save 2.5 seconds of data every time a desirable maneuver was performed. Camera calibration was done with the direct linear transform method. A conceptual look at the calibration setup is shown in Figure 4.5.

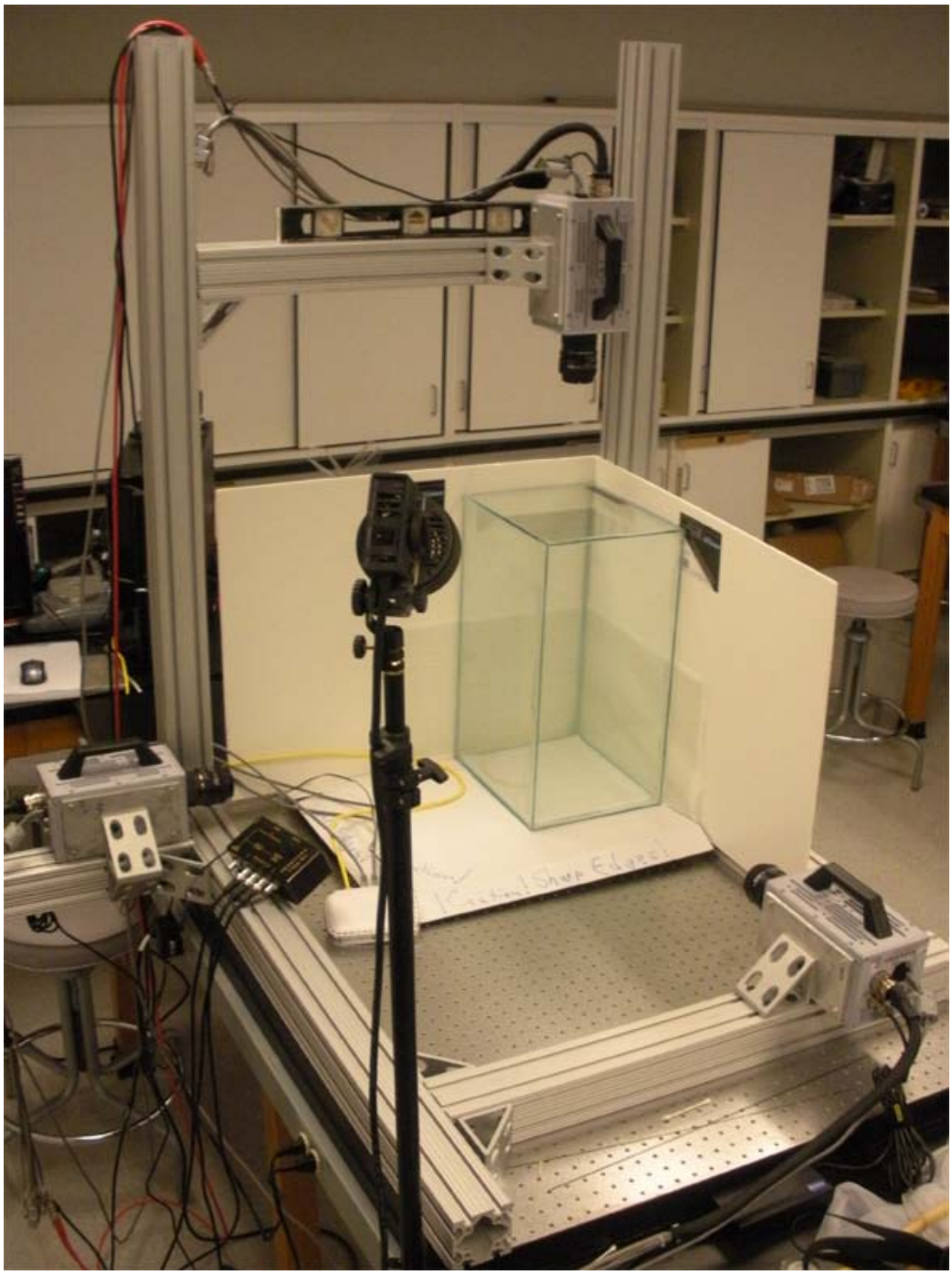

Figure 4.4: High speed photogrammetric system.

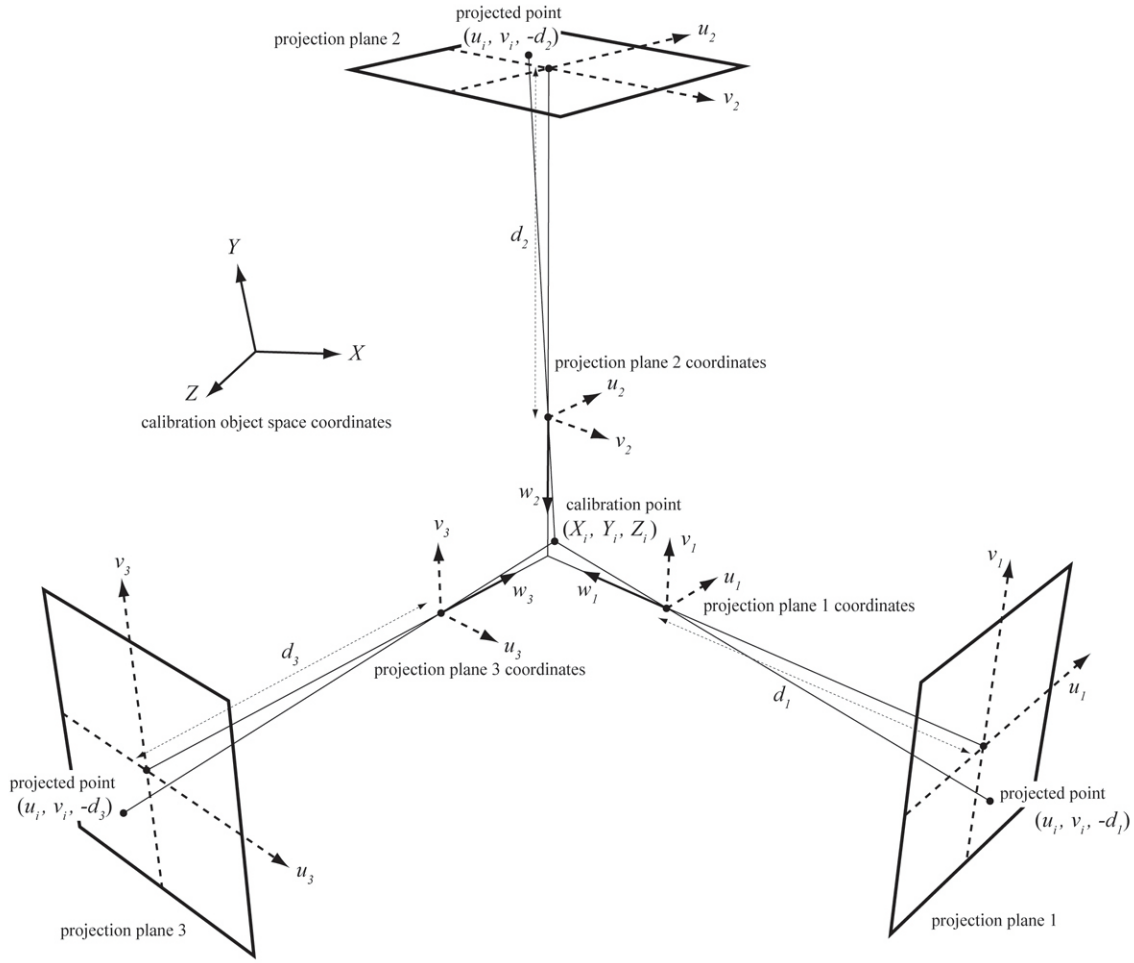

Figure 4.5: Camera calibration setup.

4.1.2.2 Image Data Acquisition

A variety of local dragonfly species have been photographed with this system. In particular, the blue dasher dragonfly was used for the first reconstruction and simulation experiment. The dragonflies are stimulated either with a fan or with a poker. Then the camera system is triggered after the dragonfly takes off, if any interesting maneuvers are observed. Examples of the image data acquired along each axis is shown in Figure 4.6. Data is chosen for reconstruction based on the following criteria:

- Reproducibility of the observed maneuver
- Image quality throughout the maneuver
- Importance of the observed maneuver

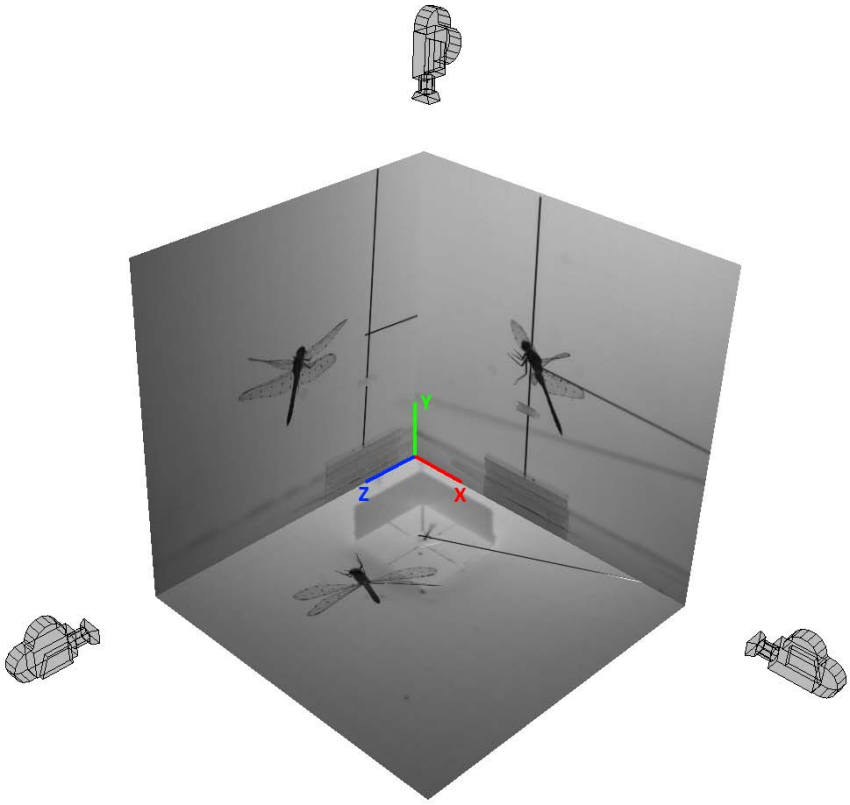

Figure 4.6: Image projections along each axis of a dragonfly in flight.

Data acquisition presented several challenges due to the speed at which the dragonfly wings are moving. First, camera synchronization was an issue. The cameras are configured as a master and two slaves. They are fairly well synchronized, but the dragonfly wings are moving extremely fast, so any small gap in time between when the three cameras fire is noticeable in the data. Camera focus is another issue. A very long lens was needed due to the small size of the details on the wings, but this left us with a

very shallow depth of field in which we could focus. If the dragonfly were to take off and fly a few inches in one direction it would go completely out of focus. Glare on the wings was another issue. Due to the speed of the wings we had to use a very low exposure time in order to avoid motion blur. This low exposure time meant we also had to use a lot of light, which lead to glare obscuring details on the wings at times. Finally, uncooperative dragonflies were a major challenge. Much trial and error was necessary to get a few good recordings of important maneuvers such as takeoff and hover.

4.1.2.3 3D Reconstruction

Prior to gathering data the dragonfly wings are marked with a marker to simplify the reconstruction process. Hierarchical Subdivision Surfaces were used to create a smooth mesh for the body and wings based on a top down image of the original dragonfly. Subdivision surfaces are essentially a unification of polygon meshes and parametric surfaces. In particular, the Catmull–Clark algorithm for subdivision surfaces was used because Stam demonstrated it is capable of smoothing meshes with arbitrary topology [111]. The initial 3D reconstruction of the blue dasher dragonfly and a corresponding image of the original insect are shown in Figure 4.7. The body was not included in the simulation, but it was reconstructed to aid in analyzing the dragonfly's kinematics.

Figure 4.7: Initial reconstruction of the dragonfly next to an image of its live counterpart.

Smoother surfaces can be generated with this surface representation by recursively repeating the subdivision algorithm with the resulting vertices of the previous subdivision. A two level Catmull-Clark subdivision surface hierarchy was used to model the dragonfly wings because it allows for enough control over the deformations the wings undergo while flapping without having so much detail that it begins to occlude part of the data it must be aligned with (Figure 4.8). More detail can always be added later prior to running the simulation.

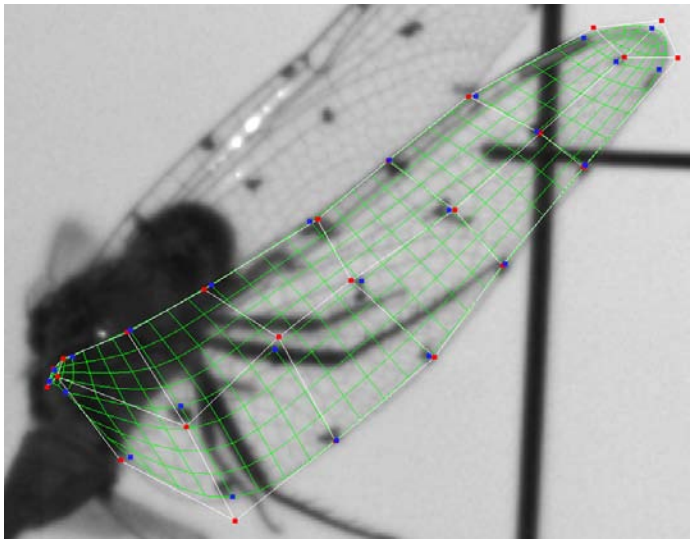

Figure 4.8: Subdivision surface based template wing aligned to the corresponding wing in the image data.

Wings based on this surface scheme are then aligned with the images of the dragonfly along each axis. Since there are three 2D images, two options are available for each of the three axes, as can be seen in Figure 4.10. Unfortunately due to camera synchronization, perspective and focus issues, the same point in each of the three images at each time step does not always lie in the exact same place on corresponding axes. In this case, the clearest option for each of the three axes is used. Figure 4.9 and Figure 4.10 show two different views of a reconstructed dragonfly and the corresponding images.

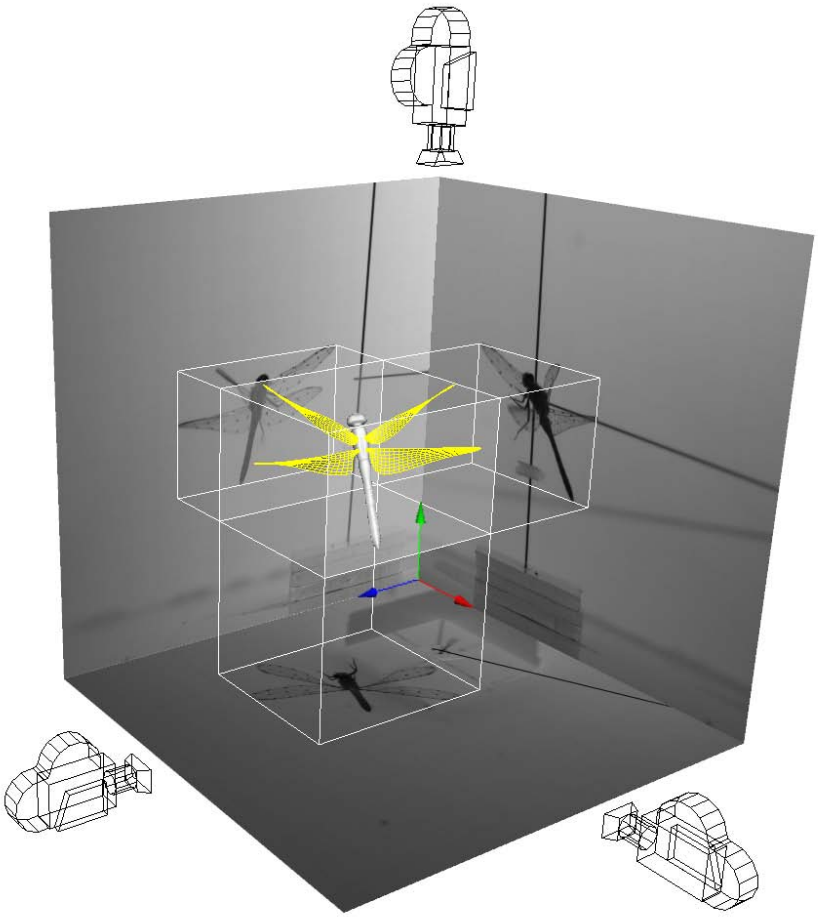

Figure 4.9: Image projections with the corresponding time step of the reconstructed dragonfly.

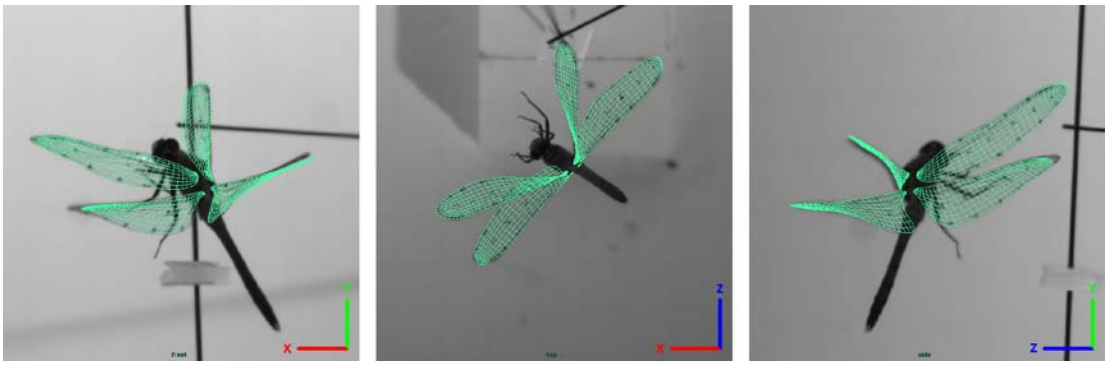

Figure 4.10: All four dragonfly wings aligned to the image data at a single time step.

In order to measure the accuracy of the reconstruction, it is combined with segmentations of the original data. The dragonfly wings and body are segmented in the original images through a series of thresholding operations. Orthographic projections of the reconstructed wings are generated at each time step (Figure 4.11). The segmentations and projections are then compared with a pixel based accuracy metric. Measurements of the number of true positive, true negative, false positive and false negative pixels are made at each time step. Figure 4.12 shows a plot of the accuracy measure recorded at 50 time steps from 0.05 seconds of wing flap for the right ipsilateral wings when the reconstruction is compared to the images taken from one of the high speed cameras.

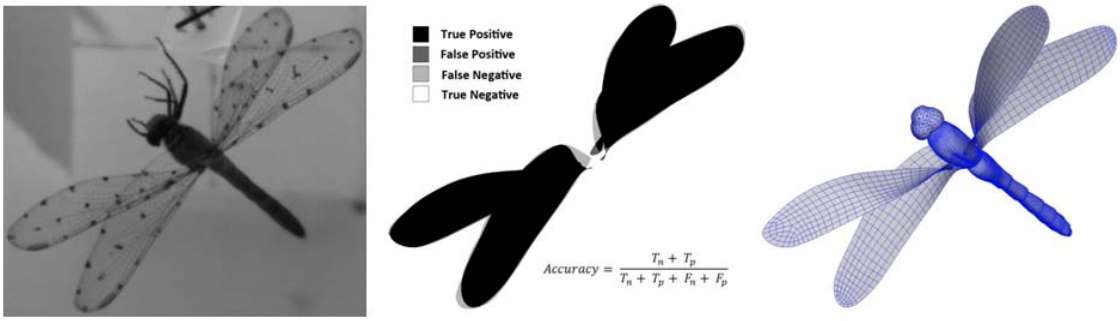

Figure 4.11: Comparison between the original image of the dragonfly and the projection of the reconstruction along the same axis.

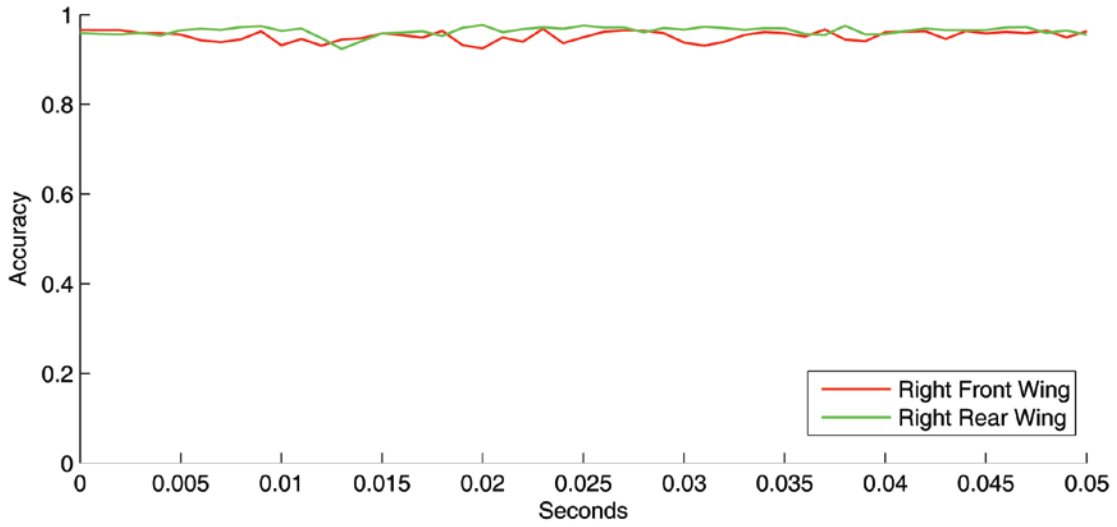

Figure 4.12: Plot of the accuracy measured for the right ipsilateral wings from 50 consecutive time steps.

4.1.3 Data Set Sizes

Large data sets are one of the most challenging aspects of unsteady flow visualization. Data sets often consist of several million grid points per time step and several thousand time steps. At each grid point a location point and a velocity vector must be stored along with multiple other scalar quantities like vorticity and helicity. Table 4.1 shows the data set sizes for the two main data sets used in this study. Also, Figure 4.13 shows a rendering of the simulation grid used for each time step of the dragonfly data set in order to convey just how large the data used in this work was.

Dataset	Grid Dimensions	Grid Points	Vector Field Size	Time Steps	Total Size
Dragonfly	176x152x192	5,136,383	1.4GB	800	~1.15TB
Flapping Disk	145x129x105	1,964,025	450MB	2400	~1.08TB

Table 4.1: Comparison of data set sizes.

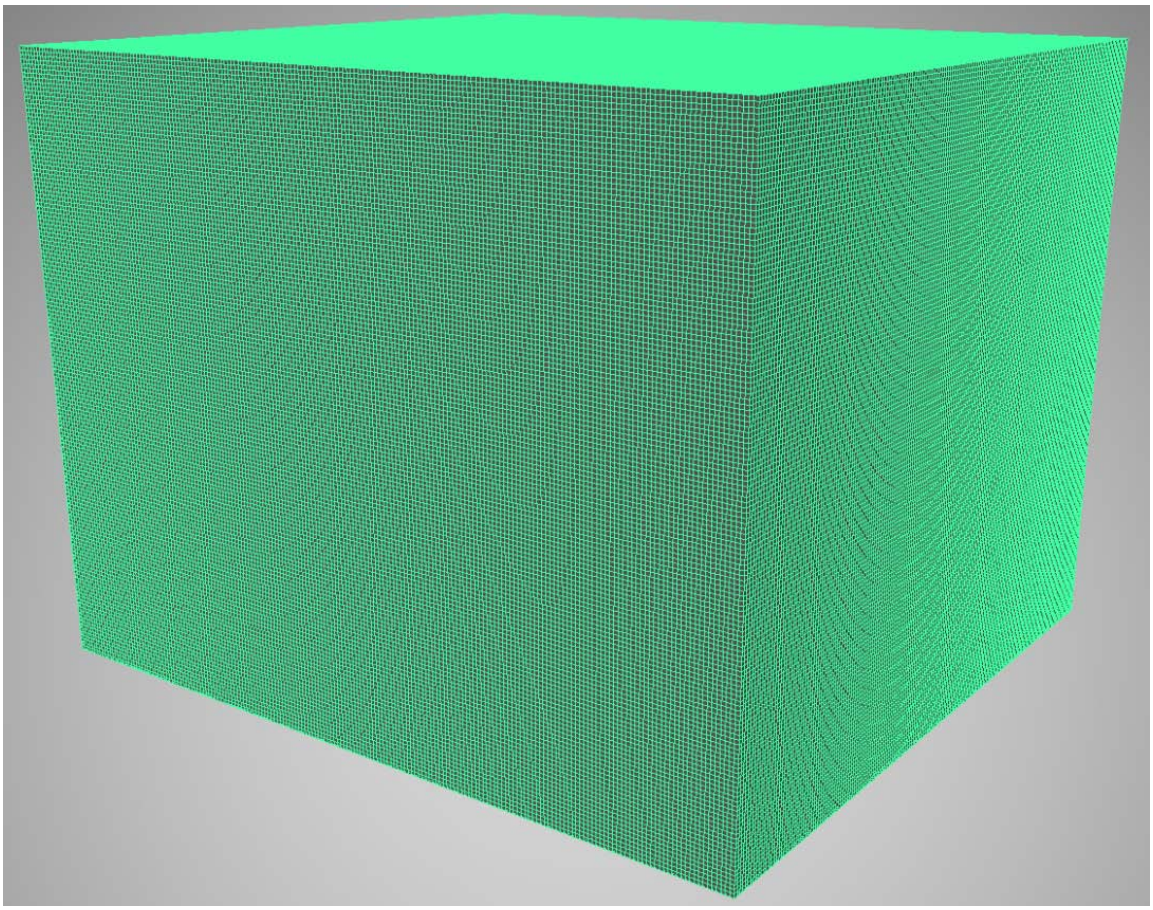

Figure 4.13: Dense grid used for each time step of the dragonfly simulation.

4.2 Dragonfly Kinematics Analysis

Ultimately the goal of this work is to improve visualization of vortices in unsteady flows with moving immersed boundaries, however these visualizations are more valuable when they can be related to some physical phenomena that caused the vortices to form or dissipate. Thus, an analysis of the Euler angles of the dragonfly body, wing motion and wing deformation was performed for the first two flaps after the dragonfly takes off and begins to fly. The results of this analysis were used to identify key time steps and wing areas to focus on when applying the flowing seed point visualization method in the next section.

Euler angles, roll, pitch and yaw, are widely used in aerospace to describe rotation of flying objects. Relative to the speed at which the wings flap, deformations in the dragonfly's body are minimal over such a small time window so the body was assumed to be rigid. Thus, the Euler angles as well as translation distance from the origin along each axis can be measured in the world coordinate system directly from the dragonfly's body, as seen in Figure 4.14. While the simulation only captures the first three wing flaps due to time constraints, the reconstruction includes nine wing flaps. Kinematic analysis was performed on all nine reconstructed flaps.

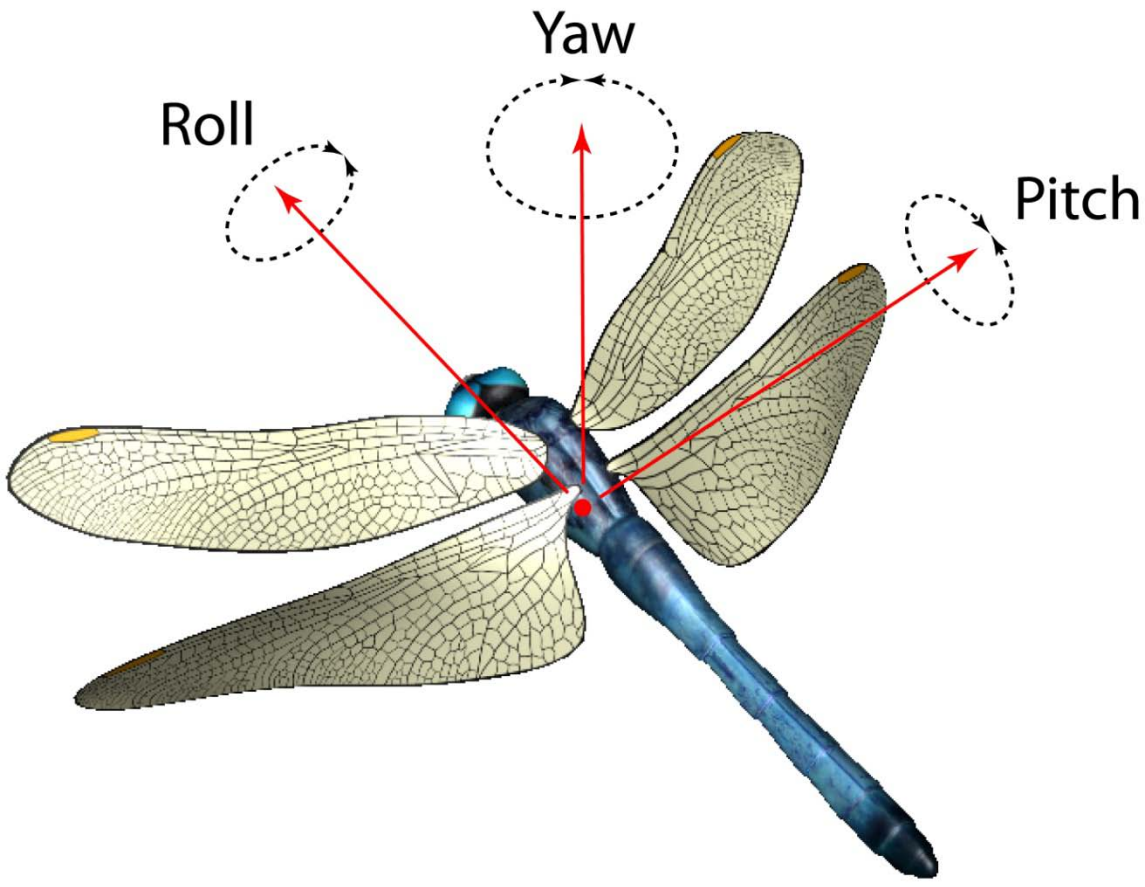

Figure 4.14: Euler angles measured from the dragonfly's body.

Once the Euler angles and translation distances are calculated they can be subtracted from both the dragonfly's body and wings to create an object coordinate space. The object coordinate space is useful for studying wing kinematics. Flapping flight descriptors such as phase difference, stroke plane inclination, angle of attack and range of motion can be measured by defining anchor points on the dragonfly's wings and then tracking their movements relative to the body. For example Figure 4.15 shows a perspective view of the wing tip trajectories over time, and Figure 4.16 shows how the range of motion around the *x* axis can be measured from the maximum wing tip trajectories projected onto the *yz* plane.

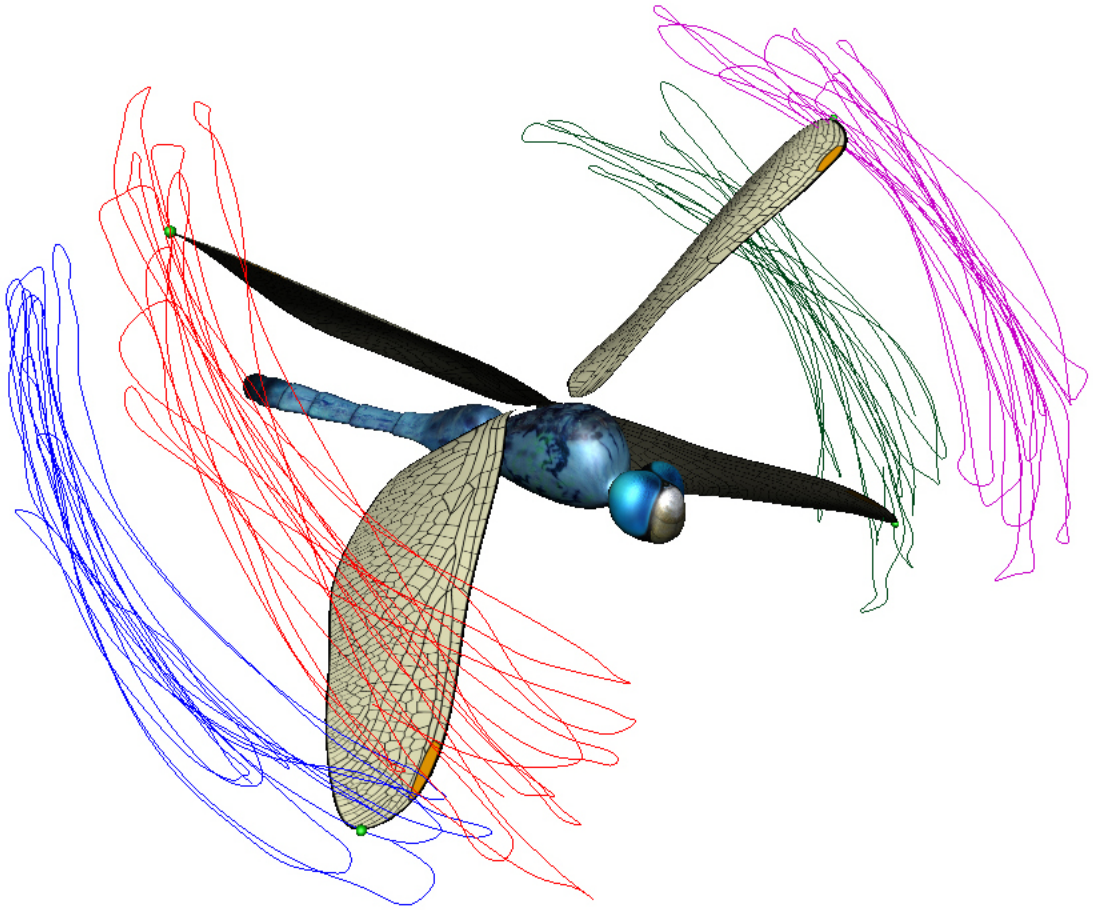

Figure 4.15: Perspective projection of the wing tip trajectories from the full nine reconstructed wing flaps.

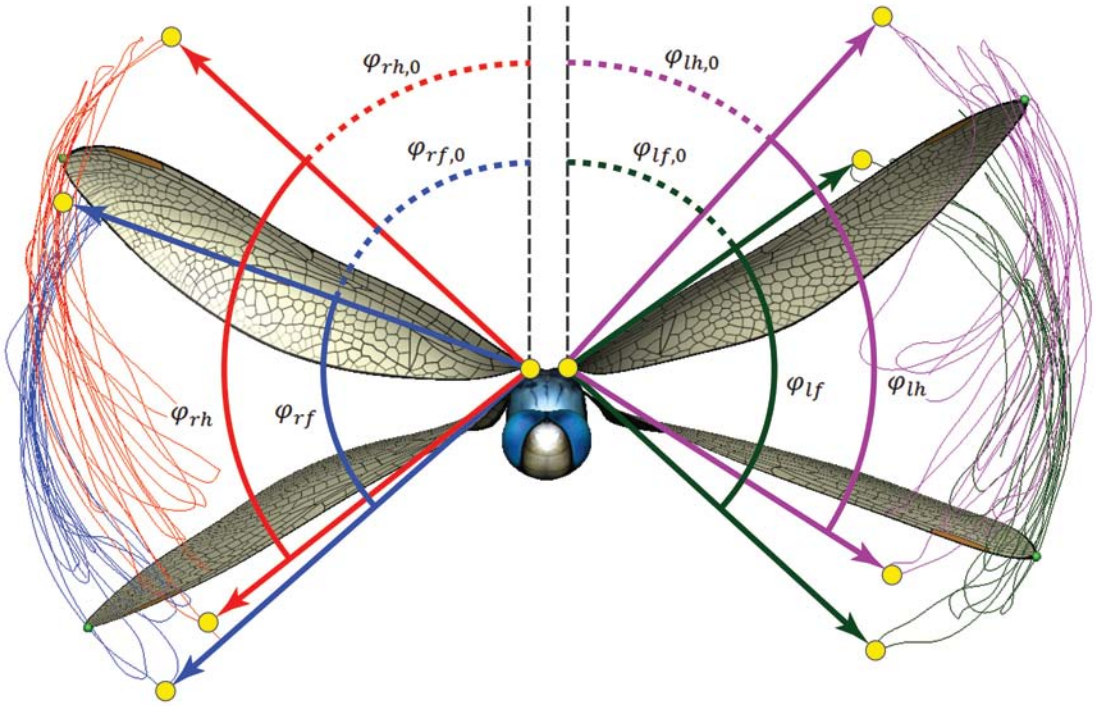

Figure 4.16: Range of motion of each of the dragonfly's wings.

Stroke change can also be determined by examining many points on the individual wings. Since the wings are constantly undergoing complex nonlinear deformations, no one point on a wing can accurately determine the time of stroke reversal, so the average motion of all wing vertices was used. Peaks in the motion history are determined and the average time step of each group of neighboring peaks is chosen as the stroke change time. The peaks in Figure 4.17 correspond to stroke reversals, so the eighteen peaks per wing correspond to the nine reconstructed flaps.

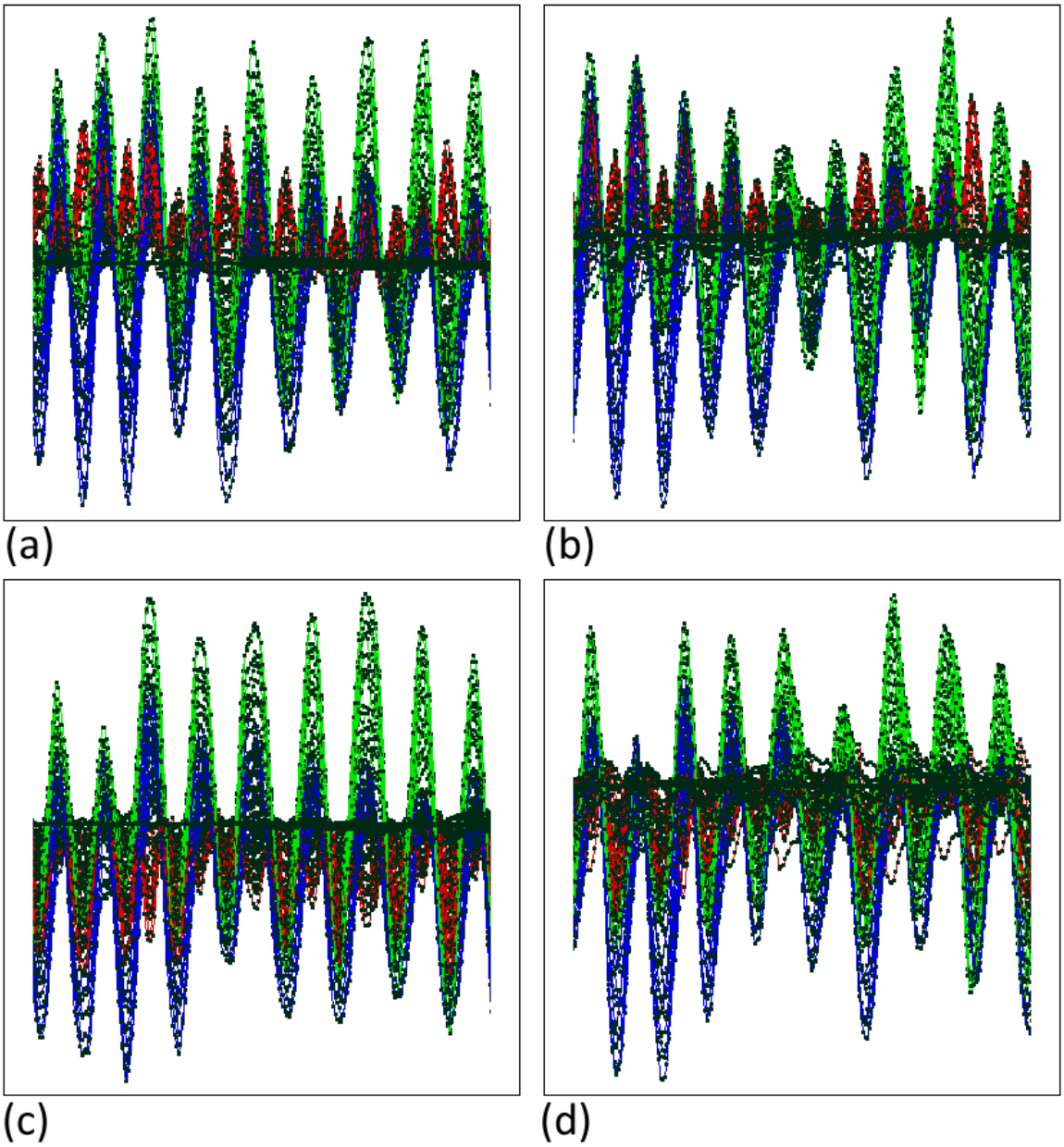

Figure 4.17: Stroke change history illustrated as peaks in the motion of each wing vertex over time: (a) left forewing, (b) left hind wing, (c) right forewing, (d) right hind wing. Movement in the x direction is shown in red, the y direction is green and the z direction is blue.

4.2.1 Camber to Chord Ratio

The degree to which the dragonfly's wings deform during takeoff was also studied. Prior to this work, the majority of flapping wing insect studies used less accurate

planar wing representations, which did not include camber. The camber to chord ratio is defined as the ratio between the maximum camber height and the wing chord (Figure 4.19). This figure also shows the evolution of the camber to chord ratio over time for a single position on the left forewing. Figure 4.18 shows the evolution of the chamber to chord ratio over time. Ultimately a rigid wing version of the same dragonfly will be created to compare the effect of wing deformation on vortex production.

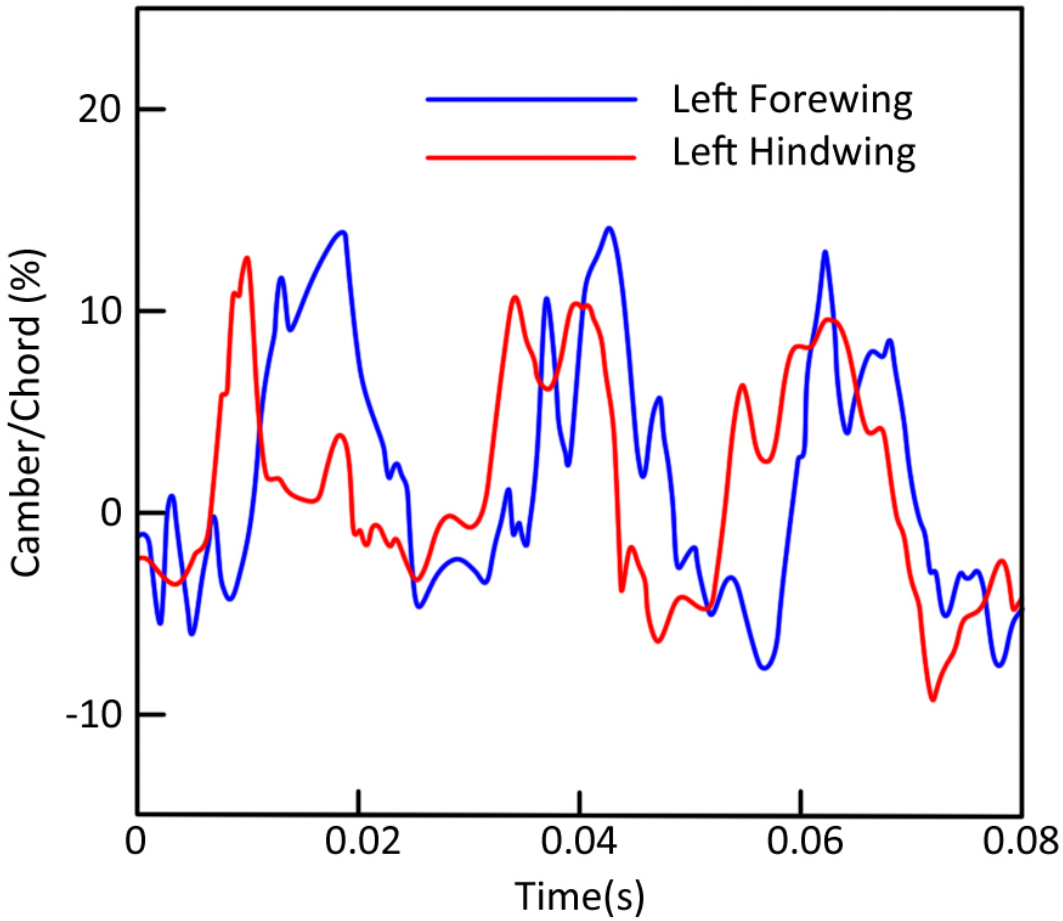

Figure 4.18: Time history of camber/chord ratio in the left wings at the mid-chord cross section. This chord to chamber ratio plot was created by Zach Gaston and Zongxian Liang.

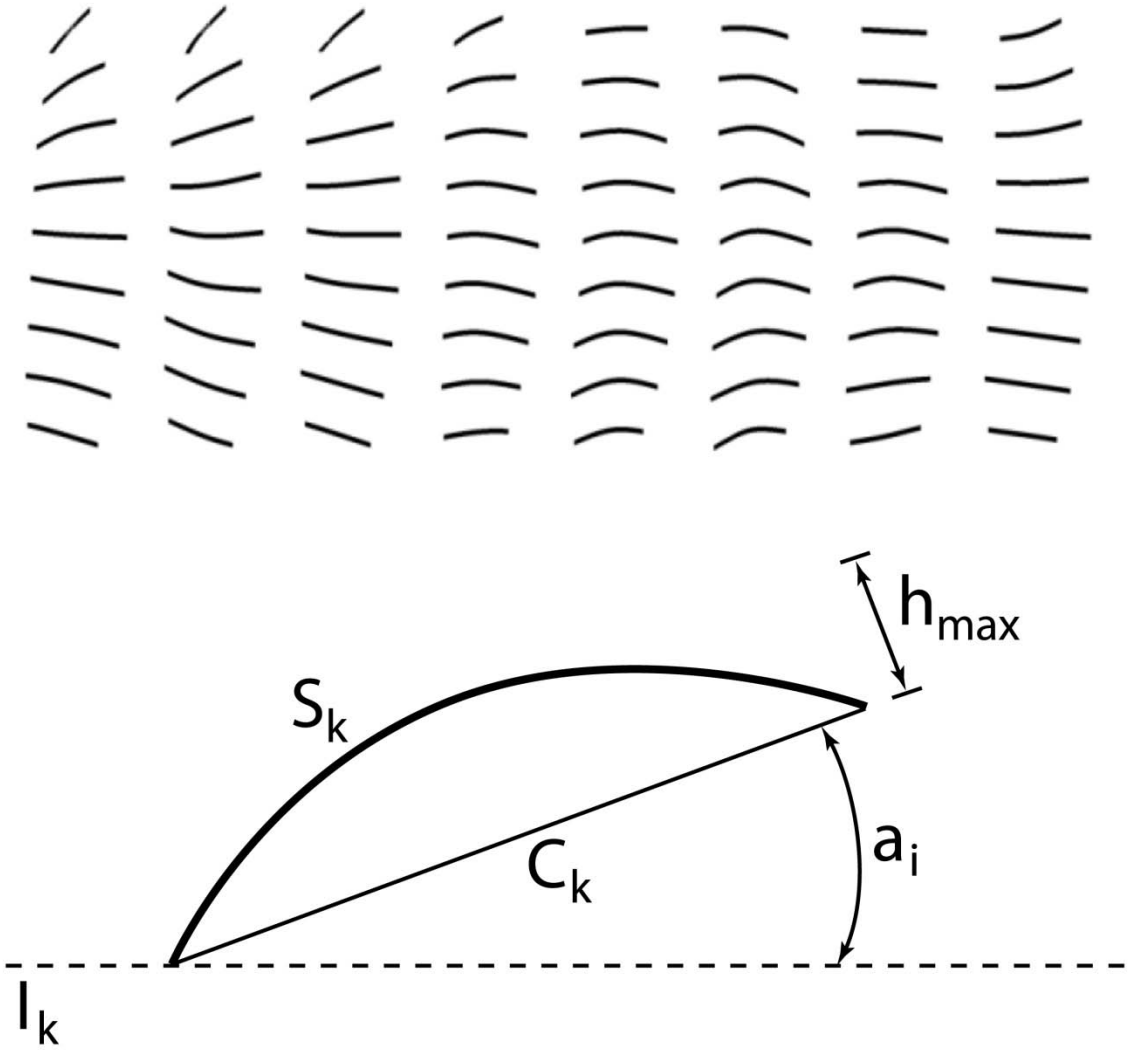

Figure 4.19: Camber of a point on the dragonfly's wing over time as well as an explanation of the camber chord measurement.

4.3 Dragonfly CFD Simulation

It is also beneficial to compare visualization results with various measurements, such as the force history, made by the CFD simulation that produced the vector fields. In particular vortex shedding at stroke reversal should correspond to a drop in lift, while delayed stall that happens during both the up and down stroke should correspond to increased lift. A second-order finite-difference based immersed-boundary solver [7] was

used to simulate the flow over the immersed 3D deformable wings. The Eulerian form of the Navier-Stokes equations are discretized on a Cartesian mesh. A ghost-cell method is used to enforce boundary conditions. A validation of this method can be found in [6]. Figure 4.20 shows the lift history for the left wings during the first two wing beats.

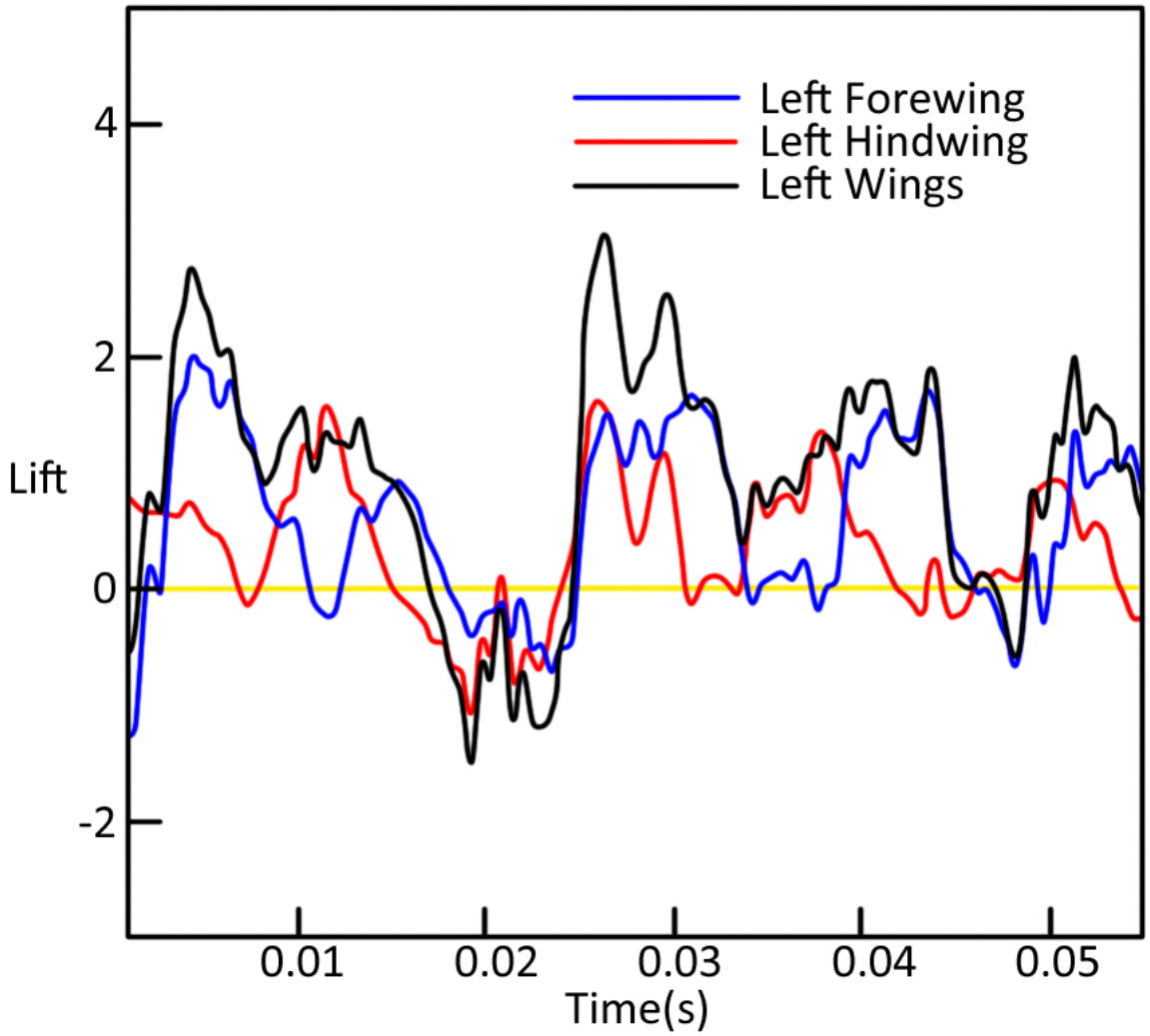

Figure 4.20: Time history of lift coefficient over the first two strokes. This force history plot was created by Zach Gaston and Zongxian Liang.

4.4 Numerical Integration

The forth order Runge-Kutta numerical integration method was used to integrate all flow lines in this work. This method has been proven accurate for streamline and path line integration. In order to further validate the correctness of my streamline integration code, the results were compared with Tecplot and VTK for several streamlines with the same seed point and integration step size. The following code snippet shows how the Runge-Kutta method is used for integrating a streamline starting at point `P1` with an integration step size of `h`. Tri-linear interpolation was used to interpolate vectors at points that do not lie directly on the grid.

```
K1 = h * vectorField->interpolate(P1);

P2 = P1 + (0.5 * K1);
K2 = h * vectorField->interpolate(P2);

P3 = P1 + (0.5 * K2);
K3 = h * vectorField->interpolate(P3);

P4 = P1 + K3;
K4 = h * vectorField->interpolate(P4);

Pk = P1 + ((0.166667 * K1) + (0.333333 * K2) + (0.333333 * K3) +
(0.166667 * K4));;
```

4.5 Integration Step Size

The CFD solver used to generate all of the data is not based on any physical unit measures. Thus, the grid units as well as time steps have no physical equivalent. The numerical integration time step used between simulation time steps is in the same "unitless units" as the simulation time steps when dealing with unsteady data (it is controlled mainly by how smooth the resulting curve is with steady data).

However, the dimensions of a single grid cell limit the range of acceptable integration time steps (as a rule of thumb, the integration time step should not be larger than the distance between grid points). This in turn controls how many time steps of the simulation actually need to be exported. The data sets used in this document use a simulation time step of 0.0015 units and the distance between grid points is approximately 0.004. Therefore, only every other time step of the simulation actually needed to be used and the default numerical integration time step was 0.003 units.

4.5.1 Adaptive Step Size

When dealing with a very large number of streamlines, the vertex count can get extremely high. This problem compounds when dealing with tubes instead of lines. One way to reduce the number of vertices and also reduce truncation and round-off errors is to use an adaptive step size when integrating. One proven adaptive time step scheme that works with fourth order Runge-Kutta integration is step-doubling. Step-doubling automatically adjusts the step size which saves memory by taking larger integration steps when the vector field is changing slowly.

In order to further reduce the footprint of the flow lines after they have been integrated they are first converted to first degree B-Spline curves. They are then rebuilt as third degree NURBS curves with a tolerance of 0.001 so there is very little error introduced but the number of vertices required to store each line is further reduced. Autodesk Maya's C++ API was used to rebuild the flow lines as NURBS curves. Figure 4.21 shows a before and after image of the number of vertices required for a single streamline. The difference is most apparent in the relatively straight portions of the streamline.

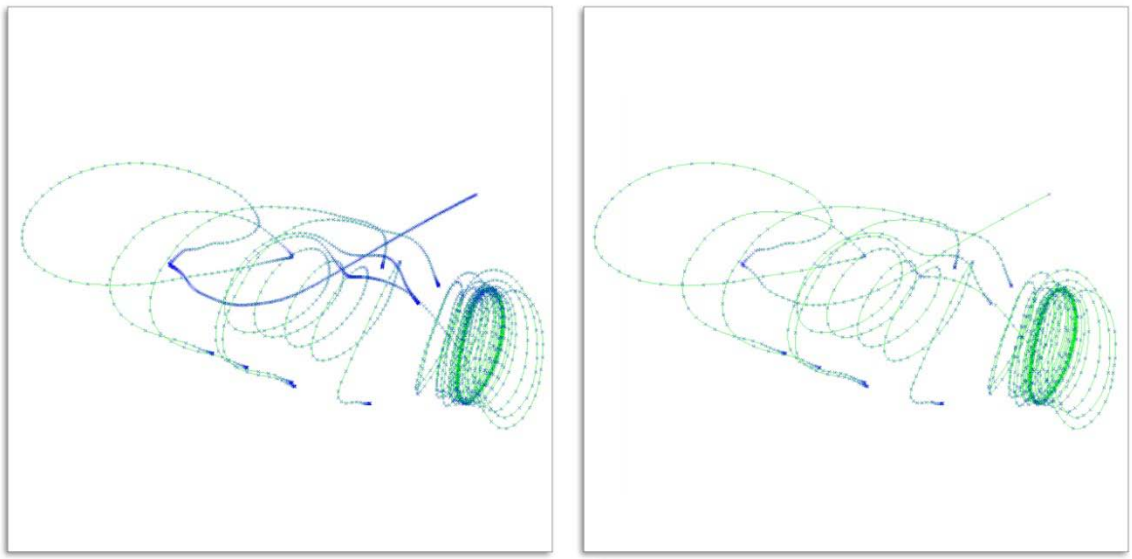

Figure 4.21: Two representations of the same streamline: (a) without an adaptive step size, (b) with an adaptive step size.

4.6 Stream Tubes

While much work has been done to improve the visual presentation of flow lines through illumination models, a mesh based tube representation offers more flexibility when rendering. Surface shininess, self shadowing and global illumination improve

shape and depth perception of the rendered tubes. In order to generate stream tubes, a circular NURBS curve was extruded along each flow line. This can be seen in Figure 4.22. A comparison between a polygonal mesh version of the same stream tube and a NURBS version with the same number of vertices is show in Figure 4.23.

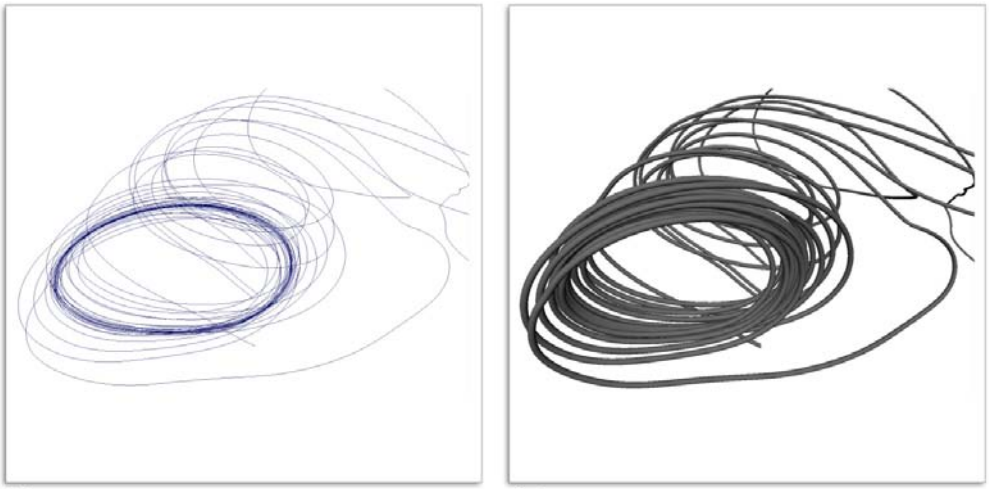

Figure 4.22: Two representations of the same streamline: (a) line representation, (b) tube representation.

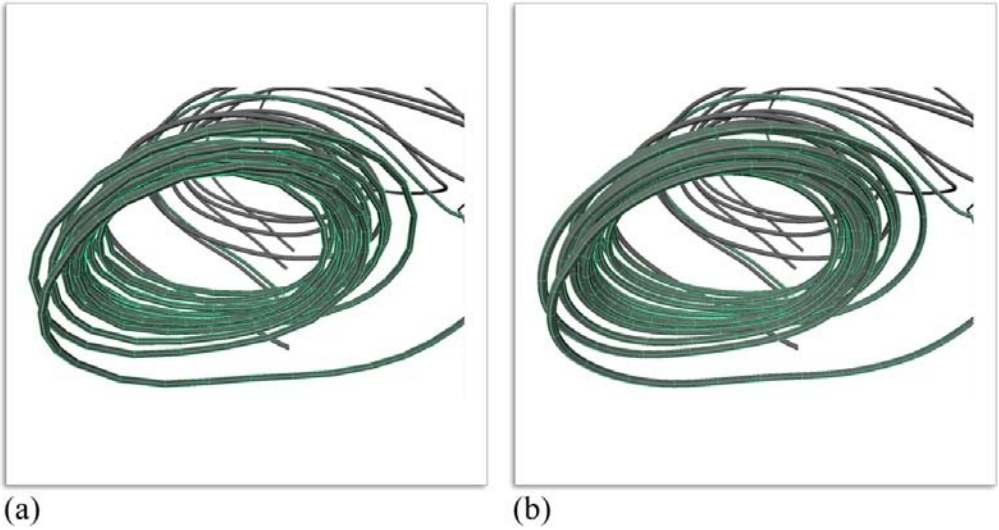

Figure 4.23: Two representations of the same streamline tube with the same number of vertices: (a) polygon mesh version, (b) NURBS surface version.

4.7 Rendering

Global illumination works by shooting thousands of rays out from light sources. These rays then take on the color of the first object they hit. As the rays are reflected onto other objects, there is a slight mixing of colors which results in more realistic rendering that is pleasing to the eye. Ambient occlusion on the other hand is a global rendering method based solely on the scene geometry. Essentially it darkens surfaces based on how close they are to other surfaces. This helps account for light attenuation due to occlusion. The ambient occlusion pass is multiplied by the color pass in order to keep these areas dark in the final composite rendering. When used with streamlines this improves depth perception, as can be seen in Figure 4.24.

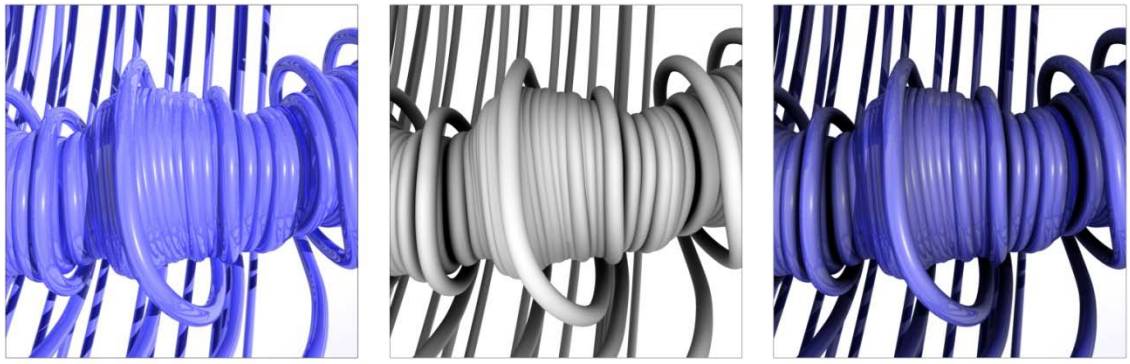

Figure 4.24: Composite rendering of several densely seeded streamlines in a vortex: (a) Global illumination rendering, (b) Ambient occlusion rendering, (c) Composite rendering of the previous two images.

Experiments were also performed to use rendering techniques and semi-transparent surfaces to highlight areas of recirculation or near recirculation. When streamlines are very densely seeded and integrated for a long time, the resulting images

can suffer from occlusion. However if the streamline tubes are semi-transparent you will be able to see only the areas where there is considerable overlap. If transparency values are high enough, you will only see closed streamlines and flow recirculation (Figure 4.25). This method only works for steady flow because the vortices must not move.

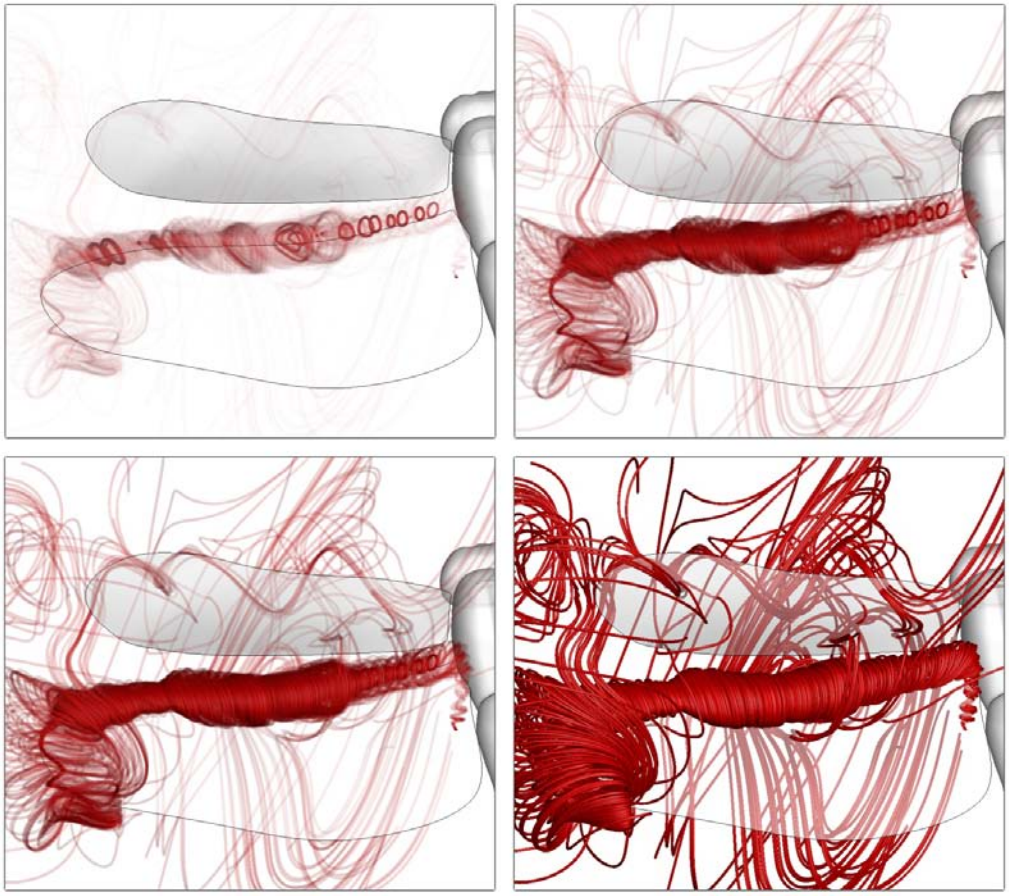

Figure 4.25: Densely seeded streamlines around the leading edge vortex of the left hind wing. When the streamline tubes are highly transparent, closed streamlines as well as areas of recirculation that do not include closed streamlines are highlighted.

Additional experiments were done on how transparency can be mapped to entire streamlines based on a scalar quantity at the streamline seed point in order to improve vortex visualization. The ideal visualization of a vortex tells the user more than just

where its core is located. It is better to be able to see the speed of the fluid at multiple levels moving outward from the core. This can cause occlusion problems when using streamlines, however visualizations can be greatly improved by mapping transparency to streamlines started at different layers of the vortex. An example of this is shown in Figure 4.26.

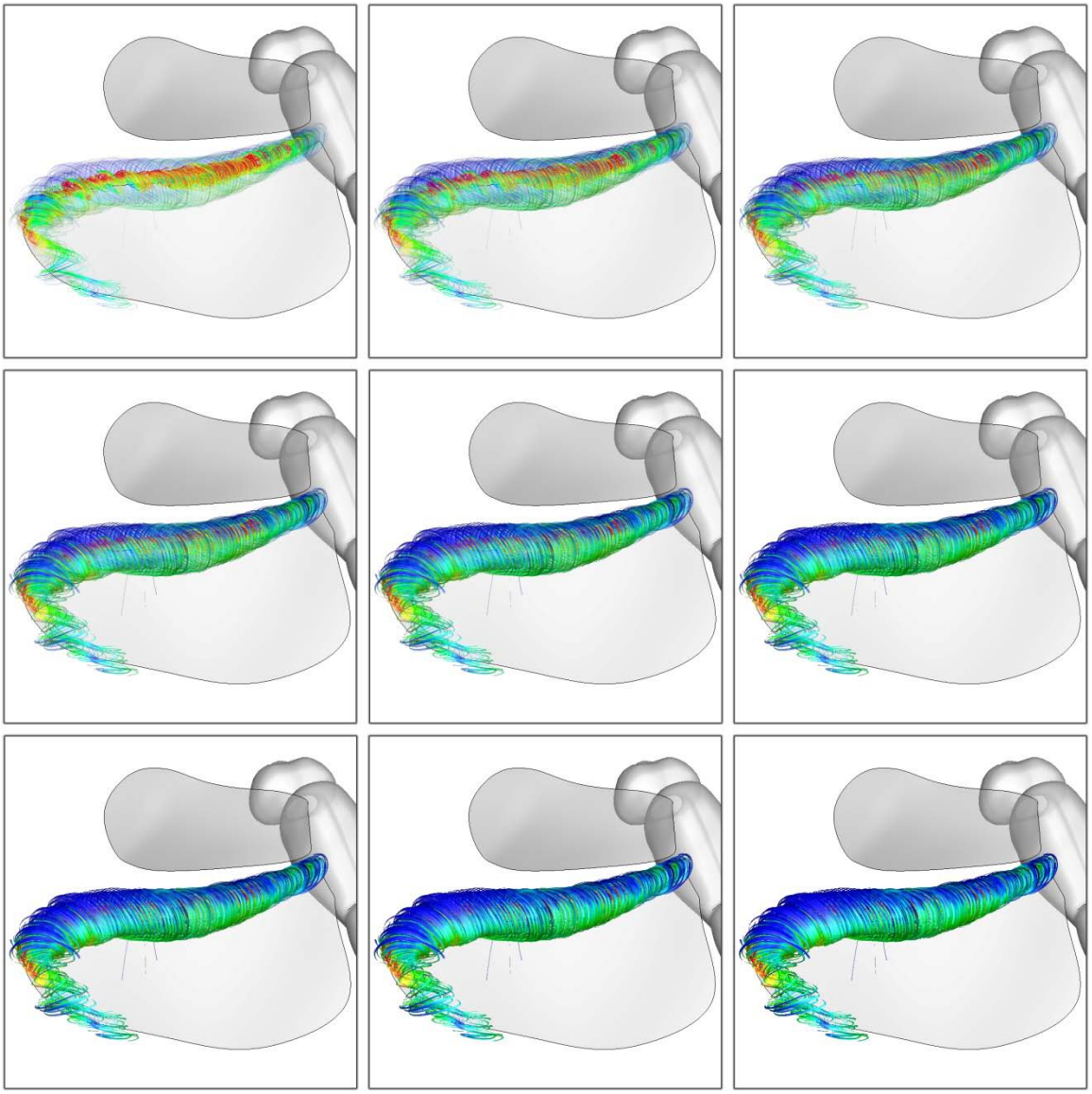

Figure 4.26: Two groups of streamlines seeded based on different iso-values. The transparency of the outer group of streamlines is incremented by 10% in each image.

4.7.1 Decoupled Particle Tracing and Rendering

While high quality renderings with global illumination and ambient occlusion can be slow, tracing hundreds of particles over thousands of time steps is an even more time consuming endeavor. However, since the base seeding algorithms are established it is possible to create an interactive visualization by pre-computing all possible particle traces that lie within the bounds of the seeding algorithm. With the particle traces stored on disk, the visualization interactivity is limited solely by rendering speed and not by particle tracing speed.

5 Results

While it is necessary to validate one's methods, the purpose of flow visualization research is not to analyze flow around a steady sphere data sets. Flow visualization must not resemble a nebulous blob of spaghetti when applied to the most complex data sets but rather it should elucidate the critical features. For the purposes of this work, the methods explained in Section 3 were used to study deformable wing insect flight.

The unsteady flow fields generated by a quad wing insect are very complicated and not well understood, which makes it an ideal test of a visualization algorithm's merit. The different areas within the flow induced by the wings require multiple seeding strategies to properly capture them all. This section presents an overview of how generalized streak seeds, streamlines, particle advection, vertex normal seeds and flowing seed points were used to visually capture flow features believed to play a role in flapping flight.

5.1 Dynamic Seed Curves and Streamlines

In section 3.4 the concept of dynamic seed curves was introduced to help automate seeding flow objects over multiple time steps in complex unsteady 3D flows. Results of using iso-seed planes and vertex normal seeds to place streamlines in the deformable wing dragonfly data set are presented here. The goals are to keep the streamlines in the leading edge vortices, get uniform coverage of the different layers within the vortices that are rotating at different speeds, and minimize the amount of user interaction needed to choose the seed points.

5.1.1 Iso-Seed Plane Results

The iso-seed plane technique is one way to restrict seeds to a vortex forming along the edge of a moving object in the flow. Figure 3.18 and Figure 3.19 show results of generating the planes and using them to detect the leading edge vortex core along each wing based on a series of iso-surfaces. Each iso-plane corresponds to a mesh point on the dragonfly wing. In order to capture all the layers of the vortices, seeds are placed on each plane at different iso-values at the point closest to the corresponding mesh point on the wing. Figure 5.1 shows the result of placing streamlines at different iso-values.

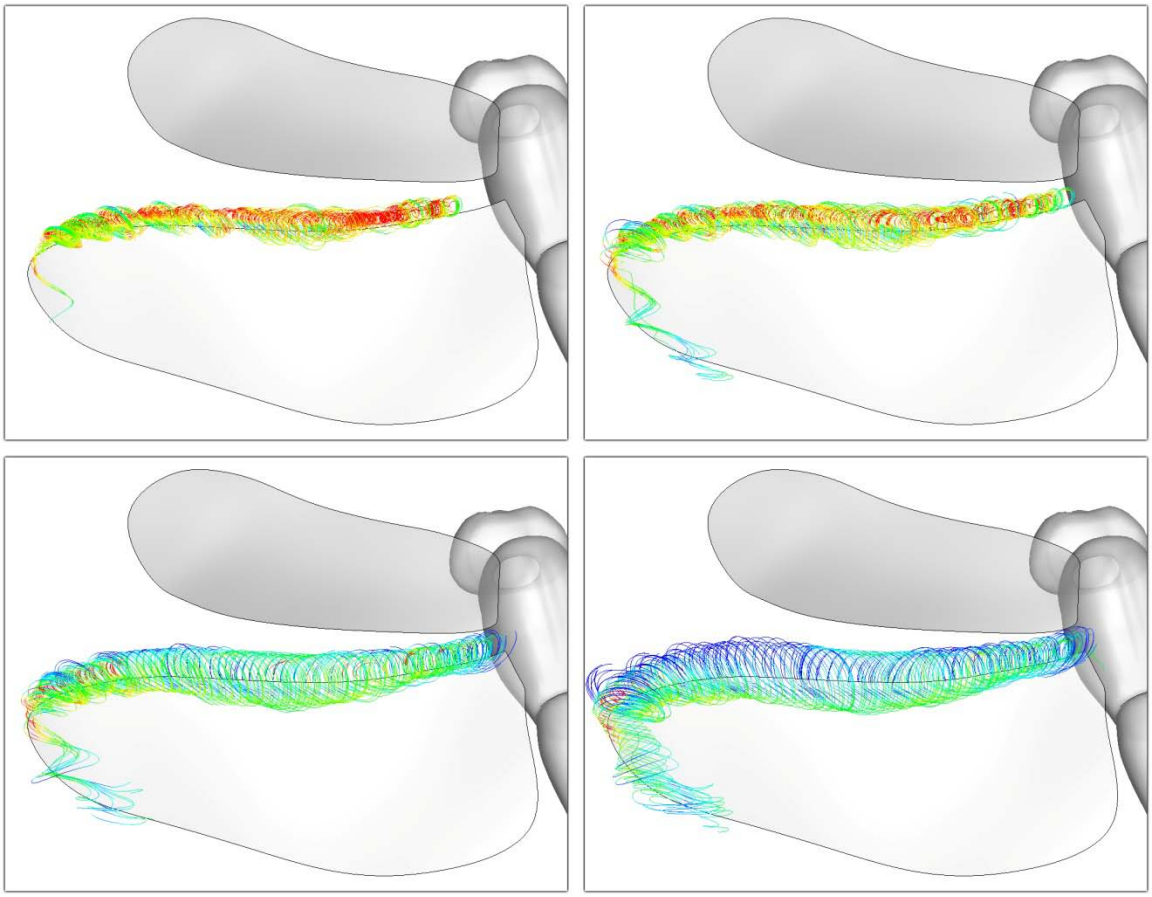

Figure 5.1: Streamlines seeded along the leading edge vortex at four different iso-values.

This seeding method just starts streamlines in key areas, however it does not prevent them from leaving those areas and potentially self occluding if integrated to long. My experiments have shown that with this kind of data it is more effective to densely seed streamlines in the most important vortices but not integrate them very long. The results of placing seeds in the leading edge vortex and integrating them for a varying number of time steps can be seen in Figure 5.2. Ultimately the iso-seeding method proved very effective for streamline placement, as can be seen in Figure 5.3

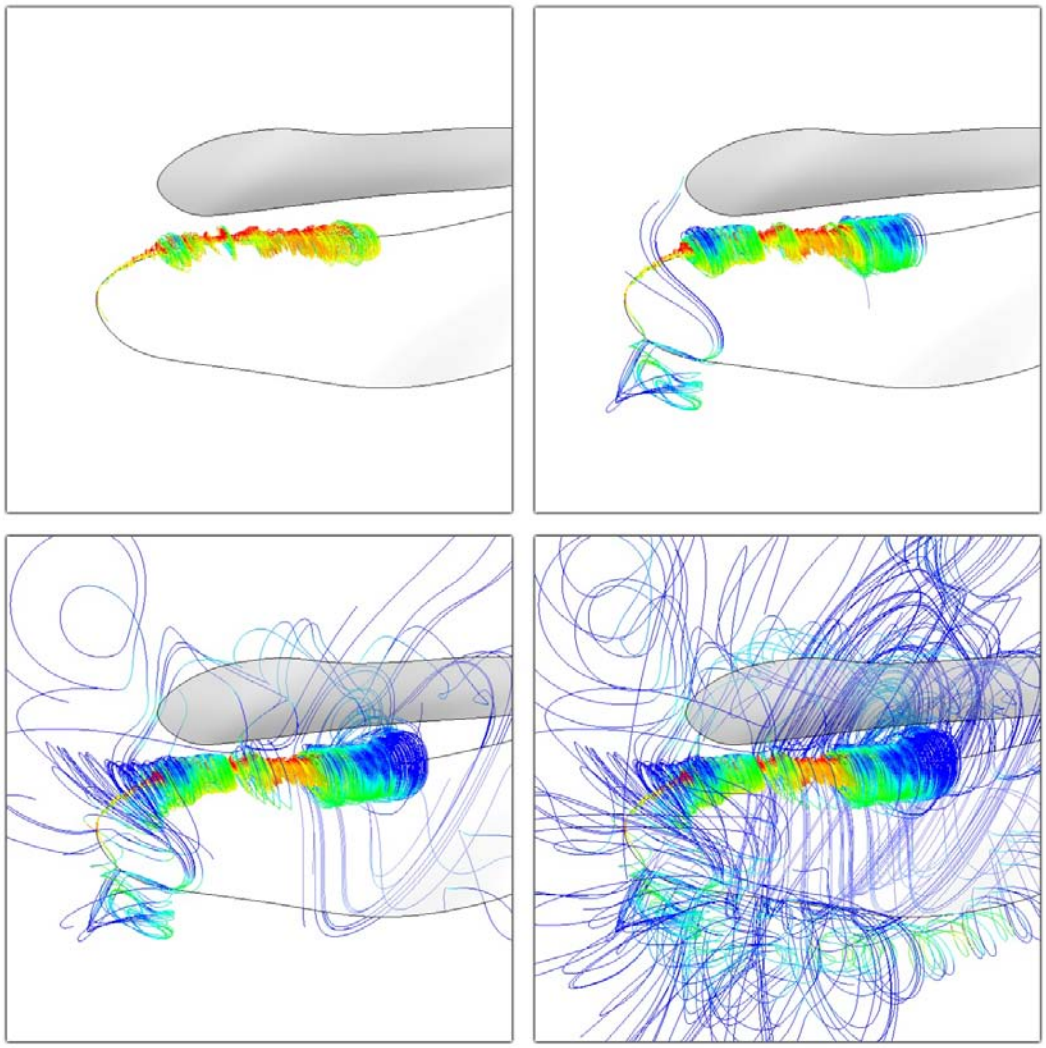

Figure 5.2: Effect of integration time on streamline quality.

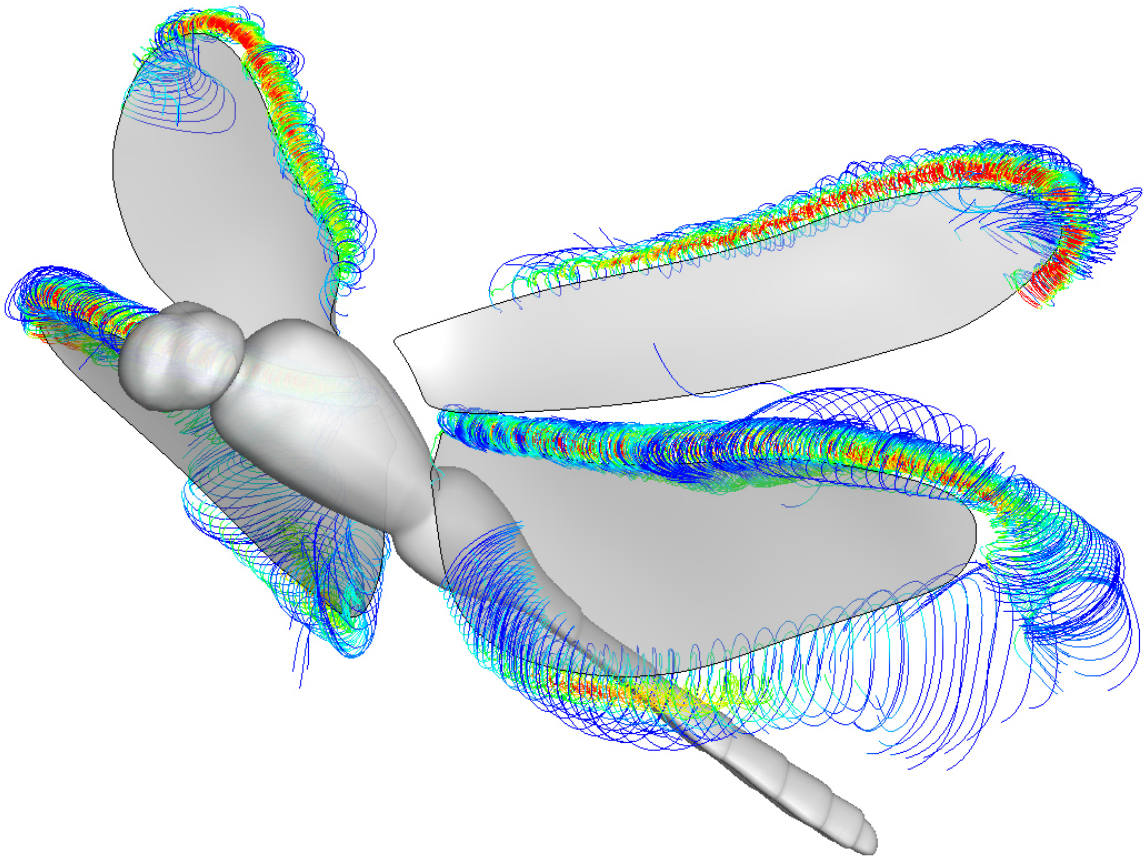

Figure 5.3: An effective visualization based on dense iso-seed placement and low integration times.

5.1.2 Vertex Normal Seed Results

Despite the effectiveness of iso-seeds, they have a few drawbacks. The main issue is that they are dependent on the results of the vortex core detection algorithm. In complex data sets there will often be multiple vorticity magnitude peaks in the iso-planes due to the presence of multiple vortices in the flow. For example, in the flapping disk data set the detected vortex core jumps from the leading edge into the dynamic stall vortex when the angle of attack changes rapidly. This can cause seed points to blink around in the flow domain. If the seed points do not move smoothly between time steps it will compound the spatial coherency problems that streamlines have when animated.

Unlike iso-seeds, vertex normal seeds remain a consistent distance above the mesh along the normal vector of the corresponding vertex. For this reason, vertex normal seeds are generally more effective at more time steps of the flapping wing data sets. The only problem with vertex seeds is that it is up to the user to choose a distance along the normal vector to place the seeds such that the desired flow features are captured. However, this can be chosen based on the average vorticity along all the normals of the selected vertices. Figure 5.4 and Figure 5.5 show several vertex normal seeds on the dragonfly wings and the corresponding streamlines.

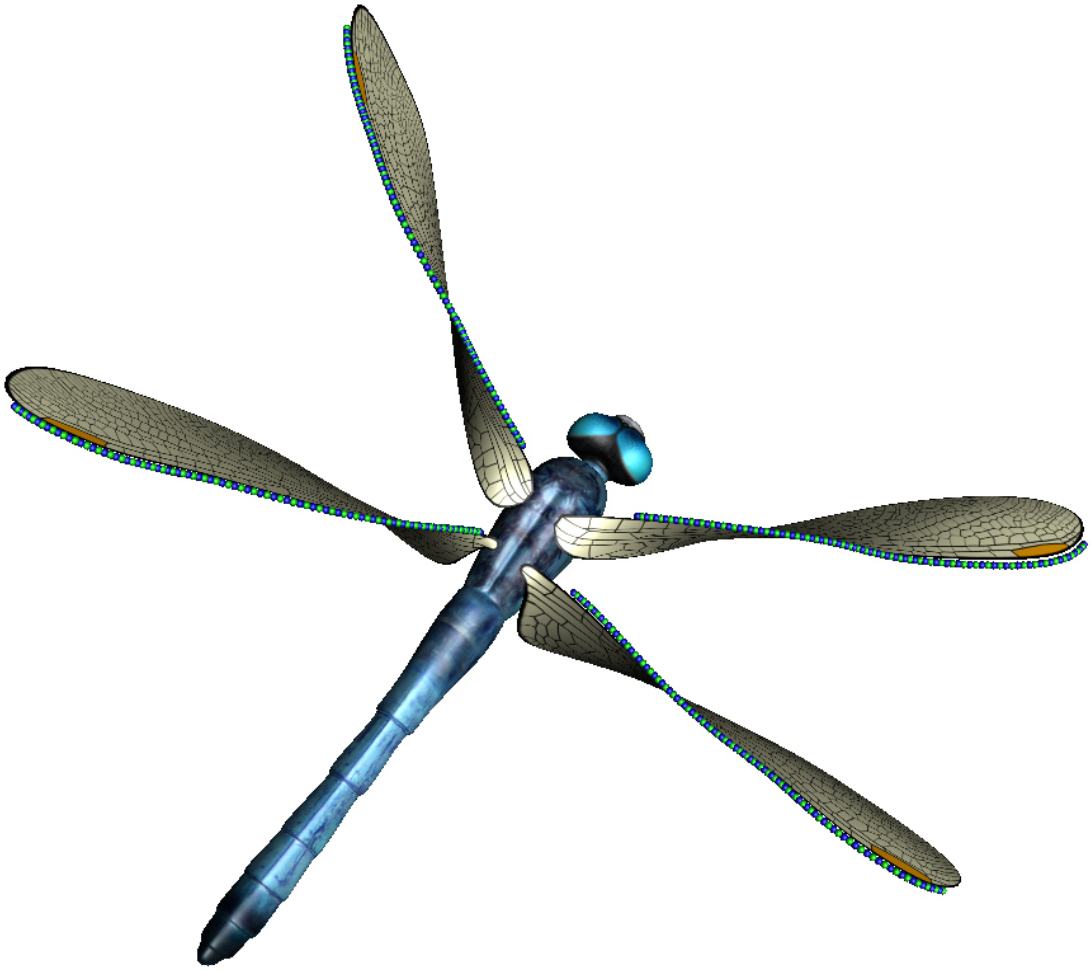

Figure 5.4: Vertex normal seeds represented as spheres on the leading edge of each wing from the root to the pterostigma.

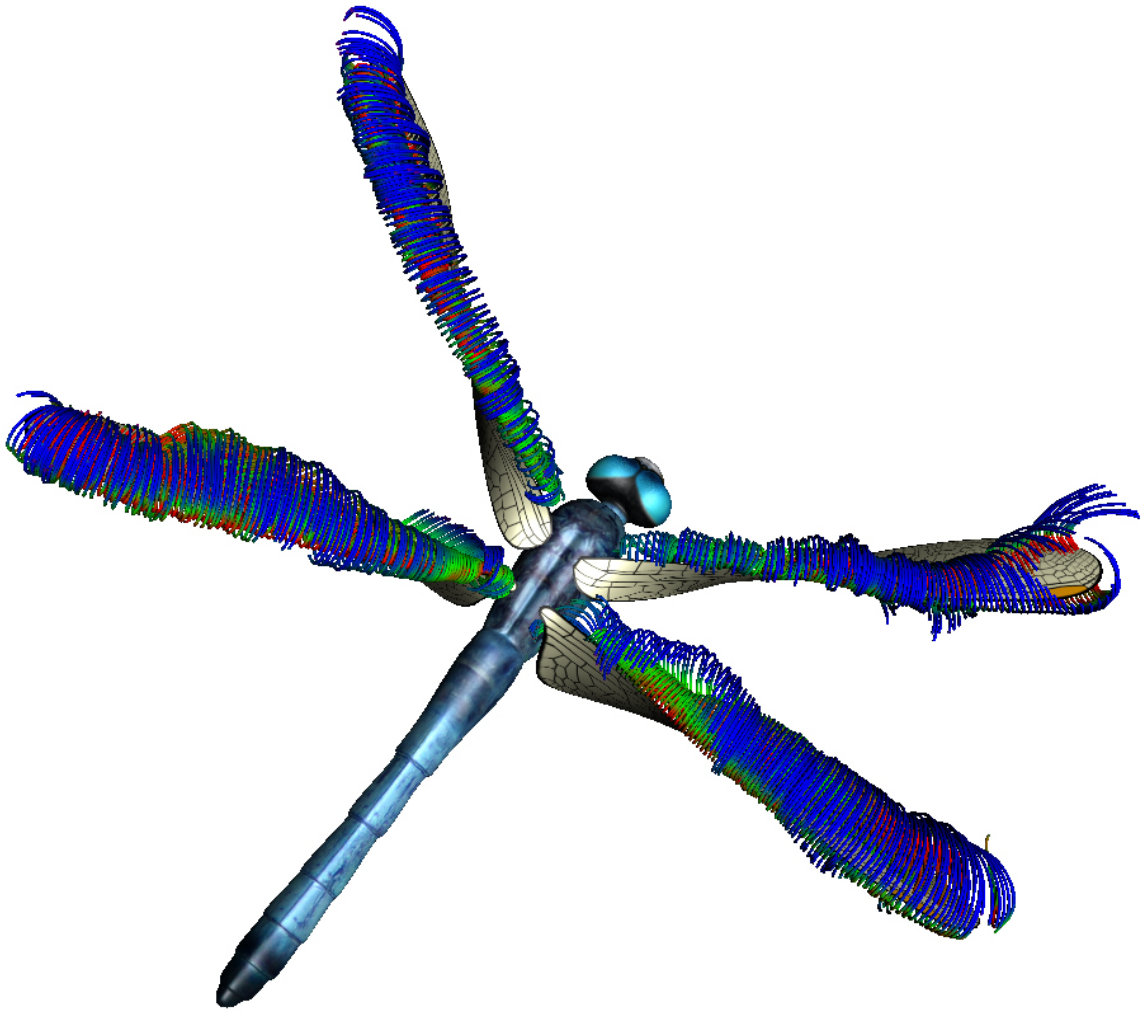

Figure 5.5: Streamline tubes with color mapped vorticity generated with vertex normal seeds on the leading edge of each wing.

5.2 Dynamic Seed Curves for Particle Emission

The drawback of using dynamic seed curves with streamlines is that you must balance the tradeoff between flow coverage and spatial coherence when animating your visualizations. Using dynamic seed objects makes seeding much easier from a user's perspective, but it does limit the area in the flow where seeds will be placed. In general this is a good thing, but other useful flow phenomena could potentially be missed if the

streamlines are not integrated long enough. For instance, in the case where seed points are bound to the leading edge, it is possible to see when the vortex sheds because it is no longer present during stroke reversals, however you cannot visualize where it goes. Also, the downwash or induced flow resulting from the near field vortices is not captured with this visualization method.

These problems are mainly caused by the nature of streamlines. Thus, dynamic seed curves were also tested for particle emission and generalized streak line seeding. Particle advection over multiple time steps allows the particles to move into areas of interesting flow where it would be difficult to automatically place seeds. In particular, particles can capture how the flow moves from the vortices near the seed curves into the wake, thus allowing the dragonfly to take off.

5.2.1 Particle Lifetime

Particle lifetime controls how long a particle will stay in the scene before being removed. When using dynamic seed curves, the particles are essentially guaranteed to at least enter the flow in the vicinity of an interesting feature. If there are too many particles kept in a scene after they are no longer in an important area it will draw attention away from those key areas. On the other hand a particle lifetime that is to short will cause particles to be removed while they are still in important areas. In the flapping disk and dragonfly data sets, particles were kept in the scene anywhere from 200 to 600 time steps after being emitted. This proved adequate to capture the induced flow but not float around aimlessly in the far corners of the data set.

5.2.2 Particle Emission Rate

The particle emission rate controls how close any given particle will stay to the particles emitted before and after it. If the particle emission rate is high enough the result is, for all practical purposes, a generalized streak line. True fully connected streak lines with adaptive refinement proved undesirable with the flapping wing data sets due to the fact that a complex mechanism to "tear" the streak lines when a flapping wing passes through them is necessary to avoid excessive stretching. Typically as the number of emission points goes up, the emission rate decreases to avoid business.

5.2.3 Iso-Plane Emission Points

The first particle emission test was done with iso-plane based seeding. Seeds similar to those shown in Figure 3.19 are used to place particles in the flow with a predetermined emission rate. The results of this are illustrated in Figure 5.6. Particles are color coded based on the wing whose LEV was used to emit them. The first thing that is obvious about these visualizations is that particles clearly are not as powerful as streamlines at conveying the precise shape of the vortices in the near field. They allow you to see the general shape of the large vortices shed during stroke reversal, but the tight shape of the leading edge vortex is missed for the most part. On the other hand, particles do show the wake structure resulting from the vortices and the direction the flow is moving. The downwash is what allows the dragonfly to take off, so this is also a valuable flow phenomena that needs to be captured. In Figure 5.6, the dragonfly begins to fly backward while taking off, which is why the majority of the wake moves forward and down after leaving the wings.

Figure 5.6: Results of using iso-seed plane based vortex core detection for particle emission.

5.2.4 Vertex Normal Emission Points

Vertex normal based seed curves were also used to inject massless particles into the flow over time. Despite the ease with which this method allows a user to define seed curves, it requires knowledge of the application domain in order to really be effective. Based on the literature review of all the major flapping wing insect flight studies

presented in Section 2.5, I choose several target areas to test vertex normal seeds on the dragonfly wings. In particular, the wing roots were chosen in order to capture any potential spanwise flow (Figure 5.7). Also the vertices along the leading edge were chosen to capture both leading edge vortex formation as well as the shedding of vortices into the wake. Finally, multiple seed curves were placed near the wing tips on both the fore and hind wings to look for wake capture in an area of high force production.

Figure 5.7: Vertex normal seeds placed at the wing roots used for particle emission.

While it is clear that the LEV is responsible for the lift generation produced by insect wings, it is still unclear how the vortex stays attached to the wings so long. In 2D simulations of a foil with a comparable angle of attack, vortex shedding occurs much earlier. Obviously in the 2D case, flow cannot move parallel to the vortex core, so one potential explanation is that spanwise flow moving along a spiraling vortex drains energy away from the vortex core, thus stabilizing the LEV. This theory is based on the phenomena that occurs in delta wing aircraft. The results of placing particles on both the leading edge and the back of the wings near the dragonfly's body are shown in Figure 5.7, Figure 5.8 and Figure 5.9. While there are signs of spanwise flow in the particle trajectories, particularly during the takeoff flap, it is not enough to say definitively whether or not this is the main reason why the LEV does not shed.

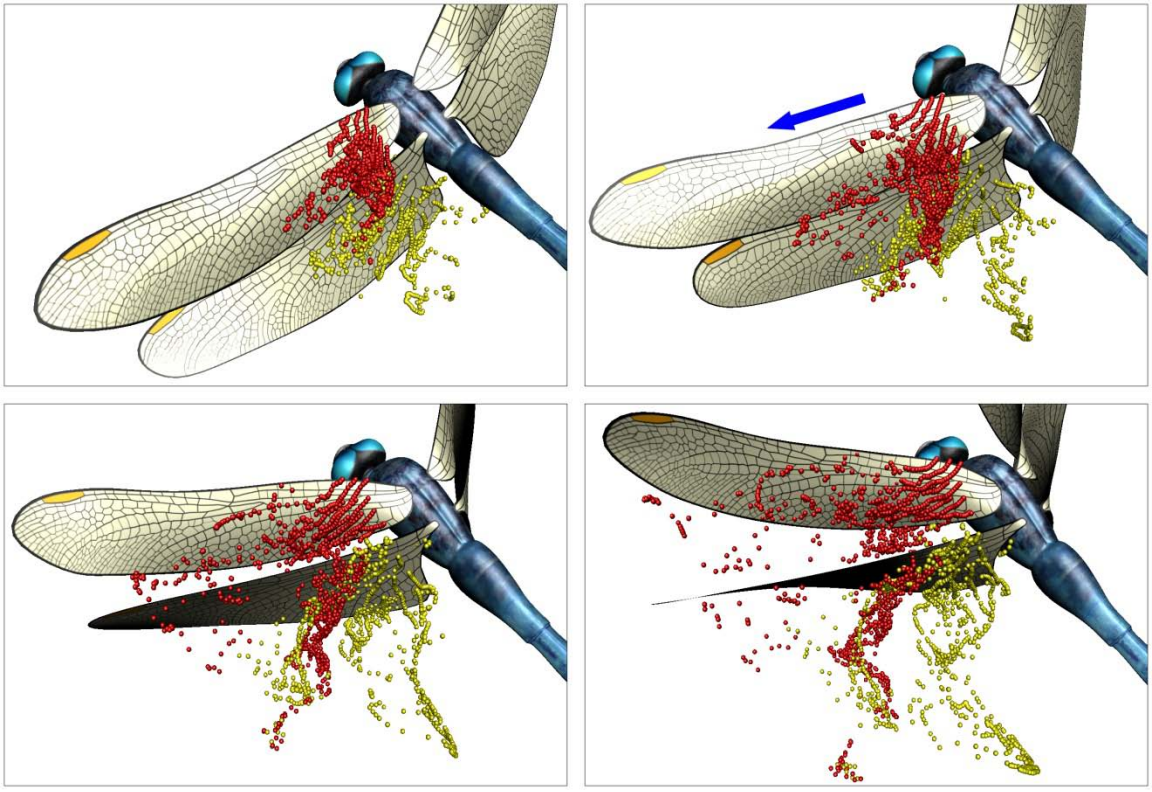

Figure 5.8: Spanwise flow along the left forewing and hind wing during takeoff.

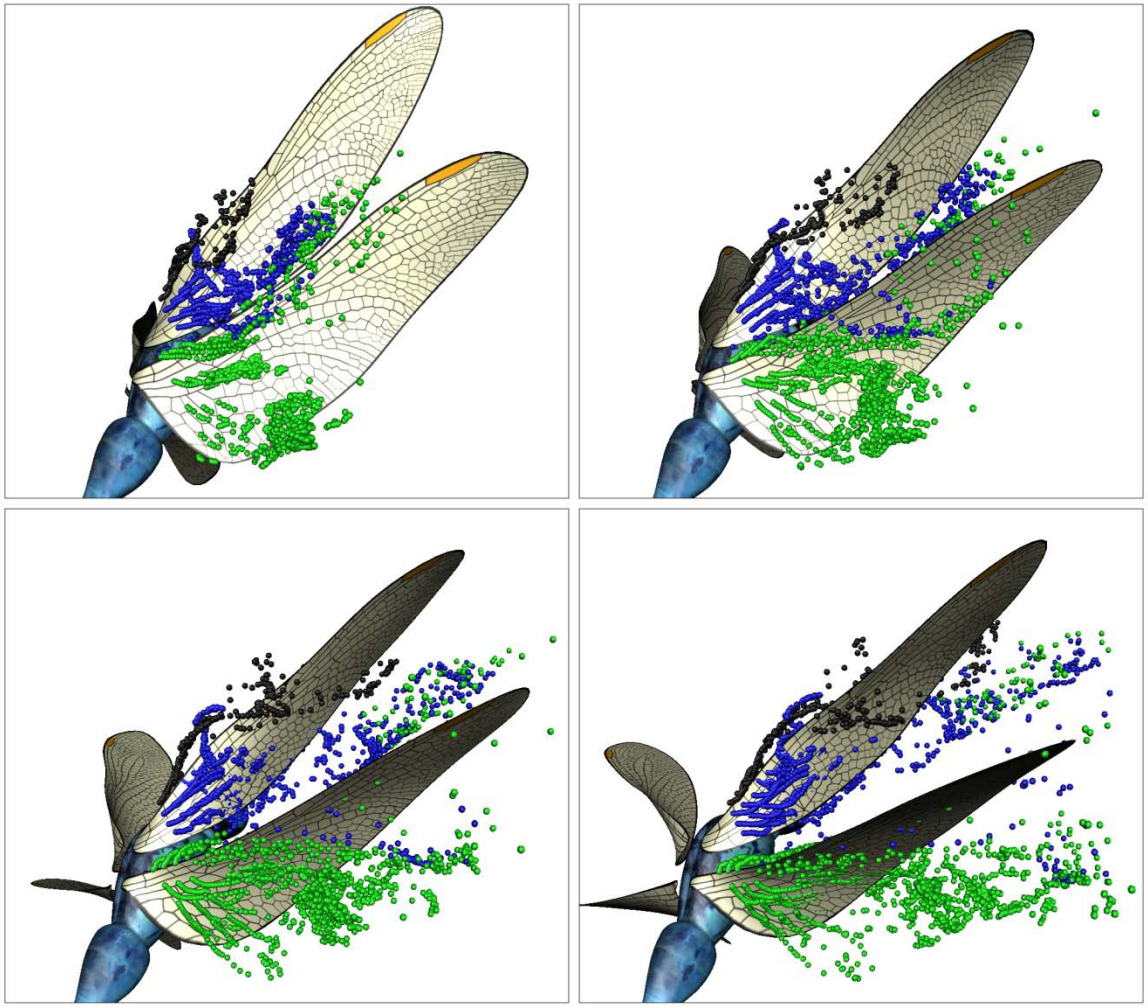

Figure 5.9: Spanwise flow along the right forewing and hind wing during takeoff, with the LEV highlighted in black.

The leading edge was the next obvious place to try emitting particles, as can be seen in Figure 5.10. Overall, the results look similar to the iso-plane seeding results, however the smooth movement of the seed curves keeps particles that were emitted at adjacent time steps closer together. This makes the flow features in the areas they move through slightly easier to discern. In general most particles tend to stay close to the leading edge during the up stroke and down stroke except those near the wing tips.

Particles are then shed from the wings during stroke reversal. While the particle behavior is essentially what is expected based on flapping flight theory, perceptually they have drawbacks. Particles are more effective for capturing the induced flow, however the structure of the vector field near the tight leading edge vortex is not as clear as it was with streamlines. Capturing both of these benefits is the motivation behind applying the flowing seed point method to flapping flight data.

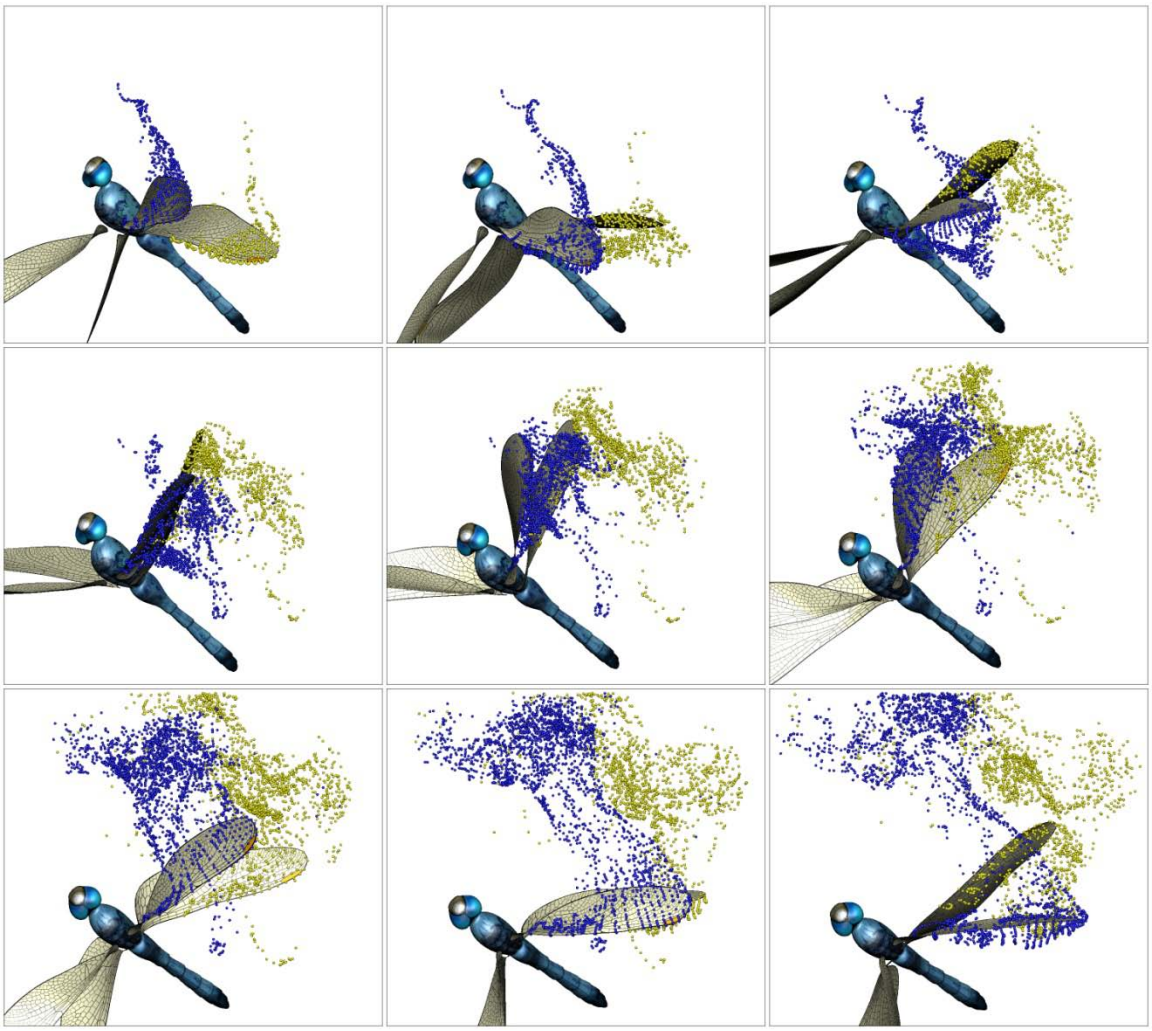

Figure 5.10: Particles emitted from several seed curves that follow the leading edges of the right fore and hind wings.

5.3 Flowing Seed Points and Dynamic Seed Curves Combined

Thus far I have visually captured the existence of the leading edge vortices that occur during both half strokes with streamline seeding and the induced flow structure has been captured with particle advection. It is still unclear what happens to the leading edge vortices when they are not obviously attached to the wing. Thus, the goal of applying flowing seed points to this data set is to track the movement of near field vortices without losing the ability to visually follow the path of individual massless particles.

Initial tests were done to determine suitable emission and integration parameters for the dragonfly data set. Next, flowing seeds were emitted from several of the same locations that streamlines and particles were placed in Sections 5.1 and 5.2. A camera animation method intended to complement flowing seed points as well as a wing chord seeding method that aims to increase seed density in a plane perpendicular to the LEV are then introduced.

Examples of situations where the flowing seed point method captures features that other visualization methods would be unable to capture are given specific attention. For example, the movement of near field vortices as they are shed at stroke reversal, as well as wake capture between wings are given specific attention. Like particle advection, flowing seed points are effective in still images but they are intended to be animated. Images are presented in groups at neighboring time steps to help convey how the visualization evolves over time. Finally, to tie the visualization results back to the simulation analysis in Section 4.3 the visualizations are compared with the lift generation and the camber to chord ratio for each of the left wings for several time steps where the lift production is at a local peak.

5.3.1 Flowing Seed Parameters

When using dynamic seed curves for particle placement the rate at which particles were emitted into the flow and the time they stayed there played a significant role in how the resulting visualization would look. These variables need to be treated differently with flowing seed points because of the overall larger number of polygons in the scene and the greater possibility of occlusion. In the case of particle emission you must balance between being able to discern which particles were emitted at consecutive time steps while not crowding the scene with too many streamlets. Figure 5.11 gives a comparison of several particle emission rates with the same underlying seed curve.

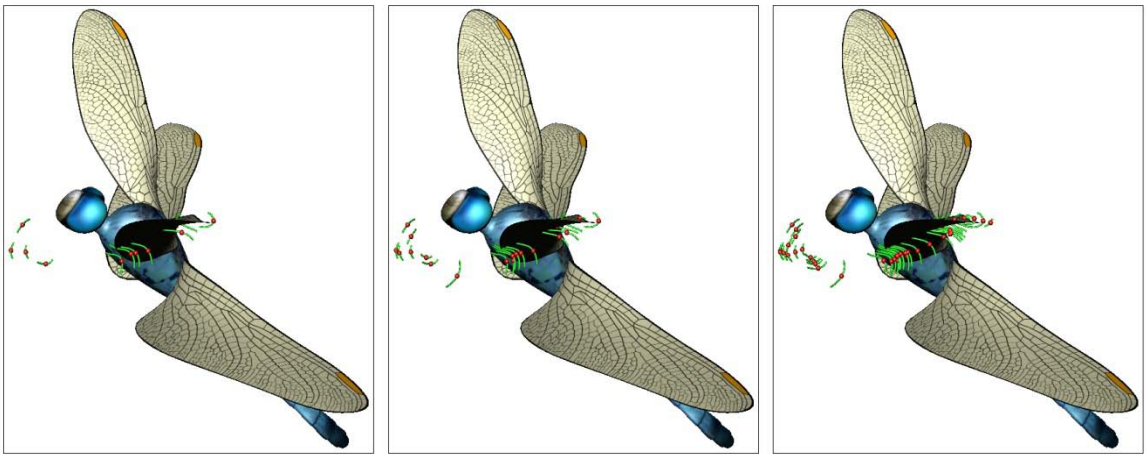

Figure 5.11: Comparison of flowing seed particle emission rates.

In addition to worrying about the particle lifetime and emission rate, when dealing with flowing seeds you must control how long each individual flowing streamlet or pathlet is integrated. If they are not integrated long enough then there is really no advantage over basic particles. If they are integrated to long the scene can become very

busy and the animations will begin to look choppy. A comparison of several different integration times is shown in Figure 5.12. In general it proved effective to integrate the streamlets and pathlets anywhere from 8 to 32 time steps in each direction depending on the density of the emission points and the particle emission rate. While a higher streamlet integration time appears better in a still image, it can become perceptually disconnected from its own flowing seed point when animated.

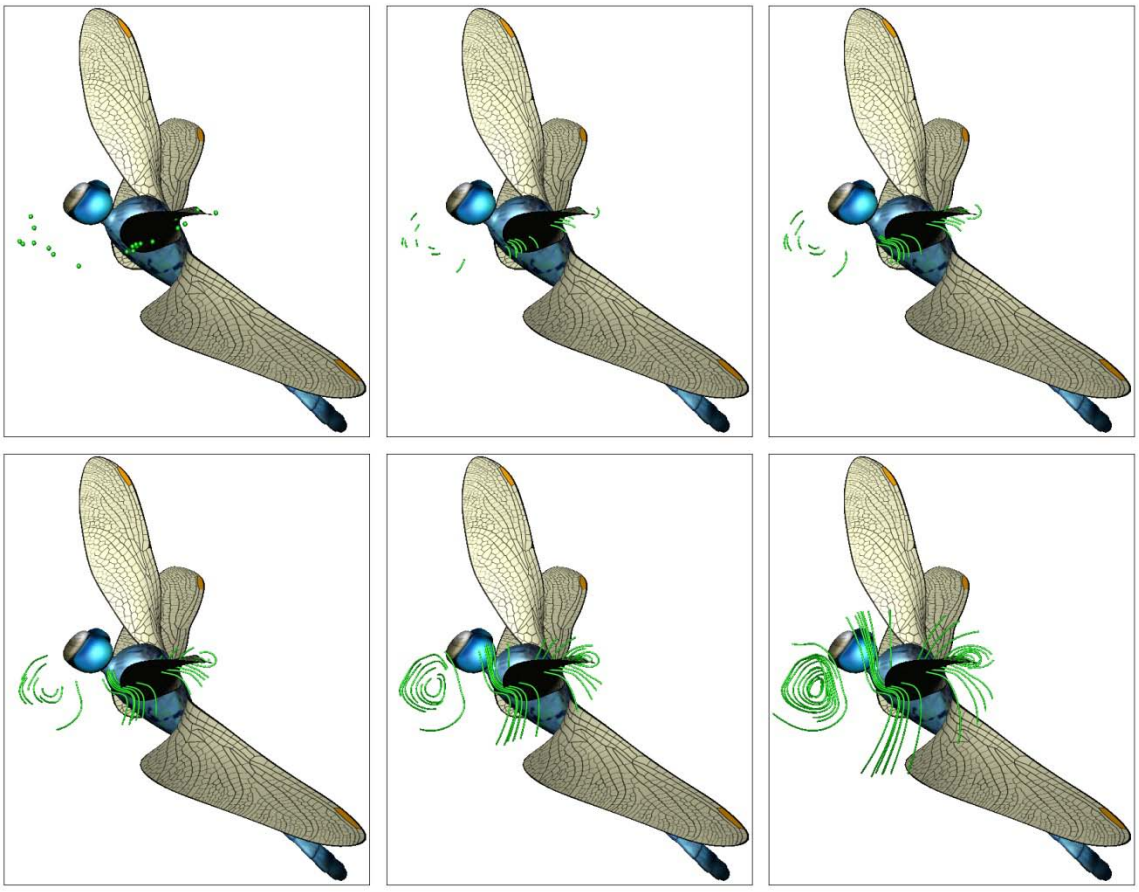

Figure 5.12: Comparison of the effect of integration time on flowing streamlets.

5.3.2 Flowing Pathlets to Capture Vortex Formation

In general streamlets proved more useful with flowing seed points than pathlets due to their ability to capture vortices in the instantaneous flow. Pathlets however have the unique ability to provide information about a range of time steps while looking at a still image. In particular they can capture where in space time the leading edge vortex for any given wing formed by showing the path taken by particles emitted earlier in time. This allows a user to see how long after stroke reversal the LEV on any given wing formed as well as where it moved while examining the alignment of the wings at a point later in time. This effect is illustrated in Figure 5.13. As the number of flowing seeds increases it becomes very difficult to capture this effect, so for the majority of the tests with the dragonfly data set, streamlets were used instead of pathlets.

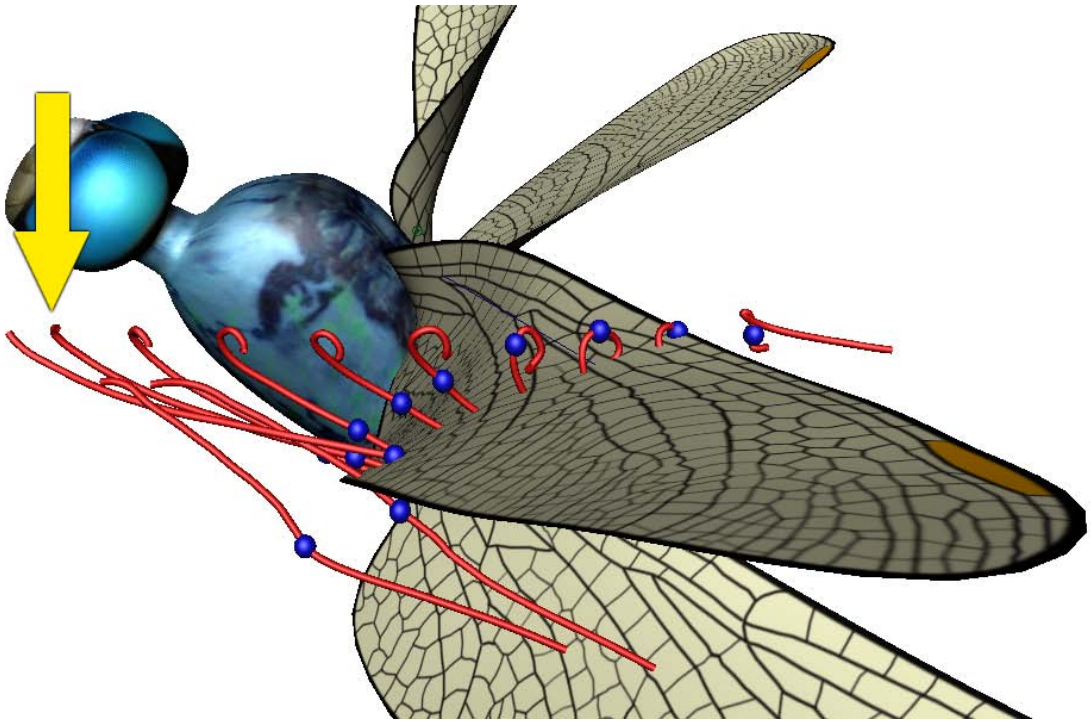

Figure 5.13: Vortex formation time and location captured in neighboring time steps with flowing pathlets.

5.3.3 Seeds Flowing off the Leading Edge

I have already demonstrated the ability to capture the existence of the leading edge vortex as well as the far field induced flow. What has not been captured is the shedding, breakdown and reforming of the LEV, so that is the main goal of applying flowing seed points. Their ability to move with the vortex as it sheds while still showing the vector field trajectories makes them well suited for this task. When placing streamlines and particles in the dragonfly dataset it proved very effective to focus on the wing roots, tip and leading edge (Figure 3.22). Naturally the same base seed curves were used in the first attempts at using flowing seeds in the dragonfly data set. An example of these seed curves is shown in Figure 5.14 and the results of inserting flowing seed points into the flow at the root, tip and leading edge of the right side wings are shown in Figure 5.15, Figure 5.16, Figure 5.17 and Figure 5.18.

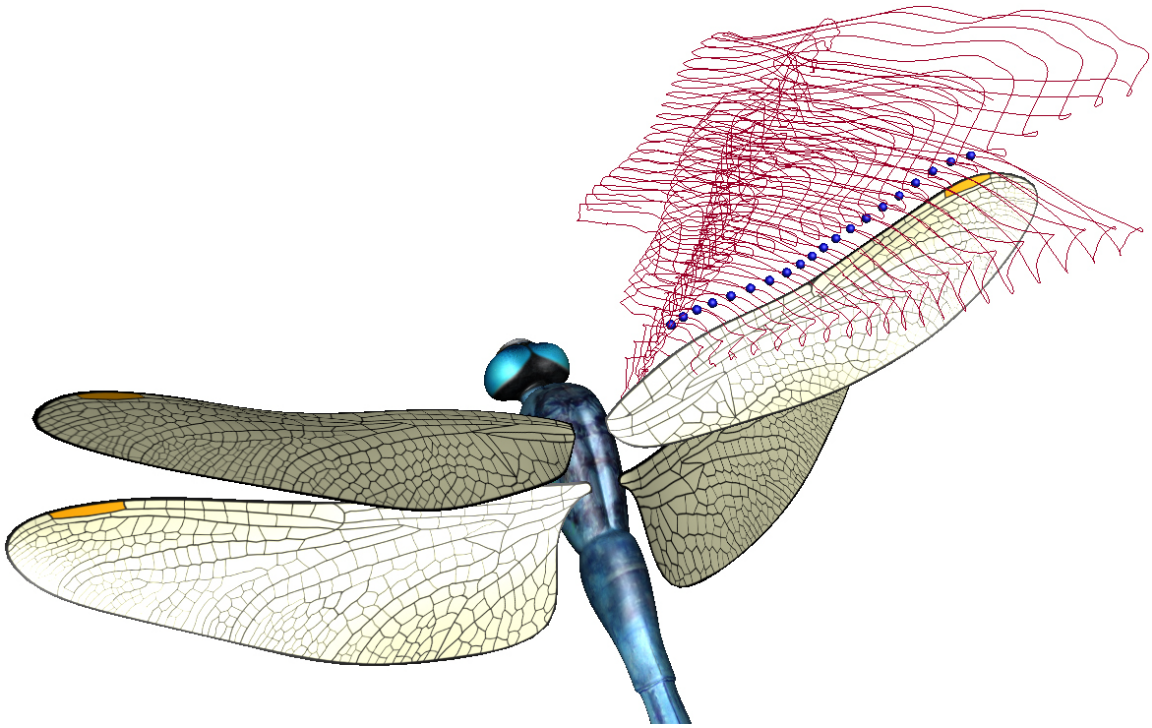

Figure 5.14: Seed curves generated on the normals extending from the leading edge vertices.

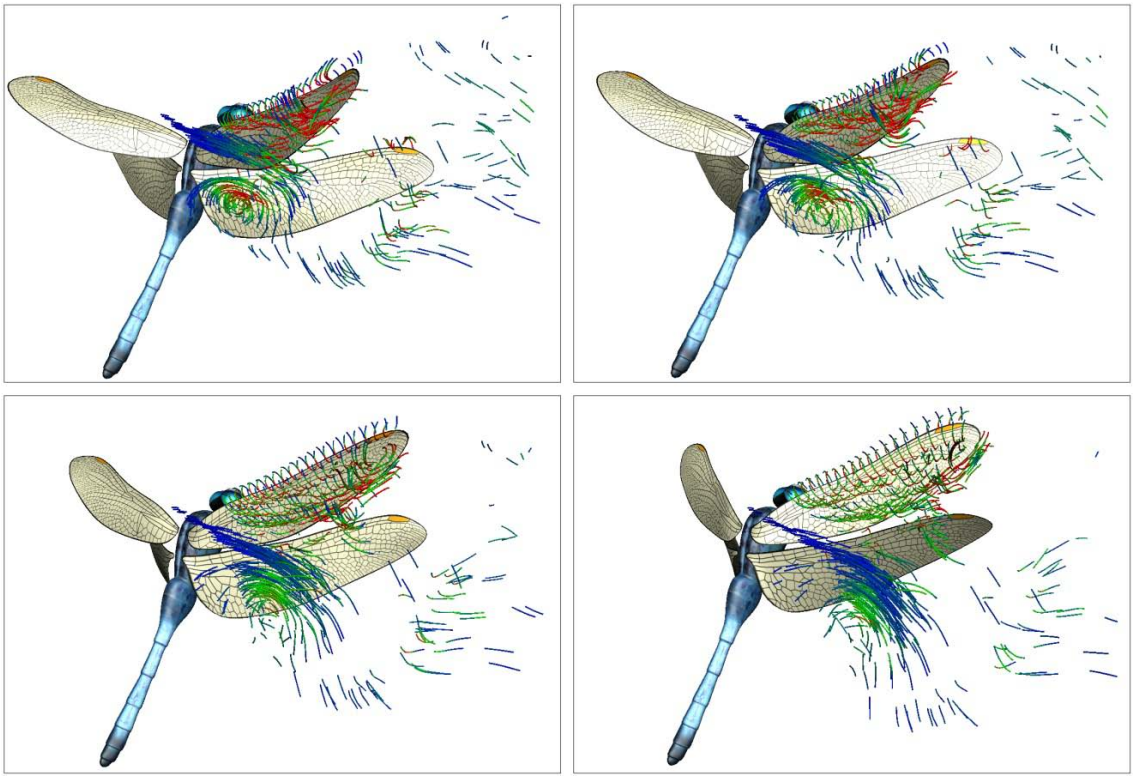

Figure 5.15: Vortex shedding and dissipation captured with flowing streamlets.

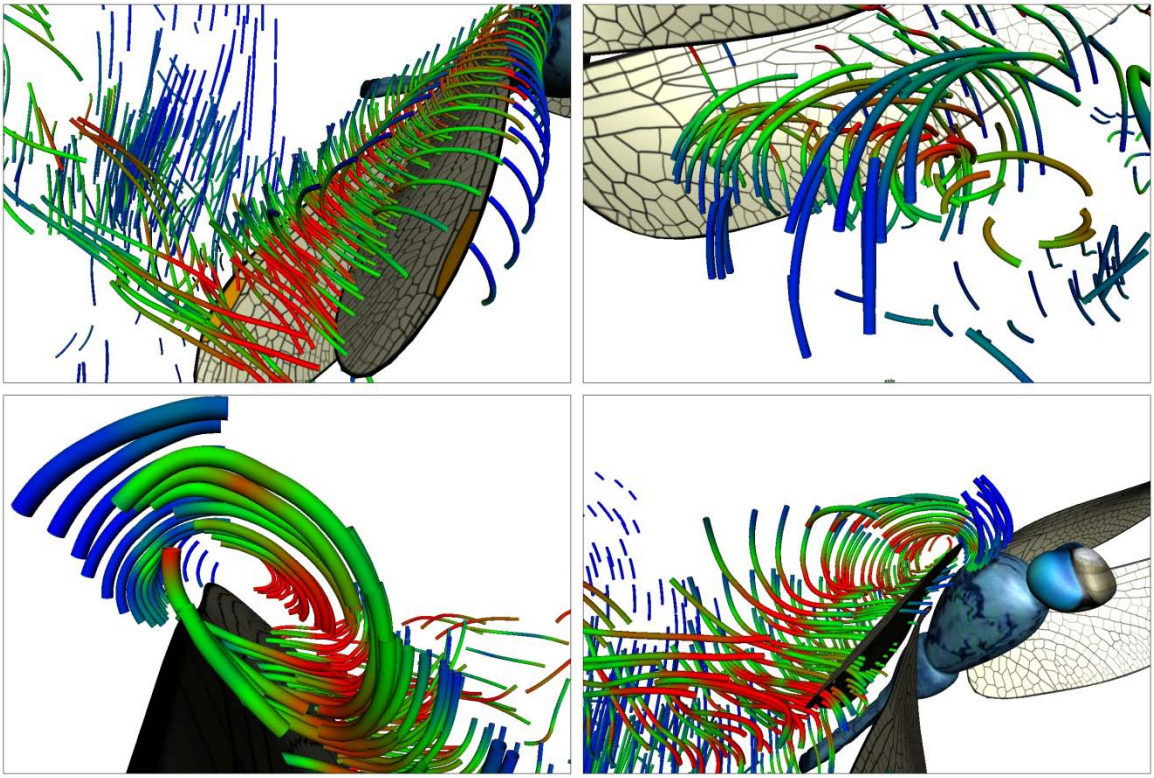

Figure 5.16: Several close ups of vortices captured by the flowing seeds emitted from the leading edge.

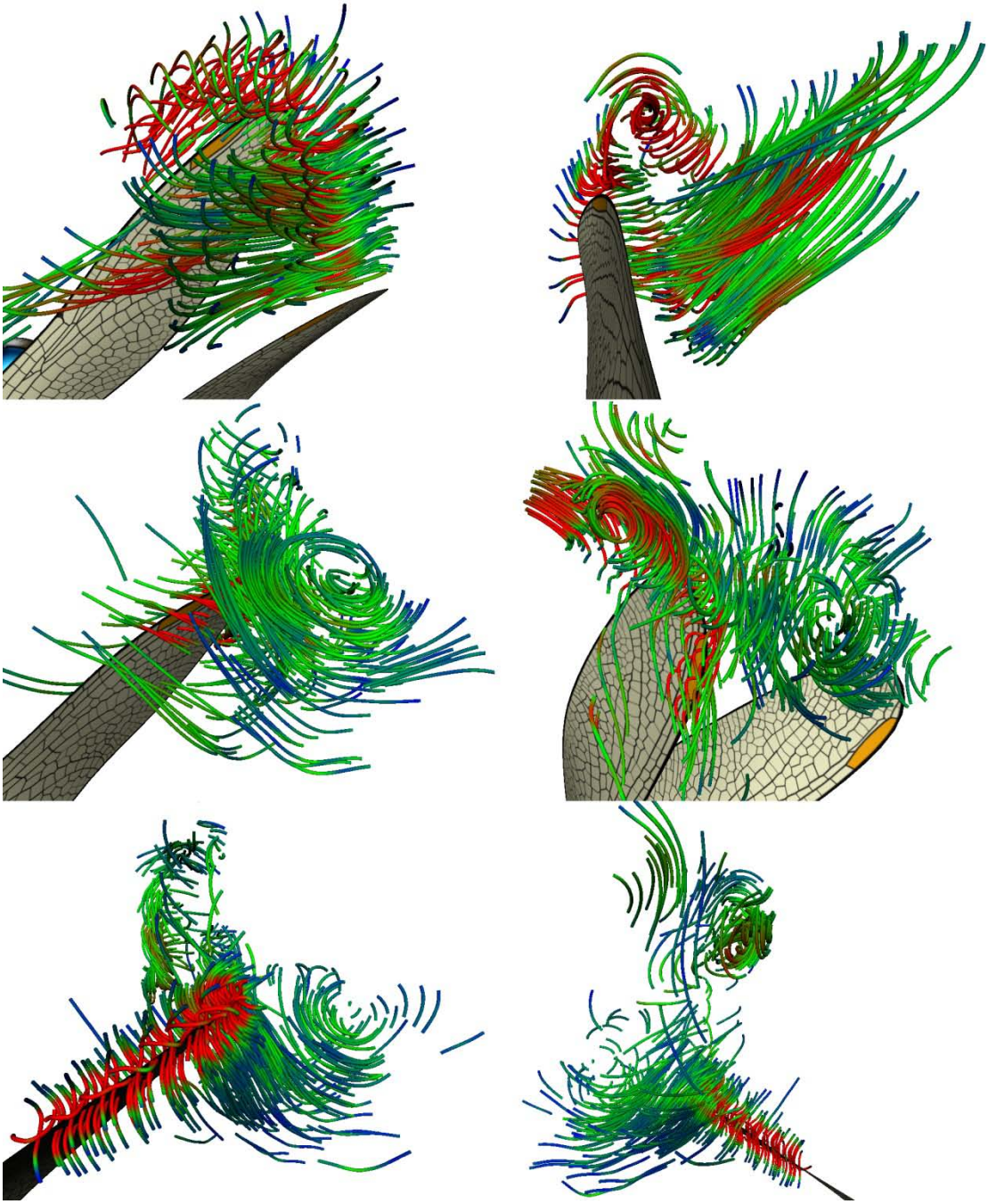

Figure 5.17: Flowing seeds emitted along vertex normal based curves at the tip of the right forewing.

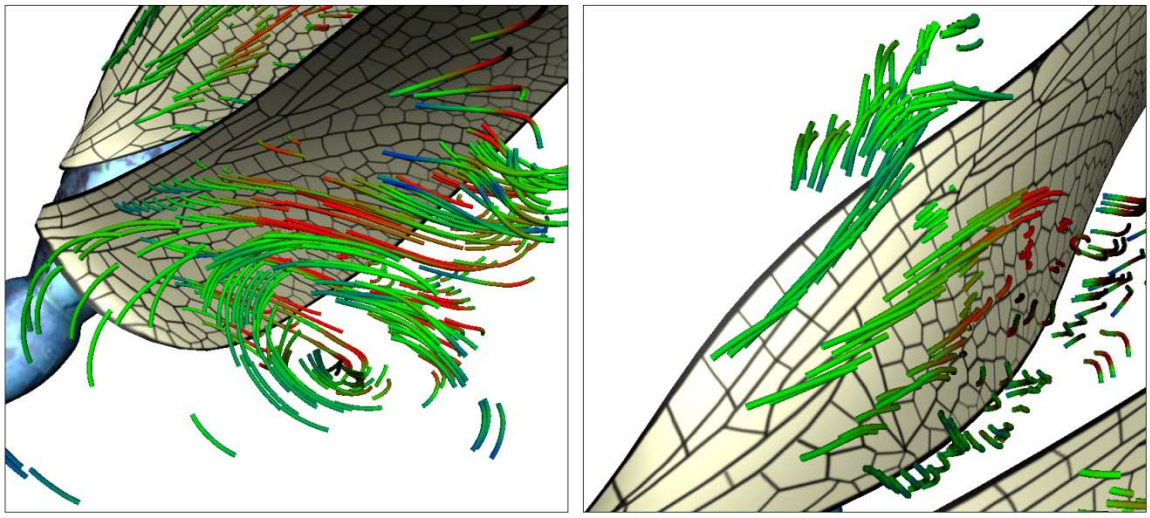

Figure 5.18: Flowing seeds emitted from the wing roots capture vortex shedding off the trailing edge as well as spanwise flow along the leading edge.

5.3.4 Wing Mounted Cameras

These first tests using flowing seed points with the flapping wing dragonfly data set suggested that the same seeding and viewing strategy that worked for streamlines, and particles was no longer ideal. For example, when flowing seed emission points are placed in a single row along the leading edge they are not able to capture the entire LEV and the downwash suffers from information overload. While they did capture the structure and velocity of the induced flow and some of the shed vortices the majority of the particles did not end up in interesting areas of the flow and did not convey much more information than simple particles. Also, it required a lot of manual panning and rotating the scene to find an angle where the desired feature was not occluded.

These early tests suggested that perhaps emitting more flowing seed particles in a plane parallel to the wing chord and then keeping the view direction perpendicular to that plane might produce before results. The goal is to get a fairly dense seed distribution at a

cross section of the leading edge vortex and then keep the camera looking down the core of the vortex. In order to achieve the desired viewing angle without manually adjusting the camera, I chose to let the wing's motion drive the camera location. A vector is calculated for each wing by subtracting the root vertex from the tip vertex. The cameras are then placed on this vector a user defined distance from the wing tip at each time step with their "look at" points bound to the root of the corresponding wing. Figure 5.20 shows the position of cameras bound to each wing and Figure 5.19 shows the dragonfly through the left forewing camera at several time steps.

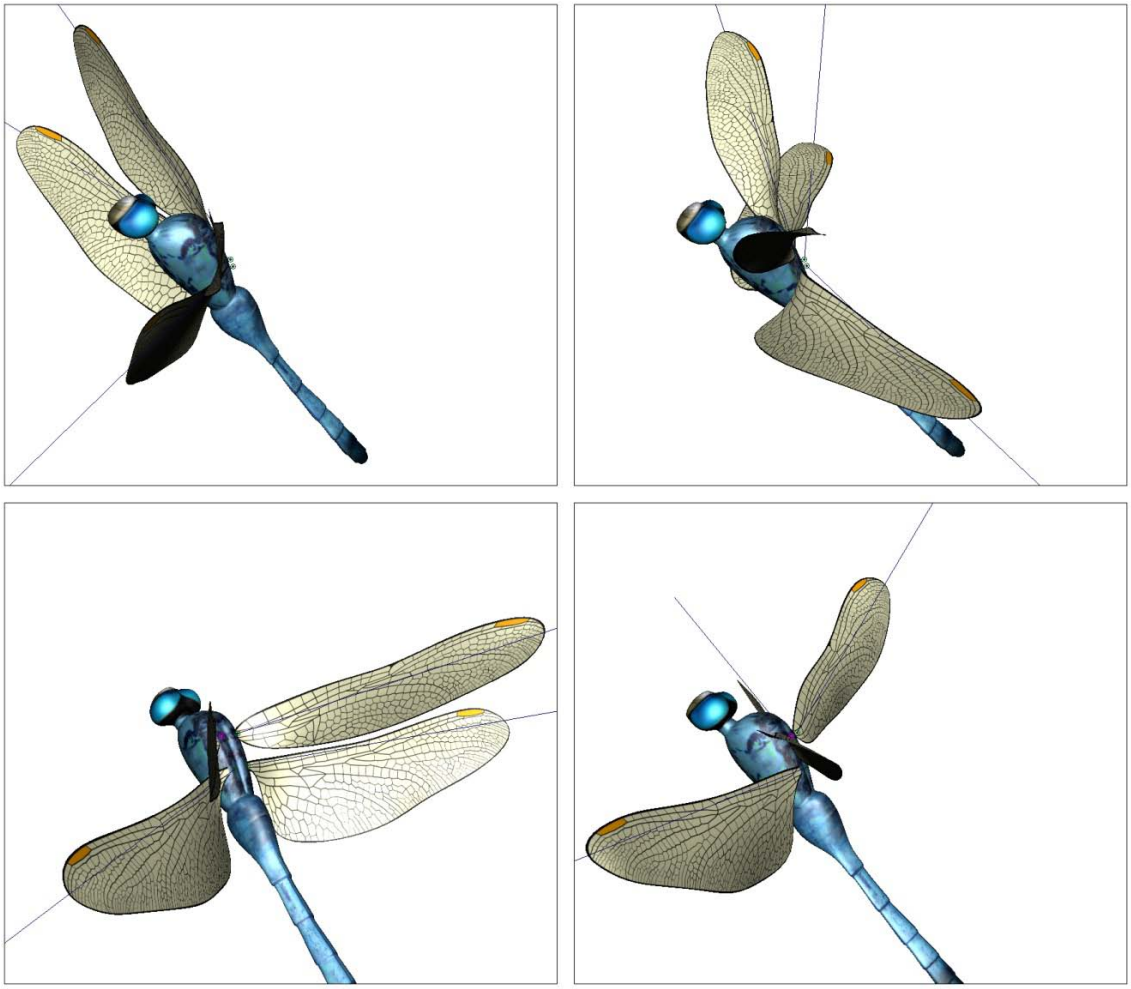

Figure 5.19: Result of viewing the dragonfly with a camera bound to the left forewing at four time steps during the same stroke.

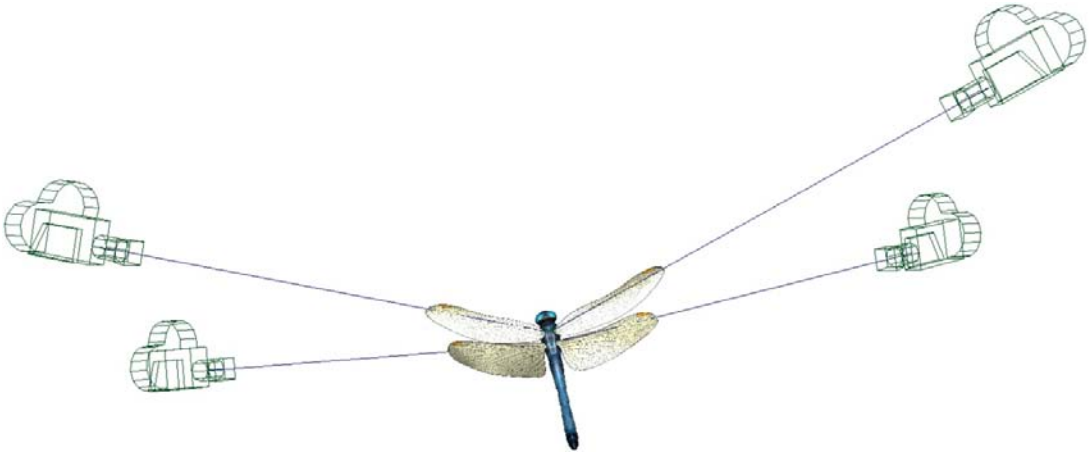

Figure 5.20: Cameras bound to all four of the dragonfly's wings.

5.3.5 Vertex Normal Seeding Along Wing Chords

A new seeding strategy known as chord seeding was also used to help further clarify the flowing seed point results. Vertex normal seeds are chosen along the wing chords and seed curves are created by connecting these points over time (Figure 5.22). Viewing long and slightly curved vortices, like those that occur on the wings, with this seeding scheme greatly reduces occlusion. Note that the wing chord emission curves and camera angles are really only effective with flowing seeds, as can be seen in Figure 5.21.

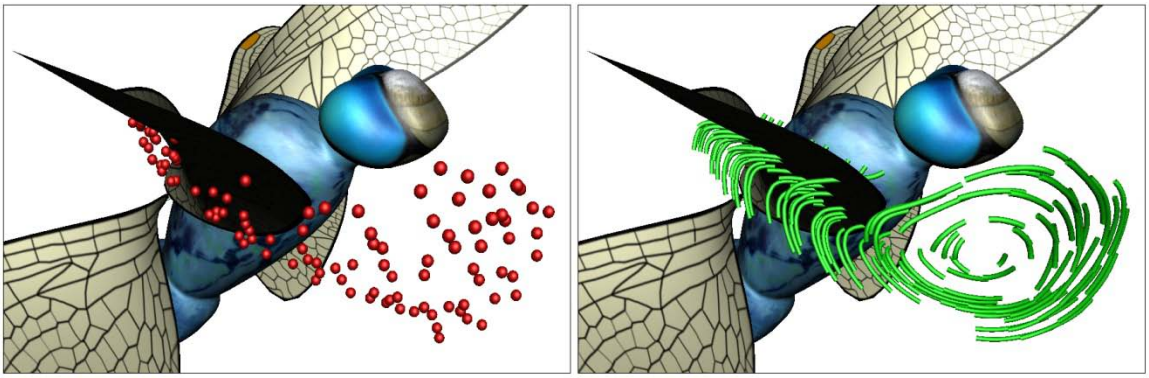

Figure 5.21: Comparison of sphere shaped particles and streamlets seeded at the same particle locations for vortex visualization.

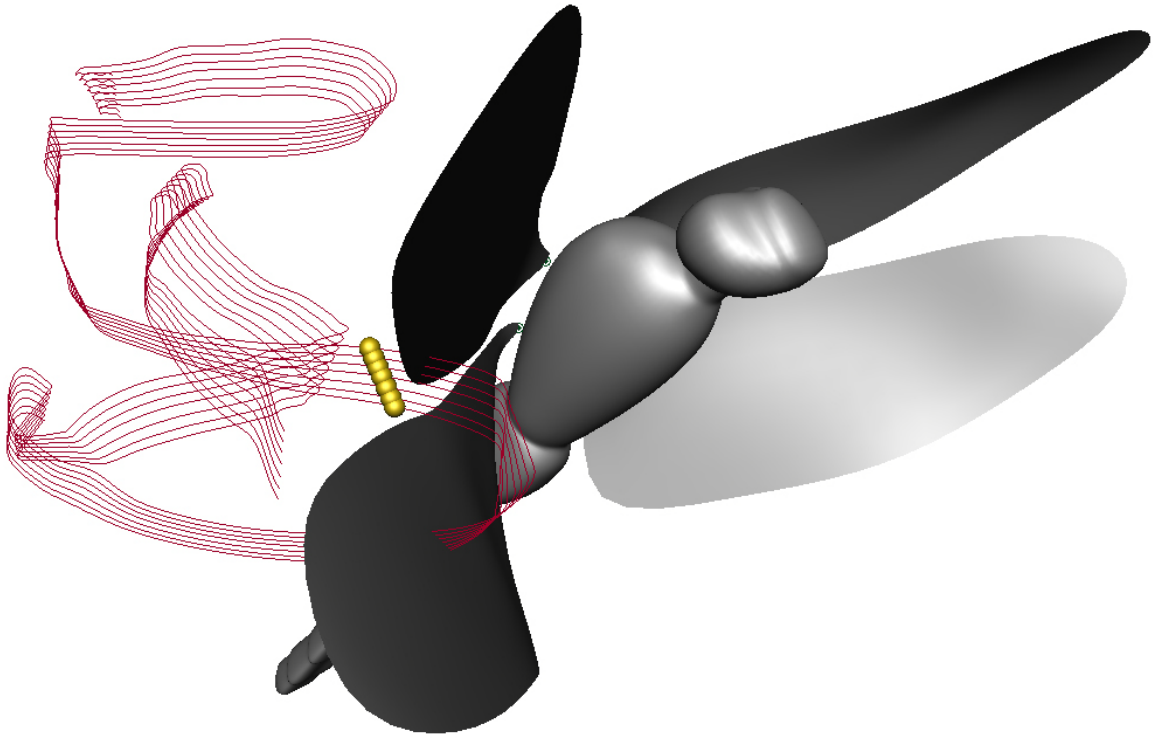

Figure 5.22: Seed curves corresponding to several vertices along the chord 35% from the wing tip.

A series of tests were done placing seeds along the chords various distances from the wing tips. Once a distance from the wing tip is chosen the user is limited to placing seeds on the plane parallel to the corresponding chord of the chosen wing. This seems like a large restriction given the size of the data, but all the desired flow features can be captured with this method. Figure 5.23 shows how chord seeds captured the leading edge vortex during both the up stroke and down stroke. Unlike the streamline images of the LEV, flow reattachment is also illustrated. The flow reattachment is key for flight because it means the vortex is stable and not shedding yet. Figure 5.24 shows how seeds at a chord near the root can be used to more effectively capture spanwise flow along the leading edge vortex core. This phenomena was only apparent in the first flap after takeoff, perhaps because more lift was needed at this time.

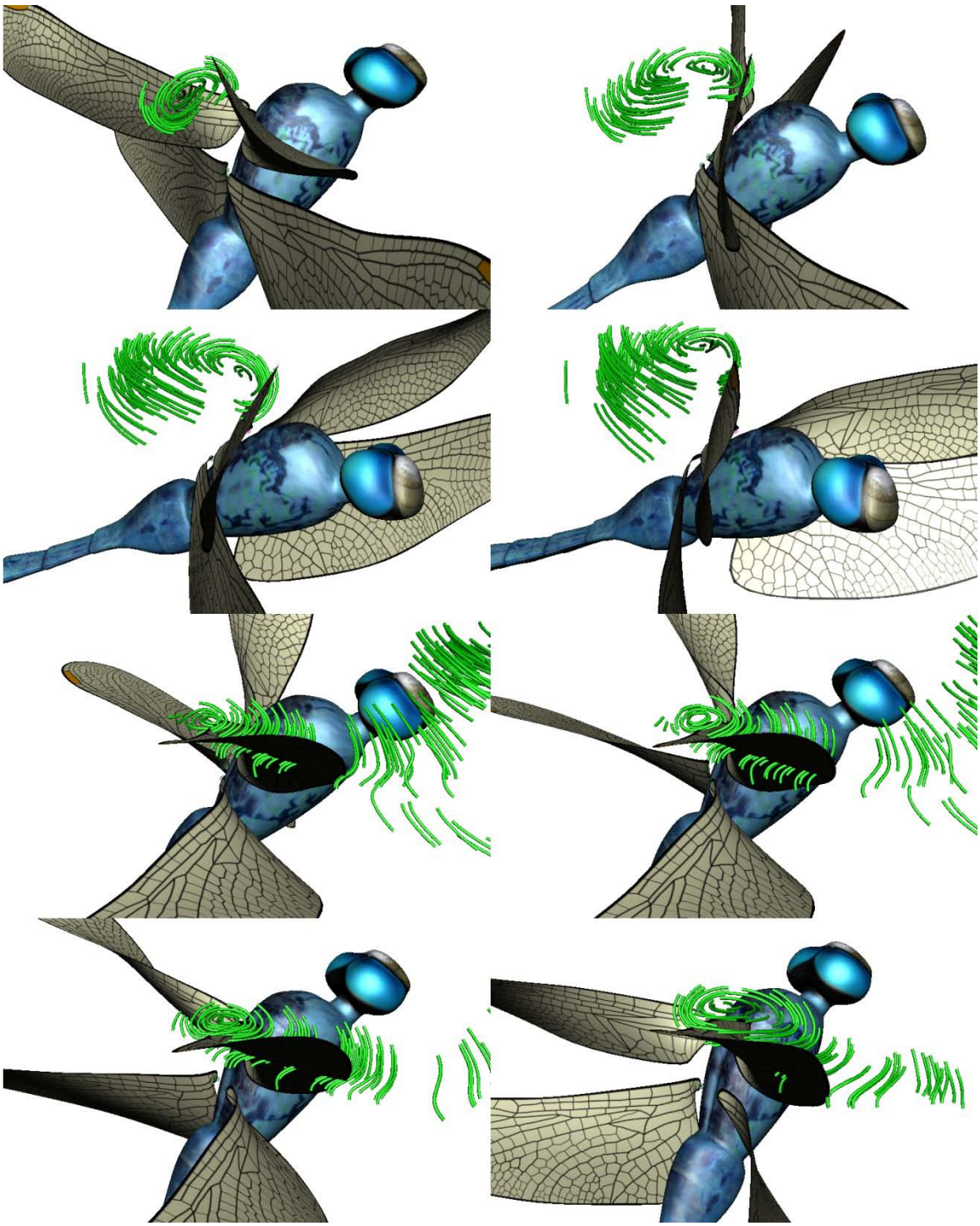

Figure 5.23: The leading edge vortex at eight time steps during both the up and down stroke.

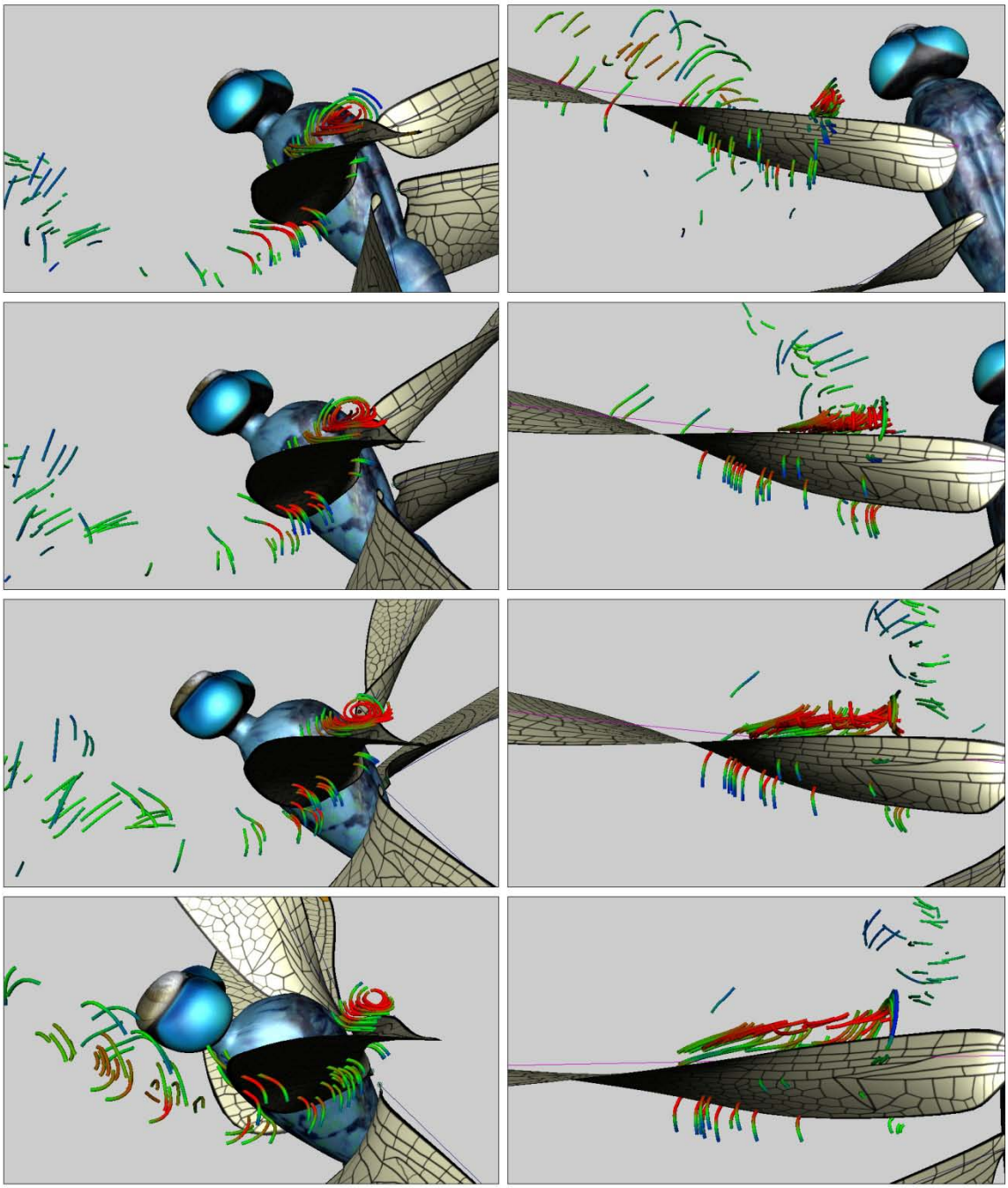

Figure 5.24: Spanwise flow captured by emitting flowing seeds along the wing chord near the root. The left column shows the LEV with a wing bound camera and the right column shows the same time steps from a camera perpendicular to the vortex.

5.3.6 Visualization of Vortex Shedding

As mentioned earlier in section 3.3.4, the combination of generalized streak seeds and carefully chosen base seed curves were useful for capturing the vortex shedding on the flapping disk as its angle of attack changed at the end of each stroke. Tests were performed with the chord seeds at stroke reversal to see if the flowing seeds could capture similar effects in the dragonfly data. When the vortex sheds over the top of the foil, it is known as dynamic stall, and it is believed to play a role in how insects generate the necessary lift to fly [96]. The up stroke to down stroke reversal showed the initial LEV dissipate and partially flow into the newly forming LEV (Figure 5.25).

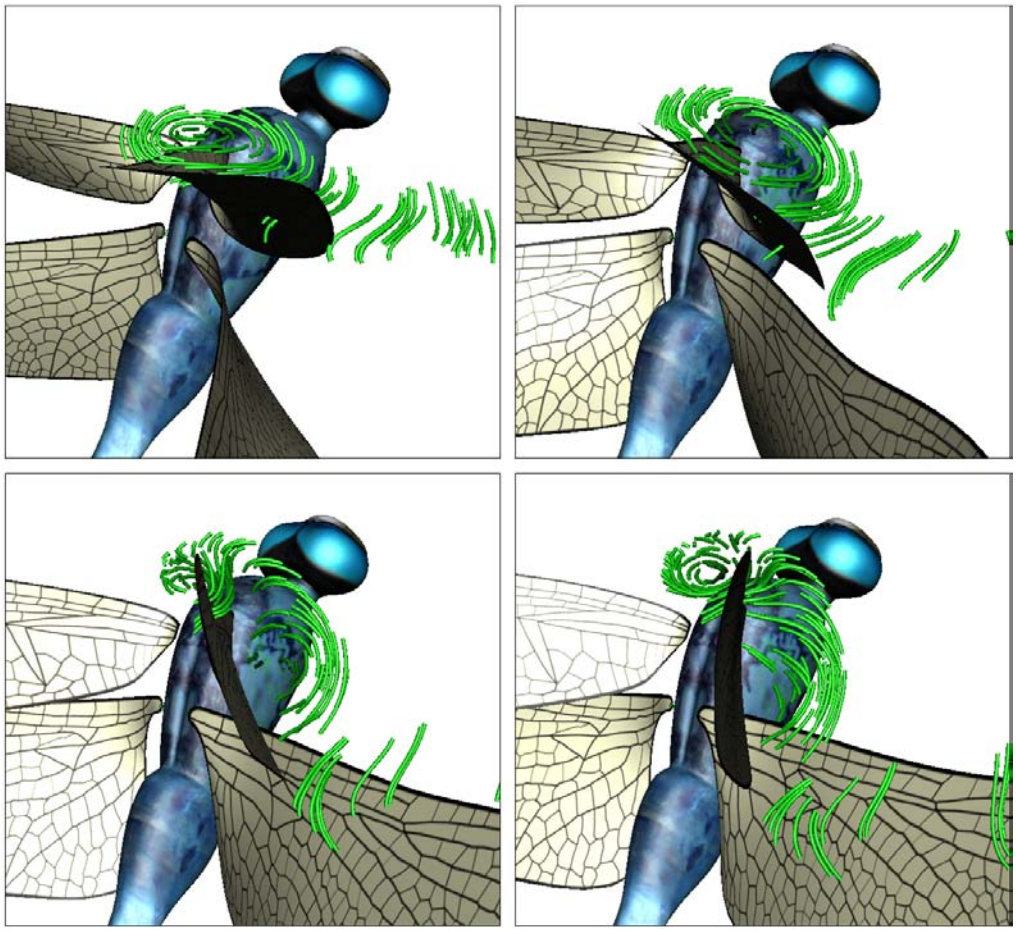

Figure 5.25: Vortex breakdown and reforming at the up stroke to down stroke reversal.

A similar test was done at the down stroke to up stroke reversal. In this case the leading edge vortex begins to grow in size and moves into the back of the wing as the angle of attack starts to change. As the vortex moves away from the wing its strength diminishes. Also, the core of the shed vortex is not a straight line. Both of these factors would make it very difficult to detect directly which is why flowing seed points are powerful for visualizing this aspect of flapping flight flow fields. Vortex shedding is important particularly in quad wing insect studies because it is possible to reclaim energy from vortices after they have been shed.

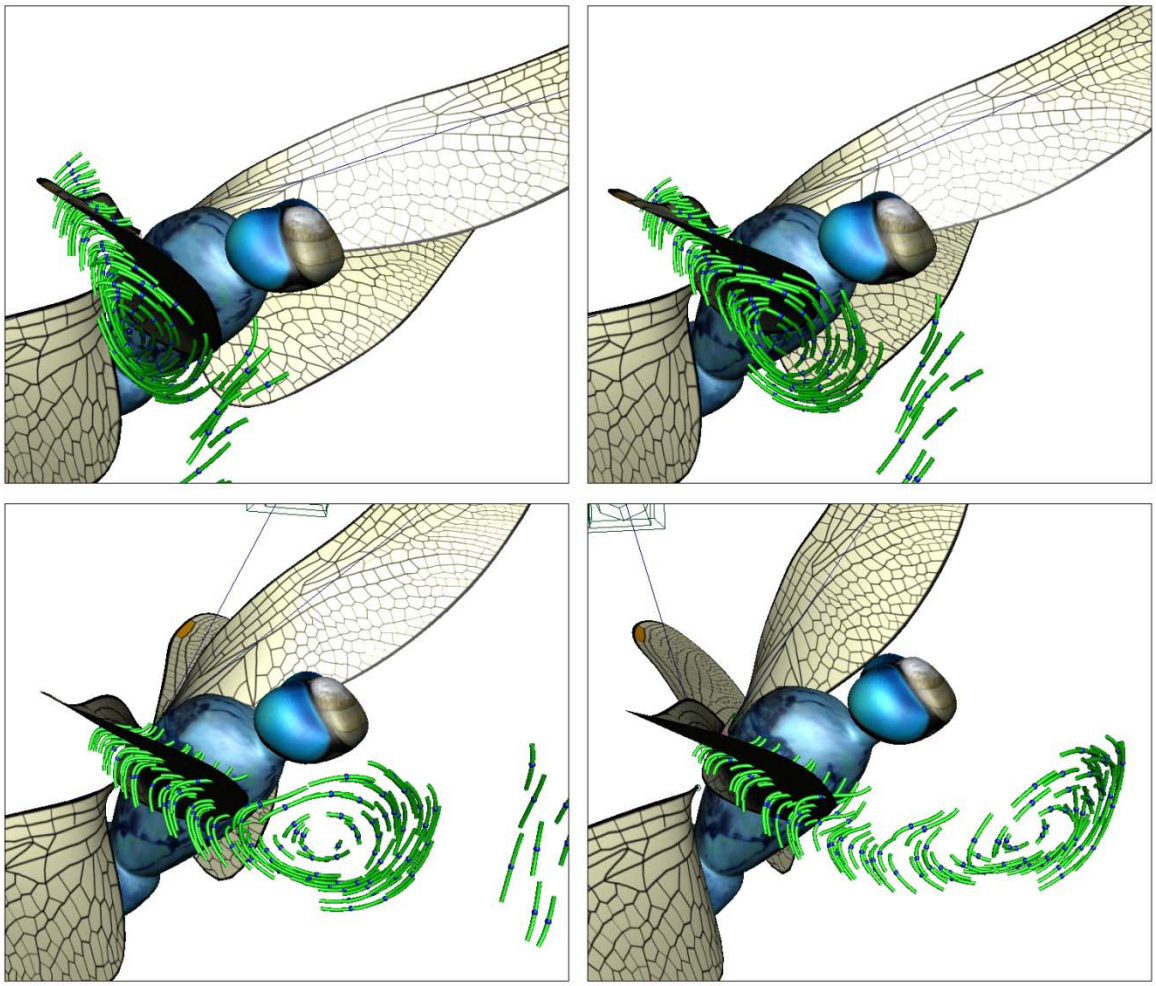

Figure 5.26: Vortex shedding at the end of the down stroke.

Clearly, the ability to capture the complex vortex behavior that occurs at stroke reversal is important for understanding how the dragonfly produces lift. Once the particles are in the right area, the streamlets integrated at each flowing seed point are what actually captures the vortex formation, shedding and breakdown. Particles alone would not capture this effect with this much clarity (Figure 3.14), nor would longer streamlines due to their spatial incoherence between frames. Direct visualization of this effect with vorticity magnitude would not be very effective because the vorticity around the shed vortices is much lower than that of the newly forming leading edge vortex. In 2D cases, texture based methods such as line integral convolution would probably capture this phenomena, however they are not as clear in 3D. Thus, visualization of how the leading edge vortex forms, sheds at stroke reversal and then reforms is an application where the flowing seed point algorithm is more useful than any traditional flow visualization method.

5.3.7 Visualization of Wake Capture

Visualization of wake capture is another phenomena which flowing seed points are more effective than other methods at capturing. It is similar to vortex shedding except that the vortices continue to interact with the wings after being shed. In a quad wing insect, there are two scenarios where wake capture can occur. In the first case the LEV from the previous stroke gets shed at reversal and partially flows into the new LEV forming at the beginning of the new stroke. Energy is captured from the original vortex which allows the insect to generate more lift (Figure 5.27). The other case is when a vortex is shed from the forewing, such that it interacts with the flow around the

corresponding hind wing. This situation is illustrated in Figure 5.28. While this phenomena is visually clear, its effects on lift production is not. In the case of wake capture between different wings on the same side of the insect, additional simulations are needed for individual wings in order to measure what effects the wake capture is really having.

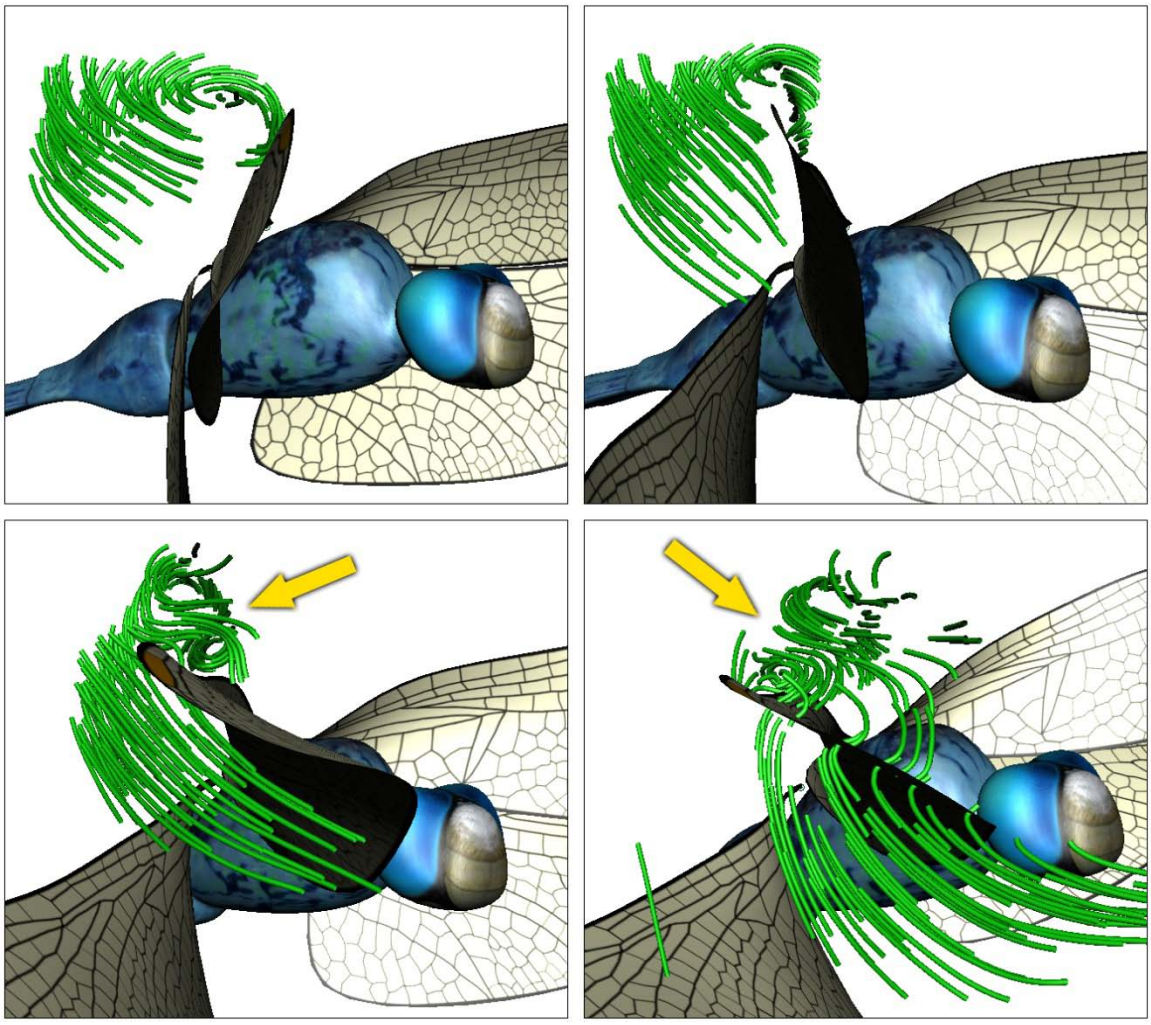

Figure 5.27: Wake capture from the previous half stroke on a single wing.

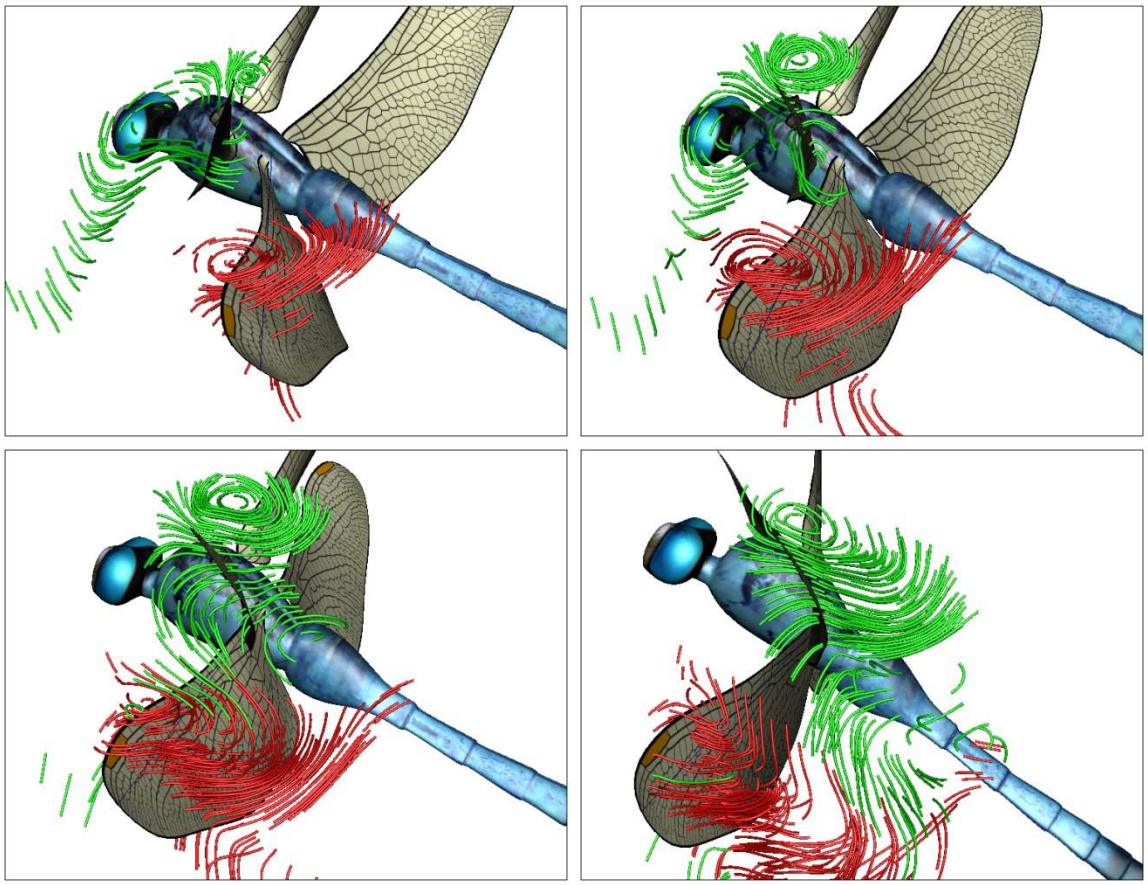

Figure 5.28: Visualization of a vortex shedding from the left forewing at stroke reversal and then partially attaching to the vortex on the left hind wing.

5.3.8 Vortex Behavior and Lift Production

Thus far I have demonstrated that the flowing seed point method can effectively capture vortex formation, movement, breakdown and recapture in the dragonfly data set, however this is only one side of the analysis. In order to prove what effects the vortices are having on the insect, the visualizations must be paired up with the studies of both the wing kinematics and lift production mentioned in Sections 4.2 and 4.3. In the first test, the left hind wing lift production was examined visually at several time steps near a peak in lift output. The results are shown in Figure 5.29.

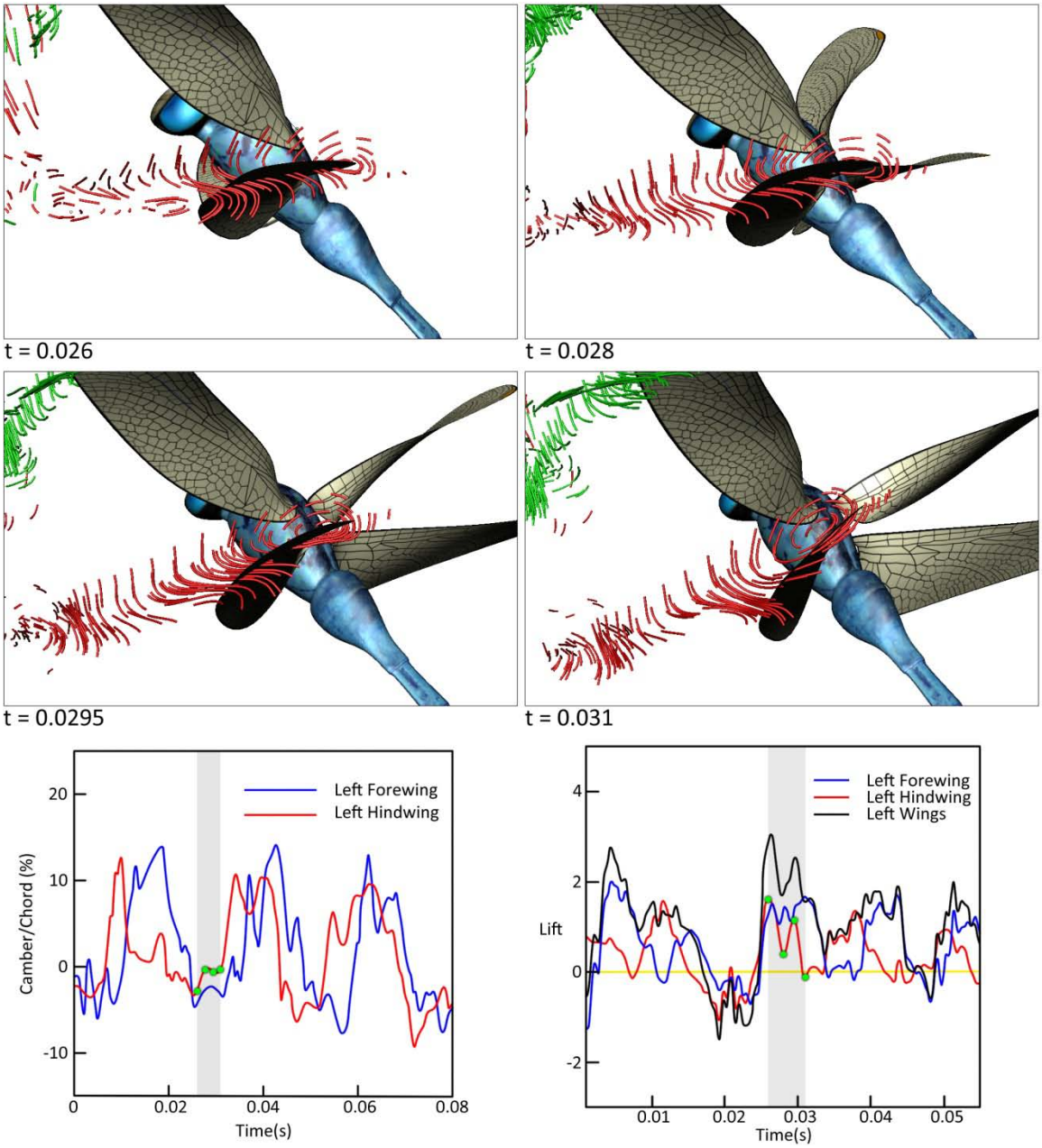

Figure 5.29: Comparison of flowing seed point visualizations of the left hind wing with the force history and camber to chord ratio over time at a high lift production interval.

As expected the peak lift production appears to occur when leading edge vortices are attached to the wings. This result is significant because most theories about quad wing flapping flight suggest that the hind wing's main purpose is to produce thrust, while

the forewings produce the majority of the lift. Clearly the hind wing is also producing lift during the takeoff maneuver, however additional simulations are needed to clarify whether this is always the case or whether it just happens during takeoff. In the next test, the flowing seeds around the left forewing were examined at a point of low lift production (Figure 5.30). Not surprisingly the corresponding flowing seed point visualizations show that the LEV is shedding at stroke reversal and that the new LEV has not started forming yet.

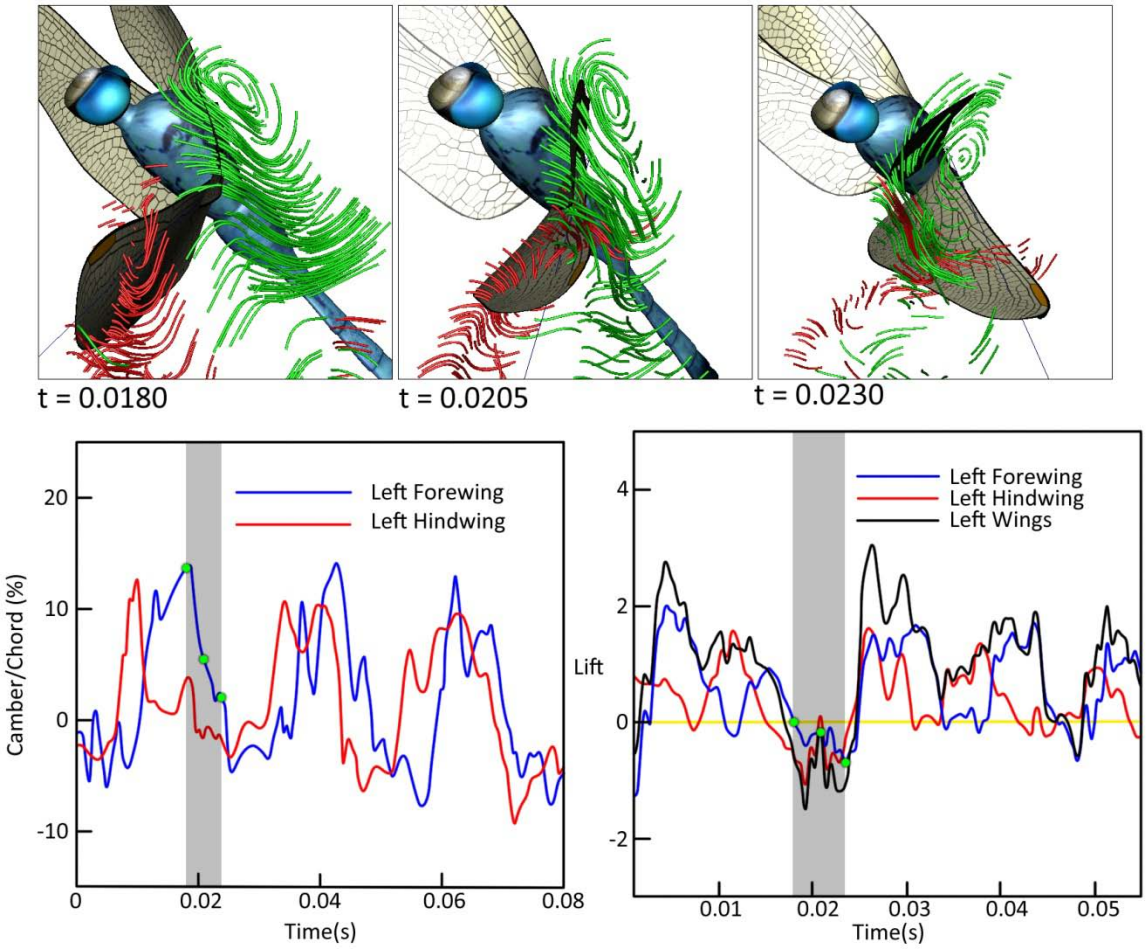

Figure 5.30: Comparison of flowing seed point visualizations of the left forewing with the force history and camber to chord ratio over time at a low lift production interval.

In both of these tests the leading edge vortex is most stable and the lift production is greatest when the camber to chord ratio is not changing much. However, it is not clear whether the wing camber is causing any of this or not. The camber to chord ratio changes rapidly at stroke reversal. Some vortex shedding obviously occurs at stroke reversal regardless of wing deformation however it is possible that the wing camber plays a role in how quickly a new LEV is formed. Until a comparative analysis is conducted with a rigid wing version of the same dragonfly it will not be clear how much of the observed vortex behavior is due to the stroke reversal and how much is due to the wing camber.

6 Conclusion

I have shown how streamlets and pathlets can be seeded at a series of points on a discretized generalized streak line or time line in order to generate one concise smoothly animated visualization. Also, I have demonstrated how dynamically deforming seed curves are much more powerful than traditional static seed objects when dealing with flows that contain multiple moving objects generating flow disturbances. Preliminary tests were done with a relatively simple flapping disk data set. In addition, an extremely complex deformable wing dragonfly takeoff and slow flight simulation was created in order to further evaluate the merits of these methods.

6.1 Contributions

The major contribution of the flowing seed point technique is that it combines the perceptual benefits of streamlines and path lines with those of generalized streak lines and particle advection into one smooth animation. Flowing seed points improve visualization of instantaneous and time varying velocity and divergence when compared to basic particles on the underlying seed curves. The flowing seed point method also proved to be very useful, compared to other visualization methods, for capturing the formation, evolution and breakdown of vortices which are only present briefly in a flow field. For example, it was able to capture the vortices that form through dynamic stall when the wings change their angle of attack.

The main contribution of dynamically deforming seed curves is the ability to achieve good particle coverage in the areas of the flow that are most important while

minimizing occlusion from less important areas. By exploiting 3D modeling algorithms such as edge loop selection, vertex normals, subdivision refinement and path extrusion, dynamic seed objects are very easy for a user to define with only a few mouse clicks. In addition, the vertex normal approach to generating dynamic seed curves in the vicinity of objects that are disturbing the flow has a significant speed advantage over any seeding method that is based on metrics taken from the underlying vector field. Dynamic seed curves also proved to be very useful for visualizing flow simulation data in cases where there is no inlet flow to place a static seed object in front of.

While its application domain is not computer science related, the reconstruction and simulation of the deformable wing dragonfly is a novel contribution in its own right. Most previous insect flight simulation studies were based on rigid wing models and do not accurately reflect the insect's true wing kinematics. Simulations of rigid wing models also do not accurately capture how a real insect produces the necessary lift to fly. The integrated analysis of this more accurate reconstruction and CFD simulation with the flowing seed point visualization method has helped provide previously unknown insight into the relationship between wing deformation, lift generation and vortex production.

6.2 Future Work

One possible extension to this work would be to combine it with some of the more sophisticated vortex detection algorithms mentioned in section 2.3 in order to improve coverage of all vortices in the flow domain. The vortex detection method I used for testing purposes zeroes in on only the leading edge vortex on each wing, however there are other vortices in the flow where flowing seeds could be injected. In particular

work is needed in detection and tracking of curved vortices in 3D unsteady flows. A 3D extension of the work in [81, 82] has potential in this area. Metrics that operate on larger portions of a series of densely seeded streamlines to measure the degree to which they are shaped like helices could potentially provide more robust vortex detection in very complex data sets.

Another possible future extension to this work is to try increasing the flowing seed point dimensionality to seed small stream surfaces. Stream surfaces tend to suffer from occlusion problems, however they are less prone to information overload than having a lot of individual streamlets. Recent developments using smoke based rendering with stream surfaces have lessened the occlusion problems so perhaps a series of short stream surfaces placed in critical areas of the flow would be better than a group of individual streamlets.

Finally it would also be useful to perform user studies of the perceptual benefits of the methods described in this document when compared to a series of other unsteady flow visualization methods. This is a very important area that is for the most part ignored in visualization research. The work by Ware suggests that the flowing seed point method for seeding streamlets at a series of time steps is perceptually more effective than even coverage of sphere shaped particles [112]. However, this study dealt only with 2D steady flow data. More work is needed to truly understand and establish criteria which measure the perceptual merit of any 3D unsteady flow visualization method.

References

[1] H. Hauser, R. S. Laramee, H. Doleisch, F. H. Post and B. Vrolijk, "The state of the art in flow visualization, part 1: Direct, texture-based, and geometric techniques," 2002.

[2] R. S. Laramee, H. Hauser, H. Doleisch, B. Vrolijk, F. H. Post and D. Weiskopf, "The State of the Art in Flow Visualization: Dense and Texture-Based Techniques," *Computer Graphics Forum,* vol. 23, pp. 203-221, 2004.

[3] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee and H. Doleisch, "The State of the Art in Flow Visualisation: Feature Extraction and Trackin," *Computer Graphics Forum,* vol. 22, pp. 775-792, 2003.

[4] T. McLoughlin, R. S. Laramee, R. Peikert, F. H. Post and M. Chen, "Over two decades of integration-based, geometric flow visualization," April. 2009.

[5] T. Salzbrunn, H. Jänicke, T. Wischgoll and G. Scheuermann, "The state of the art in flow visualization: Partition-based techniques," in *Simulation and Visualization 2008,* 2008, pp. 75-92.

[6] H. Dong, R. Mittal and F. M. Najjar, "Wake topology and hydrodynamic performance of low aspect-ratio flapping foils," *Journal of Fluid Mechanics,* vol. 566, pp. 309-343, November, 2006.

[7] R. Mittal, H. Dong, M. Bozkurttas, F. M. Najjar, A. Vargas and A. von Loebbecke, "A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries," *Journal of Computational Physics,* vol. 227, pp. 4825-4852, May, 2008.

[8] G. Turk and D. Banks, "Image-guided streamline placement," in *SIGGRAPH '96: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques,* 1996, pp. 453-460.

[9] B. Jobard and W. Lefer, "Creating evenly-spaced streamlines of arbitrary density," in *8th Eurographics Workshop on Visualization in Scientific Computing,* 1997, pp. 45-55.

[10] B. Jobard and W. Lefer, "The motion map: Efficient computation of steady flow animations," in *IEEE Visualization '97,* 1997, pp. 323-328.

[11] V. Verma, D. Kao and A. Pang, "A flow-guided streamline seeding strategy," in *Visualization ’00,* Salt Lake City, Utah, United States, 2000, pp. 163-170.

[12] A. Mebarki, P. Alliez and O. Devillers, "Farthest point seeding for efficient placement of streamlines," in *IEEE Visualization '05*, 2005, pp. 479-486.

[13] Z. Liu, R. Moorhead and J. Groner, "An Advanced Evenly-Spaced Streamline Placement Algorithm," *IEEE Transactions on Visualization and Computer Graphics,* vol. 12, pp. 965-972, September, 2006.

[14] L. Li, H. Hsieh and H. Shen, "Illustrative streamline placement and visualization," in *IEEE Pacific Visualization Symposium,* Kyoto, 2008, pp. 79-86.

[15] J. J. van Wijk, "Rendering surface-particles," in *IEEE Visualization '92,* Boston, Massachusetts, 1992, pp. 54-61.

[16] X. Mao, Y. Hatanaka, H. Higashida and A. Imamiya, "Image-guided streamline placement on curvilinear grid surfaces," in *IEEE Visualization '98,* Research Triangle Park, North Carolina, United States, 1998, pp. 135-142.

[17] B. Spencer, R. S. Laramee, C. Guoning and E. Zhang, "Evenly Spaced Streamlines for Surfaces: An Image-Based Approach," *Computer Graphics Forum,* vol. 28, pp. 1618-1631, June, 2009.

[18] O. Rosanwo, C. Petz, S. Prohaska, H. Hege and I. Hotz, "Dual streamline seeding," in *IEEE Pacific Visualization Symposium,* Beijing, China, 2009, pp. 9-16.

[19] D. N. Kenwright and G. D. Mallinson, "A 3-D streamline tracking algorithm using dual stream functions," in *IEEE Visualization '92,* Boston, Massachusetts, 1992, pp. 62-68.

[20] S. Ueng, C. Sikorski and K. Ma, "Efficient Streamline, Streamribbon, and Streamtube Constructions on Unstructured Grids," *IEEE Transactions on Visualization and Computer Graphics,* vol. 2, pp. 100-110, 1996.

[21] S. K. Lodha, A. Pang, R. E. Sheehan and C. M. Wittenbrink, "UFLOW: Visualizing uncertainty in fluid flow," in *VIS '96: Proceedings of the 7th Conference on Visualization '96,* San Francisco, California, United States, 1996, pp. 249-254.

[22] M. Schulz, F. Reck, W. Bartelheimer and T. Ertl, "Interactive visualization of fluid dynamics simulations in locally refined cartesian grids," in *VIS '99: Proceedings of the Conference on Visualization '99,* San Francisco, California, United States, 1999, pp. 413-416.

[23] V. Verma and A. Pang, "Comparative Flow Visualization," *IEEE Transactions on Visualization and Computer Graphics,* vol. 10, pp. 609-624, November, 2004.

[24] D. Pugmire, H. Childs, C. Garth, S. Ahern and G. H. Weber, "Scalable computation of streamlines on very large datasets," in *SC '09: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis,* Portland, Oregon, 2009, pp. 1-12.

[25] R. S. Laramee, D. Weiskopf, J. Schneider and H. Hauser, "Investigating swirl and tumble flow with a comparison of visualization techniques," in *IEEE Visualization '04,* 2004, pp. 51-58.

[26] Y. Xiangong, D. Kao and A. Pang, "Strategy for seeding 3D streamlines," in *IEEE Visualization '05,* 2005, pp. 471-478.

[27] L. Li and H. Shen, "Image Based Streamline Generation and Rendering," *IEEE Transactions on Visualization and Computer Graphics,* vol. 13, pp. 630-640, May-June, 2007.

[28] C. Yuan, J. D. Cohen and J. H. Krolik, "Similarity-Guided Streamline Placement with Error Evaluation," *IEEE Transactions on Visualization and Computer Graphics,* vol. 13, pp. 1448-1455, 2007.

[29] M. Zöckler, D. Stalling and H. Hege, "Interactive visualization of 3D-vector fields using illuminated stream lines," in *VIS '96: Proceedings of the 7th Conference on Visualization '96,* San Francisco, California, United States, 1996, pp. 107-113.

[30] O. Mallo, R. Peikert, C. Sigg and F. Sadlo, "Illuminated lines revisited," in *IEEE Visualization '05,* 2005, pp. 19-26.

[31] A. Fuhrmann and E. Gröller, "Real-time techniques for 3D flow visualization," in *VIS '98: Proceedings of the Conference on Visualization '98,* Research Triangle Park, North Carolina, United States, 1998, pp. 305-312.

[32] O. Mattausch, T. Theußl, H. Hauser and E. Gröller, "Strategies for interactive exploration of 3D flow using evenly-spaced illuminated streamlines," in *SCCG '03: Proceedings of the 19th Spring Conference on Computer Graphics,* Budmerice, Slovakia, 2003, pp. 213-222.

[33] R. S. Laramee and H. Hauser, "Geometric flow visualization techniques for CFD simulation data," in *SCCG '05: Proceedings of the 21st Spring Conference on Computer Graphics,* Budmerice, Slovakia, 2005, pp. 221-224.

[34] C. Weigle and D. Banks, "A Comparison of the Perceptual Benefits of Linear Perspective and Physically-Based Illumination for Display of Dense 3D Streamtubes," *IEEE Transactions on Visualization and Computer Graphics,* vol. 14, pp. 1723-1730, November-December, 2008.

[35] M. Tarini, P. Cignoni and C. Montani, "Ambient Occlusion and Edge Cueing for Enhancing Real Time Molecular Visualization," *IEEE Transactions on Visualization and Computer Graphics,* vol. 12, pp. 1237-1244, September-October, 2006.

[36] G. Schussman and K. Ma, "Anisotropic volume rendering for extremely dense, thin line data," in *IEEE Visualization '04,* 2004, pp. 107-114.

[37] T. Salzbrunn and G. Scheuermann, "Streamline Predicates," *IEEE Transactions on Visualization and Computer Graphics,* vol. 12, pp. 1601-1612, 2006.

[38] T. Salzbrunn, C. Garth, G. Scheuermann and J. Meyer, "Pathline Predicates and Unsteady Flow Structures," *The Visual Computer,* vol. 24, pp. 1039-1051, 2008.

[39] J. P. M. Hultquist, "Constructing stream surfaces in steady 3D vector fields," in *VIS '92: Proceedings of the 3rd Conference on Visualization '92,* Boston, Massachusetts, 1992, pp. 171-178.

[40] J. J. van Wijk, "Implicit stream surfaces," in *VIS '93: Proceedings of the 4th Conference on Visualization '93,* San Jose, California, 1993, pp. 245-252.

[41] H. Löffelmann, L. Mroz, E. Gröller and W. Purgathofer, "Stream arrows: enhancing the use of stream surfaces for the visualization of dynamical systems," *The Visual Computer,* vol. 13, pp. 359-369, 1997.

[42] G. Scheuermann, T. Bobach, H. Hagen, K. Mahrous, B. Hamann, K. I. Joy and W. Kollmann, "A tetrahedra-based stream surface algorithm," in *VIS '01: Proceedings of the Conference on Visualization '01,* San Diego, California, 2001, pp. 151-158.

[43] C. Garth, X. Tricoche, T. Salzbrunn, T. Bobach and G. Scheuermann, "Surface techniques for vortex visualization," in *Data Visualization, Proceedings of the 6th Joint IEEE TCVG–EUROGRAPHICS Symposium on Visualization,* 2004, pp. 155-164.

[44] R. S. Laramee, C. Garth, J. Schneider and H. Hauser, "Texture advection on stream surfaces: A novel hybrid visualization applied to CFD simulation results," in *Eurographics / IEEE VGTC Symposium on Visualization,* 2006, pp. 155-162.

[45] R. Peikert and F. Sadlo, "Topologically relevant stream surfaces for flow visualization," in *Spring Conference on Computer Graphics,* 2009, pp. 43-50.

[46] W. J. Schroeder, C. R. Volpe and W. E. Lorensen, "The stream polygon: A technique for 3D vector field visualization," in *VIS '91: Proceedings of the 2nd Conference on Visualization '91,* San Diego, California, 1991, pp. 126-132.

[47] N. Max, B. Becker and R. Crawfis, "Flow volumes for interactive vector field visualization," in *VIS '93: Proceedings of the 4th Conference on Visualization '93,* San Jose, California, 1993, pp. 19-24.

[48] B. G. Becker, N. L. Max and D. A. Lane, "Unsteady flow volumes," in *VIS '95: Proceedings of the 6th Conference on Visualization '95,* 1995, pp. 329-335.

[49] D. Xue, C. Zhang and R. Crawfis, "Rendering implicit flow volumes," in *VIS '04: Proceedings of the Conference on Visualization '04,* 2004, pp. 99-106.

[50] B. Jobard and W. Lefer, "Unsteady Flow Visualization by Animating Evenly-Spaced Streamlines," *Computer Graphics Forum,* vol. 19, pp. 31-39, 2000.

[51] W. Lefer, B. Jobard and C. Leduc, "High-Quality Animation of 2D Steady Vector Fields," *IEEE Transactions on Visualization and Computer Graphics,* vol. 10, pp. 2-14, 2004.

[52] A. Wiebel, X. Tricoche, D. Schneider, H. Janicke and G. Scheuermann, "Generalized Streak Lines: Analysis and Visualization of Boundary Induced Vortices," *IEEE Transactions on Visualization and Computer Graphics,* vol. 13, pp. 1735-1742, 2007.

[53] D. A. Lane, "Visualization of time-dependent flow fields," in *VIS '93: Proceedings of the 4th Conference on Visualization '93,* San Jose, California, 1993, pp. 32-38.

[54] D. A. Lane, "UFAT - A particle tracer for time-dependent flow fields," in *VIS '94: Proceedings of the Conference on Visualization '94,* Washinton, D.C., 1994, pp. 257-264.

[55] D. N. Kenwright and D. A. Lane, "Interactive Time-Dependent Particle Tracing Using Tetrahedral Decomposition," *IEEE Transactions on Visualization and Computer Graphics,* vol. 2, pp. 120-129, 1996.

[56] C. Teitzel, R. Grosso and T. Ertl, "Efficient and reliable integration methods for particle tracing in unsteady flows on discrete meshes," in *Visualization in Scientific Computing '97,* 1997, pp. 31-41.

[57] C. Teitzel, R. Grosso and T. Ertl, "Particle tracing on sparse grids," in *Visualization in Scientific Computing '98,* 1998, pp. 81-90.

[58] C. Teitzel and T. Ertl, "New approaches for particle tracing on sparse grids," in *Data Visualization '99, Eurographics,* 1998, pp. 73-84.

[59] M. Schirski, A. Gerndt, T. van Reimersdahl, T. Kuhlen, P. Adomeit, O. Lang, S. Pischinger and C. Bischof, "ViSTA FlowLib –– A framework for interactive visualization and exploration of unsteady flows in virtual environments," in *EGVE '03: Proceedings of the Workshop on Virtual Environments 2003,* Zurich, Switzerland, 2003, pp. 77-85.

[60] S. Bryson and C. Levit, "The virtual windtunnel: An environment for the exploration of three-dimensional unsteady flows," in *VIS '91: Proceedings of the 2nd Conference on Visualization '91,* San Diego, California, 1991, pp. 17-24.

[61] A. Wiebel and G. Scheuermann, "Eyelet particle tracing - steady visualization of unsteady flow," in *IEEE Visualization '05*, 2005, pp. 607-614.

[62] A. Helgeland and T. Elboth, "High-Quality and Interactive Animations of 3D Time-Varying Vector Fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, pp. 1535-1546, 2006.

[63] H. Jänicke and G. Scheuermann, "Steady visualization of the dynamics in fluids using e-machines," *Computers & Graphics*, vol. 33, pp. 597-606, October, 2009.

[64] T. Schafhitzel, E. Tejada, D. Weiskopf and T. Ertl, "Point-based stream surfaces and path surfaces," in *GI '07: Proceedings of Graphics Interface 2007*, Montreal, Canada, 2007, pp. 289-296.

[65] C. Garth, H. Krishnan, X. Tricoche, T. Bobach and K. I. Joy, "Generation of Accurate Integral Surfaces in Time-Dependent Vector Fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, pp. 1404-1411, 2008.

[66] W. von Funck, T. Weinkauf, H. Theisel and H. Seidel, "Smoke Surfaces: An Interactive Flow Visualization Technique Inspired by Real-World Flow Experiments," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, pp. 1396-1403, November-December, 2008.

[67] T. McLoughlin, R. S. Laramee and E. Zhang, "Easy integral surfaces: A fast, quad-based stream and path surface algorithm," in *CGI '09: Computer Graphics International*, Victoria, British Columbia, Canada, 2009, pp. 73-82.

[68] H. Krishnan, C. Garth and K. I. Joy, "Time and Streak Surfaces for Flow Visualization in Large Time-Varying Data Sets," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, pp. 1267-1274, November-December, 2009.

[69] K. Buerger, F. Ferstl, H. Theisel and R. Westermann, "Interactive Streak Surface Visualization on the GPU," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, pp. 1259-1266, November-December, 2009.

[70] N. J. Zabusky, O. N. Boratav, R. B. Pelz, M. Gao, D. Silver and S. P. Cooper, "Emergence of coherent patterns of vortex stretching during reconnection: A scattering paradigm," *Physical Review Letters*, vol. 67, pp. 2469-2472, October, 1991.

[71] J. Villasenor and A. Vincent, "An algorithm for space recognition and time tracking of vorticity tubes in turbulence," *CVGIP: Image Understanding*, vol. 55, pp. 27-35, January, 1992.

[72] Y. Levy, D. Degani and A. Seginer, "Graphical Visualization of Vortical Flows by Means of Helicity," *AIAA JOURNAL*, vol. 28, pp. 1347-1352, August, 1990.

[73] S. K. Robinson, "Coherent Motions in the Turbulent Boundary Layer," *Annual Review of Fluid Mechanics,* vol. 23, pp. 601-639, January, 1991.

[74] D. C. Banks and B. A. Singer, "A Predictor-Corrector Technique for Visualizing Unsteady Flow," *IEEE Transactions on Visualization and Computer Graphics,* vol. 1, pp. 151-163, June, 1995.

[75] M. Roth and R. Peikert, "Flow visualization for turbomachinery design," in *VIS '96: Proceedings of the 7th Conference on Visualization '96,* San Francisco, California, United States, 1996, pp. 381-384.

[76] D. Sujudi and R. Haimes, "Identification Of Swirling Flow In 3-D Vector Fields," *AIAA Paper,* June, 1995.

[77] D. N. Kenwright and R. Haimes, "Automatic Vortex Core Detection," *IEEE Computer Graphics and Applications,* vol. 18, pp. 70-74, July-August, 1998.

[78] M. Roth and R. Peikert, "A higher-order method for finding vortex core lines," in *VIS '98: Proceedings of the Conference on Visualization '98,* Research Triangle Park, North Carolina, United States, 1998, pp. 143-150.

[79] M. Jiang, R. Machiraju and D. Thompson, "A novel approach to vortex core region detection," in *VISSYM '02: Proceedings of the Symposium on Data Visualisation 2002,* Barcelona, Spain, 2002, pp. 217.

[80] M. Jankun-Kelly, J. Ming, D. Thompson and R. Machiraju, "Vortex Visualization for Practical Engineering Applications," *IEEE Transactions on Visualization and Computer Graphics,* vol. 12, pp. 957-964, September-October, 2006.

[81] I. A. Sadarjoen and F. H. Post, "Geometric methods for vortex extraction," in *Joint Eurographics-IEEE TVCG Symposium on Visualization,* pp. 53-62.

[82] I. A. Sadarjoen and F. H. Post, "Detection, quantification, and tracking of vortices using streamline geometry," *Computers & Graphics,* vol. 24, pp. 333-341, 2000.

[83] R. S. Laramee, C. Garth, H. Doleisch, J. Schneider, H. Hauser and H. Hagen, "Visual analysis and exploration of fluid flow in a cooling jacket," in *IEEE Visualization '05,* 2005, pp. 623-630.

[84] D. Bauer, R. Peikert, M. Sato and M. Sick, "A case study in selective visualization of unsteady 3D flow," in *VIS '02: Proceedings of the Conference on Visualization '02,* Boston, Massachusetts, 2002, pp. 525-528.

[85] Q. You, S. Fang and L. Zhu, "Visualizing vortex shedding of an elastic plate interacting with a 3D viscous flow," in *CIT '09: Proceedings of the 2009 Ninth IEEE International Conference on Computer and Information Technology,* 2009, pp. 312-317.

[86] B. Soni, D. Thompson and R. Machiraju, "Visualizing Particle/Flow Structure Interactions in the Small Bronchial Tubes," *IEEE Transactions on Visualization and Computer Graphics,* vol. 14, pp. 1412-1427, November-December, 2008.

[87] X. He, G. Duckwiler and D. J. Valentino, "Lattice Boltzmann simulation of cerebral artery hemodynamics," *Computers & Fluids,* vol. 38, pp. 789-796, April, 2009.

[88] R. Wootton, "Aerodynamics: How flies fly," *Nature,* vol. 400, pp. 112-113, July, 1999.

[89] T. Weis-Fogh, "Quick Estimates of Flight Fitness in Hovering Animals, Including Novel Mechanisms for Lift Production," *Journal of Experimental Biology,* vol. 59, pp. 169-230, August, 1973.

[90] C. P. Ellington, "The Aerodynamics of Hovering Insect Flight. I. The Quasi-Steady Analysis," *Philosophical Transactions of the Royal Society of London.Series B, Biological Sciences,* vol. 305, pp. 1-15, February, 1984.

[91] E. C. Polhamus, "Predictions of Vortex-Lift Characteristics by a Leading-Edge Suction Analogy," *Journal of Aircraft,* vol. 8, pp. 193-199, April, 1971.

[92] J. H. Marden, "Maximum Lift Production During Takeoff in Flying Animals," *Journal of Experimental Biology,* vol. 130, pp. 235-258, July, 1987.

[93] A. R. Ennos, "The effect of size on the optimal shapes of gliding insects and seeds," *Journal of Zoology,* vol. 219, pp. 61-69, September, 1989.

[94] J. M. Zanker and K. G. Gotz, "The Wing Beat of Drosophila Melanogaster II. Dynamics," *Philosophical Transactions of the Royal Society of London.Series B, Biological Sciences,* vol. 327, pp. 19-44, February, 1990.

[95] R. Dudley, "Biomechanics of Flight in Neotropical Butterflies: Aerodynamics and Mechanical Power Requirements," *Journal of Experimental Biology,* vol. 159, pp. 335-357, September, 1991.

[96] A. P. Willmott, C. P. Ellington and A. L. R. Thomas, "Flow visualization and unsteady aerodynamics in the Flight of the hawkmoth, Manduca sexta," *Philosophical Transactions: Biological Sciences,* vol. 352, pp. 303-316, March, 1997.

[97] T. Maxworthy, "Experiments on the Weis-Fogh mechanism of lift generation by insects in hovering flight. Part 1. Dynamics of the ‘fling’," *Journal of Fluid Mechanics,* vol. 93, pp. 47-63, 1979.

[98] M. H. Dickinson and K. G. Gotz, "Unsteady Aerodynamic Performance of Model Wings at Low Reynolds Numbers," *Journal of Experimental Biology,* vol. 174, pp. 45-64, January, 1993.

[99] S. P. Sane, "The aerodynamics of insect flight," *Journal of Experimental Biology,* vol. 206, pp. 4191-4208, December, 2003.

[100] C. P. Ellington, d. B. van, A. P. Willmott and A. L. R. Thomas, "Leading-edge vortices in insect flight," *Nature,* vol. 384, pp. 626-630, December, 1996.

[101] C. van den Berg and C. P. Ellington, "The Three-Dimensional Leading-Edge Vortex of a Hovering Model Hawkmoth," *Philosophical Transactions: Biological Sciences,* vol. 352, pp. 329-340, March, 1997.

[102] J. M. Birch and M. H. Dickinson, "Spanwise flow and the attachment of the leading-edge vortex on insect wings," *Nature,* vol. 412, pp. 729-733, August, 2001.

[103] M. H. Dickinson, "The Effects of Wing Rotation on Unsteady Aerodynamic Performance at Low Reynolds Numbers," *Journal of Experimental Biology,* vol. 192, pp. 179-206, July, 1994.

[104] M. H. Dickinson, F. Lehmann and S. P. Sane, "Wing Rotation and the Aerodynamic Basis of Insect Flight," *Science,* vol. 284, pp. 1954-1960, June, 1999.

[105] J. M. Birch, W. B. Dickson and M. H. Dickinson, "Force production and flow structure of the leading edge vortex on flapping wings at high and low Reynolds numbers," *Journal of Experimental Biology,* vol. 207, pp. 1063-1072, March, 2004.

[106] A. L. R. Thomas, G. K. Taylor, R. B. Srygley, R. L. Nudds and R. J. Bomphrey, "Dragonfly flight: free-flight and tethered flow visualizations reveal a diverse array of unsteady lift-generating mechanisms, controlled primarily via angle of attack," *Journal of Experimental Biology,* vol. 207, pp. 4299-4323, November, 2004.

[107] R. Dudley, "BIOMECHANICS: Enhanced: Unsteady Aerodynamics," *Science,* vol. 284, pp. 1937-1939, June, 1999.

[108] R. Wootton, "Aerodynamics: From insects to microvehicles," *Nature,* vol. 403, pp. 144-145, January, 2000.

[109] G. V. Lauder, "Aerodynamics: Flight of the robofly," *Nature,* vol. 412, pp. 688-689, August, 2001.

[110] Z. J. Wang, "Dissecting Insect Flight," *Annual Review of Fluid Mechanics,* vol. 37, pp. 183-210, January, 2005.

[111] J. Stam, "Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values," in *SIGGRAPH '98: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques,* 1998, pp. 395-404.

[112] C. Ware, "Toward a Perceptual Theory of Flow Visualization," *IEEE Computer Graphics and Applications*, vol. 28, pp. 6-11, 2008.