Automated Visual Tracking for Behavioral Analysis of Biological Model Organisms

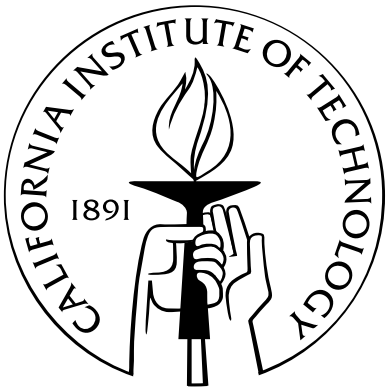
Thesis by

Ebraheem Ihsan Fontaine

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy



California Institute of Technology

Pasadena, California

2008

(Defended April 16, 2008)

Acknowledgements

وَلَوْ أَنَّمَا فِي الْأَرْضِ مِن شَجَرَةٍ أَقْلَامٌ وَالْبَحْرُ يَمُدُّهُ مِن بَعْدِهِ سَبْعَةُ أَبْحُرٍ مَّا نَفِدَتْ كَلِمَاتُ اللَّهِ إِنَّ اللَّهَ عَزِيزٌ حَكِيمٌ

If all the trees on the Earth were pens, and the sea replenished with seven more seas were ink, the words of Allah would not be exhausted. Indeed Allah is all-mighty, all-wise. (Luqman 27)

Completing my PhD studies over the last 5+ years has truly been a rewarding and humbling experience. First and foremost, I extend my sincere gratitude to God (Glory and Greatness be to Him) for bestowing upon me a life full of bounties. You alone do I worship, and you alone do I seek for help (al-Fatihah 5). Pursuing my education at the doctorate level has given me insight into the vast nature of scientific knowledge. It is with this insight that I quote the above verse from the Quran, which expounds upon the Divine knowledge. This theme in also captured in the common phrase, "The more you know, the more you realize what you don't know." Thus, it has been rewarding to contribute to scientific knowledge with a doctoral dissertation, and even more rewarding and humbling to recognize that this document is a small subset of all scientific knowledge, which is a small subset of the Divine knowledge.

Many people have provided me with invaluable support over my tenure as a graduate student. I sincerely thank my advisor Joel for providing financial stability, guidance, and good conversation. Despite my technical and analytic shortcomings, Joel used his expertise to coach me along the way and make sure I didn't pursue a path that was certainly doomed to fail. In addition, Al Barr functioned as an unofficial co-advisor for me and dedicated significant time, especially in the

early stages, to helping formulate this work and providing useful career advice. Joel and Al were able to support this work due to support of the Beckman Institute at Caltech, who recognized the importance of this work and awarded a multi-year pilot grant. I also thank the Student Life and Housing Office staff for giving me the opportunity to work as a Residential Associate and providing the flexibility to balance my research projects with the residential duties.

Finally, I wish to acknowledge my family who have also provided tremendous amounts of support. In particular, I thank my mother, who home schooled me for the first 10 years of my life, which provided the foundation to catapult me into future academic success. My three sisters, Muneera, Hajure, and Nargis, continue to provide that direct and open advice that can only come from a sibling. I am also indebted my father who taught me numerous lessons about manhood through his actions and words. Last, but not least, I thank my beautiful wife Danielle for the patience and love she has provided over the years. We knew our life would be an adventure when we embarked off to California in 2002, and I am eager to open the next chapter of our lives together as our family and love continue to grow.

Abstract

Capturing the detailed motion and behavior of biological organisms plays an important role in a wide variety of research disciplines. Many studies in biomechanics, neuroethology, and developmental biology rely on analysis of video sequences to understand the underlying behavior. However, the efficient and rapid quantification of these complex behavioral traits imposes a major bottleneck on the elucidation of many interesting scientific questions. The goal of this thesis is to develop a suite of model-based visual tracking algorithms that will apply across a variety of model organisms used in biology. These automated tracking algorithms operate in a high-throughput, high-resolution manner needed for a productive synthesis with modern genetic approaches. To this end, I demonstrate automated estimation of the detailed body posture of nematodes, zebrafish, and fruit flies from calibrated video.

The current algorithm utilizes a generative geometric model to capture the organism's shape and appearance. To accurately predict the organism's motion between video frames, I incorporate a motion model that matches tracked motion patterns to patterns in a training set. This technique is invariant with respect to the organism's velocity and can easily incorporate training data from completely different motion patterns. The prediction of the motion model is refined using measurements from the image. In addition to high-contrast feature points, I introduce a region, segmentation model based on level sets that are formally integrated into the observation framework of an Iterated Kalman Filter (IKF). The prior knowledge provided by the geometric and motion models improves tracking accuracy in the presence of partial occlusions and misleading visual cues.

The method is used to track the position and shape of multiple nematodes during mating behavior, zebrafish of different ages during escape response, and fruit flies during take off maneuvers.

These applications demonstrate the modular design of this model-based visual tracking system, where the user can specify which components are appropriate to a given experiment. In contrast to other approaches, which are customized to a particular organism or experimental setup, my approach provides a foundation that requires little re-engineering whenever the experimental parameters are changed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the last few decades, many technological advances have permitted the automated quantification of an organism's genotype[1]. For instance, sequencing and assaying robots were a great benefit to the human genome project. However, techniques to automatically quantify the phenotype[2] remain in their infancy. One reason for the lag in techniques to quantify phenotype is the significant technical challenge involved [4]. The phenotype encompasses a wide array of characteristics from static morphological features to dynamic behavior. To address these complicated challenges, the new field of *Phenomics* [44] was born to (1) formally define the phenotypic features important to a particular characterization and (2) develop new computational techniques to measure, model, analyze, and/or classify the phenotypic features. Success in each area is coupled to progress in the other. For instance, new automated techniques are needed to measure large numbers of features in an organism, and subsequent analysis will elucidate those features important to the phenotype characterization. Similarly, defining the phenotypic features properly will guide the development of novel measurement techniques by focusing information extraction on particular traits. Recognizing that quantitative, automated phenotype analysis presents the major bottleneck in current biological science, several projects have emerged devoted specifically to humans and mice [10, 38].

Success in the area of Phenomics will have huge impacts in all areas of biological science and medicine. Let us examine a particular success to appreciate the development of such techniques. The small nematode Caenorhabditis elegans (*C. elegans*) is a widely used model organism in the

[1] The genetic makeup of an organism

[2] An organism's total physical appearance and constitution. It is produced by the interaction of the genotype and the environment.

study of genetics and developmental biology. Automated visual tracking of the nematode motion in a petri dish permitted researchers to efficiently screen large numbers of nematodes to measure changes in locomotion due to genetic mutations and/or changing experimental conditions. The resulting automated phenotyping of these organisms from the collected measurements has facilitated the discovery of neural pathways involved in nicotine response and alcohol intoxication [35, 26]. Besides increasing general scientific knowledge, discovery of these pathways in model organisms can lead to drug discovery for ongoing human ailments [58]. Although some success stories can be identified, significant work is needed to bring these technologies to the same level of sophistication that is currently provided by modern genetic techniques that are available to cell and molecular biologists [1, 44].

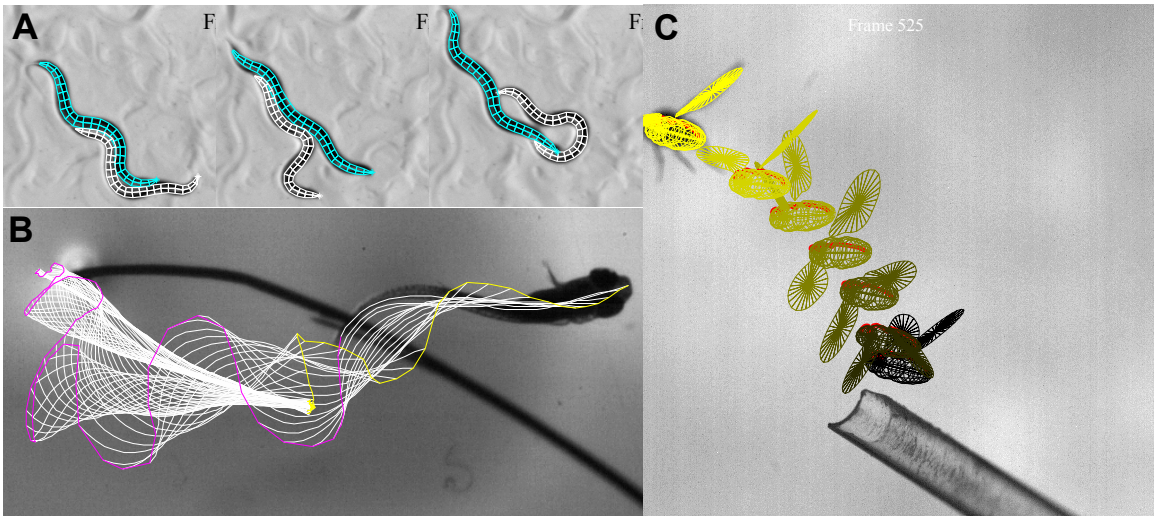


Figure 1.1: Automated tracking results for genetic model organisms. The estimated location of the model is overlaid on the original camera image in order to show the fidelity of the tracking. (A) Nematodes during mating behavior, (B) zebrafish executing escape response, and (C) fruit flies during flight initiation are demonstrated in this thesis. (A) and (B) are planar tracking from a single camera, while the 3D motion in (C) is shown from one of three calibrated camera views.

The goal of this thesis is to contribute to the general area of Phenomics by developing automated visual tracking methods for analyzing the behavior of biological model organisms. To this end, I focus on detailed body posture estimation of nematodes, zebrafish, and fruit flies from calibrated video (Figure 1.1). These three genetic model organisms constitute a significant portion of all biological research and were targeted to have the greatest impact on scientific discovery. They benefit

from an expanding array of genetic and molecular tools, making it possible to ask even more detailed questions through genotype quantification. I also focus on visual analysis because cameras represent a rich and exciting sensor that many researchers rely upon due to its non-invasive nature. Thus my techniques assume the behavioral trait is manifest as a change in body motion, orientation, or shape. Other types of sensors are necessary for behaviors that are not measured by images (e.g., audio processing of birdsong in zebra finches) [4, 109]. Finally, I focus on detailed posture estimation of these biological organisms. This involves video recording and measuring at a high enough resolution to capture detailed motions. For instance, quantifying the escape response behavior in zebrafish involves measuring the position of a complex time-varying surface (i.e., its body). A significant amount of previous work has been done to automatically track animals at low resolutions where their level of abstraction is a planar blob. These results have spawned the development of some commercially available software packages (e.g., EthoVision$^{\text{TM}}$, BIOBSERVE$^{\text{TM}}$), and lead to successful behavioral quantification of nematodes [27], zebrafish [5, 9], fruit flies, and rodents (see [4, 109] and the references therein). However, there are numerous unresolved scientific questions where this coarse level of measurement is insufficient. The expanding array of genetic and molecular tools permit scientists to manipulate the physiology of specific cells and circuits within the brain [6, 65]. With this fine level of genetic manipulation, an even finer assay is required to identify and quantify the behavioral effect. Thus, my goal is to leverage recent advances in computer vision to create a new generation of instruments that can operate with both the high throughput and resolution required for a productive synthesis with modern genetic approaches.

1.1 Automated Visual Tracking

This section reviews some of the related work in the computer vision field. Related work published in the biology field typically focuses on applications to a particular organism, so I will discuss this work in subsequent chapters when the particular organism is addressed. The problem of tracking moving objects continues to be an active area of research within computer vision, and advances in image segmentation and probabilistic inference continue to permit greater advances. The visual tracking

literature is large and vast, so this review is by no means exhaustive. The intended applications of tracking nematodes, zebrafish, and fruit flies overlaps with several different areas within the tracking literature, so my goal is to present some of the work within each of these subfields.

1.1.1 Planar Tracking

Tracking moving objects with changing shape continues to be an active area of research. One of the most predominant methods for tracking an object is to calculate the boundary of the object in the image. Geometric active contours formulated in the level-set framework has emerged as a general and robust method [78, 90] for calculating this boundary contour. A primary advantage of this approach is that its implicit representation of the contour permits changes in topology within the image. In addition, this framework can utilize different energy functionals based on edge detection [64, 79] or region-based statistics [18, 118, 120] to drive the contour evolution. The level-set framework also facilitates the incorporation of prior knowledge or object models through the use of shape priors. This allows for successful tracking when the images contain background clutter, noise, and/or occlusions. A nice review of these recent advances has been written by Cremers [21]

Previous level-set based trackers that use a dynamic state space framework include [19, 25, 53, 76, 82]. The advantage of the state space model is that it uses the temporal coherence of level sets to guide the tracking process. Instead of using the solution from the previous frame as the initial guess for the current frame, the state space model includes a model of the moving object's dynamics. Within the framework of Bayesian inference, this corresponds to a prior probability distribution on state of the system.

My model-based approach incorporates a quite general notion of a finitely parametrized generative model, as opposed to some prior works which use more limited model classes. In [19], Cremers uses a finite parameterization of shape deformations that consist of a linear combination of eigenmodes learned from principal component analysis (PCA) of training shapes. The training set is used to learn a second-order Markov Chain dynamic model. Using the image segmentation model of Chan and Vese [18], the author is able accurately track the periodic silhouette of a walking human

in extremely noisy images.

In [83], Rathi also uses an energy functional based on the Chan-Vese Model, and performs state estimation using a particle filter. Their state representation consists of a set of affine parameters and the contour itself. Prior knowledge about the object is incorporated into the update step of the particle filter, instead of being explicitly modeled in the state, as is done in our approach. In [25], Dambreville represents shape deformations using PCA, utilizes the Chan and Vese Model, and performs state estimation using an unscented Kalman filter (UKF). Although they explicitly incorporate the shape deformations into the state, their observation model relies on having an invertible function that transforms the current observed contour (embedded in signed distance function (SDF)) into the shape parameters.

This prior work in level-set based object tracking is not directly applicable to animal pose estimation because the underlying model representation does not describe meaningful kinematic parameters. While contour subspace models, such as PCA [19, 25, 105], kernel densities [20, 24], and locally linear embedding [88] are appropriate for many types of applications, they parameterize the shape deformations in a manner that is not physical. Instead, with the application of animal tracking in mind, I construct generative geometrical models whose finite parameterizations are adapted to the given problem. By specifying the degrees of freedom and deformations of the model, the state estimation process performs inference on physical quantities that are of interest to experimental goals. In Chapter 2, I show that the segmentation model of Chan and Vese can be formally formulated within the framework of an iterated Kalman filter. Using statistical linearization to solve the Kalman update equation, I present a novel observation model that converges to the optimal state estimate and incorporates the arbitrary user-defined generative model.

There also exists a significant amount of work on tracking multiple objects from a single camera. The well-known BraMBLe tracker by Isard introduces a novel multi-blob likelihood function to handle severe occlusions [51]. Branson extended the BraMBLe to include a contour model and applied the algorithm to tracking multiple mice from a side view [11]. Tweed used a subordinated sampling algorithm with occlusion reasoning to track multiple flying birds in wildlife footage [106].

Khan introduced a Rao-Blackwellized filter to match appearance templates for tracking a honeybee inside a hive with numerous other identical looking objects [62]. In a separate paper, he presented an algorithm for tracking multiple ants (upwards of 20) walking around in an arena [60]. Special care was taken to sample the high dimensional state space efficiently, and deal with split and merged measurements [61]. In other approaches, researchers have relied on the statistical fusion of various image cues such as color histograms, edges, and texture in order to design very general algorithms to track arbitrary rectangular regions in various types of video [80, 117].

The advantage of these algorithms is their ability to handle complex visual data and accurately track even in the presence of full occlusions. The trade off, however, is a much coarser model representation (i.e., rectangular box or blob). This is not sufficient to measure many of the kinematic parameters needed in modern biological studies. Instead, I design novel geometric generative models that are used to track zebrafish and multiple nematodes in Chapters 4 and 5. The algorithm estimates the detailed posture of these animals during behaviors of biological interest and measures important kinematic variables that were previously inaccessible except through laborious manual digitization. I utilize edges and region-based image statistics based on level-set segmentation to achieve accurate tracking during partial occlusions due to background clutter and other organisms.

1.1.2 3D Tracking

The overwhelming majority of papers discussing 3D model-based tracking have focused on the problem of markerless human motion capture. Thomas Moeslund et al. have compiled 2 extensive surveys that discuss in depth the advances in this field over the last 20+ years [72, 73]. The most widely used approach to addressing this problem is the direct use of an explicit human model where the kinematics, shape, and/or appearance are directly parameterized. Using what Moeslund calls an "analysis-by-synthesis" approach, this methodology optimizes the similarity between the model projections and the observed images. In some approaches, authors optimize this similarity by minimizing the distance between the model's projected boundary and image boundaries [29, 57, 100]. Here, the occluding contour of the model must be calculated since parts of the 3D model will undergo

self-occlusions when seen from a particular viewpoint. Others have utilized optical flow measurements to optimize the position of their human model [12, 102]. Carranza et al. perform a pixel-wise error metric calculation on a graphics processing unit (GPU) to maximize the overlap between the model and image silhouettes [17]. Instead of optimizing the overlap of projected silhouettes, Mikic et al. use the silhouettes to reconstruct voxel images and maximize the volumetric overlap with their model in 3D space [71]. Rosenhahn et al. recently introduced an exciting approach that uses level-set segmentation to bias the contour extraction towards the model projection, and the contour extraction to update the model pose parameters [86, 85]. In this approach, the projection rays are reconstructed from the image contour, and the distance between the model and rays is minimized in 3D space. All of these approaches utilize a gradient-descent or Kalman filter based estimator to solve for the local optimal solution. In [30, 59], the authors utilize a stochastic sampling based filter that is typically able to converge to a global optimal. They both introduce techniques to properly sample within the high-dimensional state of a fully articulated human model. While all of the previous studies assume the use of multiple cameras, Sminchisescu et al. have worked on several approaches to perform 3D pose estimation from a *single* camera [96, 97]. This is an extremely ill-conditioned problem because of the many depth ambiguities and unobservable degrees of freedom present in a monocular view, so carefully designed algorithms must be explored.

In Chapter 6, I present an algorithm to accurately estimate the 3D body and wing kinematics of fruit flies from multiple cameras. Like many previous techniques for human motion capture, I design an explicit model of the fly and follow an "analysis-by-synthesis" approach that minimizes the distance between the model and its corresponding projection rays according to [86]. However, I address several challenges unique to flies that are not present in human tracking. For instance, the rotational angle around the fly's longitudinal axis is nearly unobservable due to the body's cylindrical shape. Humans have cylindrical shaped limbs, but rotation about the axis of symmetry is negligible for nearly all biomechanical motions. In contrast, flies undergo significant rotations about this symmetric axis during various flight maneuvers. In order to compensate for this rotational ambiguity, I design a constraint function based on the gross symmetry of flapping flight to provide

a cue for the body's rotational angle about this axis of symmetry. This constraint is correctly incorporated into a Kalman filter based estimator so the state probability density maintains the proper statistics.

I also extend the prior motion model of Rosenhahn [85] to include quaternion representation of rotations. This motion model provides accurate predictions of pose, given pose estimates from previous time steps. Flies exhibit complex wing rotations during flapping flight that requires a global parameterization of $SO(3)$. Although a local parameterization using joint angles is sufficient for human motion, this same parameterization would undergo singularities for wing motion and result in incorrect prediction. This motion model representation is also velocity invariant, so it can provide accurate prediction for video captured at different frame rates and/or animals moving at different velocities. Also, it is not restricted to specific stereotyped motions (e.g., walking, running) and can easily switch between motion types within the same video sequence. I also use this motion model to predict the body undulations of swimming zebrafish in Chapter 4. Finally, in Chapter 7, I summarize the contributions made in this thesis and provide direction for extensions to the current algorithms.

Chapter 2

Techniques for Model-Based Visual Tracking

This chapter reviews the methodology used for visual tracking of model organisms, including prior theories that are needed to develop the approach. After summarizing the tracking approach, I then review a result of Bell and Cathey [7], who showed that the update equation of an iterated Kalman filter (IKF) is identical to the Gauss-Newton method for nonlinear least squares minimization. In my approach, it is convenient to see the Kalman filter as minimizing a particular error function. Next, I introduce the concept of statistical linearization (a.k.a *weighted statistical linear regression*) and the family of sigma point Kalman filters that use this approach to achieve greater accuracy than traditional extended Kalman filters. Then, an improved technique for incorporating nonlinear constraints into the Kalman filter framework is presented. This technique is important for animal tracking because the state represents the pose of the organism and constraining the state estimate against anatomically infeasible poses will improve the tracking performance in the presence of misleading image measurements. Next, I review the popular Chan and Vese model for level set image segmentation and formally integrate this technique into an IKF framework. Finally, I introduce a technique due to Rosenhahn [85] that provides a convenient way to incorporate a priori knowledge about the organism's motion into the dynamic prediction step of the Kalman filter. It allows the training data to be rescaled to different velocities and to consist of completely different motion patterns.

2.1 Generative Model-Based Image Tracking

In this work, models for vision-based animal tracking are constructed using a *generative modeling* approach [99, 98]. A generative model is constructed by applying a continuous transformation to a shape, known as a *generator*. Mathematically, the generator is represented by the parametric function $G(q) : \mathbb{R}^l \rightarrow \mathbb{R}^m$, and the continuous *transformation* by $T(x\,;r) : \mathbb{R}^m \times \mathbb{R}^k \rightarrow \mathbb{R}^n$, where $x \in \mathbb{R}^m$ is a point in space to be transformed, and $r \in \mathbb{R}^k$ is a parameter that defines the transformation. The resulting generative model is

$$H(p) = T(G(q)\,;r) : \mathbb{R}^{l+k} \rightarrow \mathbb{R}^n \tag{2.1}$$

where H is the surface defined by the model and $p = \begin{bmatrix} q \\ r \end{bmatrix}$ is the parameterization of the organism's shape, q, and its position, r, which can be defined, for example, by the angles of its joints. Because they are expressed as a composition of parametric functions, generative models exhibit *closure* (i.e., a generative model can be used as the generator of another generative model).

The state of the organism consists of the parameters that describe pose (i.e., the translation of a body fixed frame with respect to a reference frame) and shape (e.g., angles associated with the orientation of appendages), although other parameters can be included. The visual tracking problem involves sequentially estimating the state, p, via measurements, z, in a discrete time dynamic state space model

$$p_k = f(p_{k-1}, \xi_{k-1}) \tag{2.2}$$

$$z_k = h(p_k, \omega_k) \tag{2.3}$$

where ξ_k and ω_k are noise processes, and the subscript k denotes the value of a variable at the k^{th} time step. Equation (2.2) describes the underlying dynamic process that governs the organism's motion, while (2.3) describes the measurement process. I assume that the function $f(p, \xi)$ is available to describe the organism's motion between frames, and the measurements consist of different visual

cues from 2-dimensional images. The state space model provides a generic framework for model-based tracking. The functions $f(p, \xi)$ and $h(p, \omega)$ can be modified appropriately depending on the experimental parameters of the video. Examples of generative models are given in Chapter 3.

2.2 IKF Update as a Gauss-Newton Method

As shown in [7], a single iteration of the update equation in an iterated Kalman filter (IKF) is equivalent to a single Gauss-Newton iteration of a nonlinear least squares problem. To establish this equivalence, assume that the state estimate $\hat{p} \in \mathbb{R}^n$ and observation $z \in \mathbb{R}^m$ come from the multivariate normal distributions

$$\hat{p} \sim \mathcal{N}(p, \mathcal{P}), \qquad z \sim \mathcal{N}(h(p), \mathcal{R}). \tag{2.4}$$

From Bayes rule, the maximum a posteriori (MAP) estimate of p is found by maximizing $\mathrm{P}(p|z) \simeq \mathrm{P}(z|p)\mathrm{P}(p)$ with respect to z, where

$$\mathrm{P}(z|p) = \frac{\exp\Big(-\frac{1}{2}(z - h(p))^T \mathcal{R}^{-1}(z - h(p))\Big)}{\sqrt{(2\pi)^m |\mathcal{R}|}} \tag{2.5}$$

$$\mathrm{P}(p) = \frac{\exp\Big(-\frac{1}{2}(\hat{p} - p)^T \mathcal{P}^{-1}(\hat{p} - p)\Big)}{\sqrt{(2\pi)^n |\mathcal{P}|}} \tag{2.6}$$

where $h(p)$ is the observation function of (2.3). Maximizing the posterior distribution is equivalent to minimizing the negative log of the function $\mathrm{P}(z|p)\mathrm{P}(p)$,

$$\mathrm{E}(p) = \frac{1}{2}\Big[(z - h(p))^T \mathcal{R}^{-1}(z - h(p)) + (\hat{p} - p)^T \mathcal{P}^{-1}(\hat{p} - p)\Big] \tag{2.7}$$

after dropping a constant. By defining the following augmented state and its associated observation function and covariance,

$$Z = \begin{bmatrix} z \\ \hat{p} \end{bmatrix}, \qquad g(p) = \begin{bmatrix} h(p) \\ p \end{bmatrix}, \qquad Q = \begin{bmatrix} \mathcal{R} & 0 \\ 0 & \mathcal{P} \end{bmatrix},$$

Equation (2.7) can be written as

$$\mathrm{E}(p) = \frac{1}{2}\Big[(Z - g(p))^T Q^{-1} (Z - g(p))\Big]. \tag{2.8}$$

Letting $S^T S = Q^{-1}$ and defining

$$f(p) = S(Z - g(p)) \ , \tag{2.9}$$

then minimization of (2.8) is equivalent to a nonlinear least squares problem with an error functional of the form:

$$\mathrm{E}(p) = \tfrac{1}{2}\|f(p)\|^2. \tag{2.10}$$

The Gauss-Newton method for minimizing (2.10) defines a sequence over i of approximations p_i

$$p_{i+1} = p_i - (f'(p_i)^T f'(p_i))^{-1} f'(p_i)^T f(p_i) \tag{2.11}$$

that converges to a local minimum of (2.10) for problems where $||f(p)||$ is locally convex at the solution. Since $f'(p_i) = -SG_i$ where

$$G_i = \begin{bmatrix} \frac{\partial h_1}{\partial p_1} & \cdots & \frac{\partial h_1}{\partial p_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_m}{\partial p_1} & \cdots & \frac{\partial h_m}{\partial p_n} \\ 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix}, \qquad G_i \in \mathbb{R}^{m+n \times n} \tag{2.12}$$

is the Jacobian matrix of $g(p_i)$, then (2.11) becomes:

$$p_{i+1} = (G_i^T Q^{-1} G_i)^{-1} G_i^T Q^{-1} (Z - g(p_i) + G_i p_i). \tag{2.13}$$

After some matrix algebra and utilizing the matrix inversion lemma, (2.13) reduces to

$$p_{i+1} = \hat{p}_k + K_i\big(z - h(p_i) - H_i(\hat{p}_k - p_i)\big), \tag{2.14}$$

which is the IKF update equation for the k^{th} time step. H_i is the Jacobian matrix of $h(p_i)$ and $K_i = \mathcal{P} H_i^T (H_i \mathcal{P} H_i^T + \mathcal{R})^{-1}$ is the Kalman gain. At the first iteration ($i = 0$), $p_i = \hat{p}_k$, and after convergence, the updated state is set to the current value, $\hat{p}_k = p_{i+1}$. Thus, the update equation of an iterated Kalman filter is equivalent to a nonlinear least squares problem of the form (2.10) and (2.9).

2.3 Sigma Point Kalman Filters

In (2.14), there are several terms which involve the Jacobian matrix H_i. These terms introduce linearization errors by calculating the derivative at the state mean without taking into account the underlying uncertainty associated with the random variable. Recently, a number of related algorithms have been proposed that demonstrate significant improvement over the traditional extended Kalman filters (EKF) when applied to nonlinear motion and measurement models [52, 77, 56, 111, 92]. The authors in [111] collectively called these techniques sigma point Kalman filters (SPKF) due to their common use of statistical linearization [42]. This technique takes a weighted average of a set of regression points drawn from the prior distribution of the random variable. These regression points are selected to lie on the principle component axes of the input covariance. In this work, I utilize the weighting parameters described in [77], known as the central difference Kalman filter (CDKF), to estimate the mean and covariance of our regression points.

The CDKF is constructed as follows: Consider a nonlinear function $y = g(x)$ that is evaluated

at r points $(\mathcal{X}_i, \mathcal{Y}_i)$ with $\mathcal{Y}_i = g(\mathcal{X}_i)$. Let

$$\mathcal{X}_0 = \hat{x} \qquad\qquad w_0 = \frac{h^2 - L}{h^2} \tag{2.15}$$

$$\mathcal{X}_i = \hat{x} + \left(\sqrt{h^2 P_x}\right)_i \qquad i = 1, \ldots, L \qquad\qquad w_i = \frac{1}{2h^2} \quad i = 1, \ldots, 2L \tag{2.16}$$

$$\mathcal{X}_i = \hat{x} - \left(\sqrt{h^2 P_x}\right)_i \qquad i = L+1, \ldots, 2L \tag{2.17}$$

where L is the dimension of the state space and h is the half-step size of the central divided difference derivative estimate. Estimates of the mean and covariance are calculated from the Sigma points $\mathcal{X}$ and the transformed Sigma points $\mathcal{Z}$ using

$$\hat{y} = E[y] = \sum_{i=0}^{2L} w_i \mathcal{Y}_i \tag{2.18}$$

$$\mathcal{P}_{yy} = E[(y - \hat{y})(y - \hat{y})^T] = \sum_{i=0}^{2L} w_i (\mathcal{Y}_i - \hat{y})(\mathcal{Y}_i - \hat{y})^T \tag{2.19}$$

$$\mathcal{P}_{xy} = E[(x - \hat{x})(y - \hat{y})^T] = \sum_{i=0}^{2L} w_i (\mathcal{X}_i - \hat{x})(\mathcal{Y}_i - \hat{y})^T. \tag{2.20}$$

Based on a survey of the literature, the *unscented* Kalman filter (UKF) appears to be more widely used than the CDKF. The authors in [111] note that, in practice, they have not observed any difference in performance between the two filters, although both are clearly better than the EKF. I choose the CDKF because the authors in [77] show that it has a theoretically higher accuracy covariance estimate than the UKF and is parameterized by a single scalar which has an optimal value for Gaussian noise distributions (the UKF has 3 parameters). In addition, I utilize the technique introduced by the authors in [92] who add iteration (see Section 2.2) to the Sigma point approach. In that way, (2.14) can be modified such that the Kalman gain and other linearized terms are estimated using (2.18) – (2.20),

$$\begin{aligned} p_{i+1} &= \hat{p}_k + K_i\big(z - h(p_i) - H_i(\hat{p}_k - p_i)\big) \\ &= \hat{p}_k + \mathcal{P}_{pz}\big(\mathcal{P}_{zz} + \mathcal{R}\big)^{-1}\left(z - h(p_i) - \mathcal{P}_{pz}^T \mathcal{P}_{pp}^{-1}(\hat{p}_k - p_i)\right) \end{aligned} \tag{2.21}$$

and the covariance is updated according to

$$\begin{aligned}\mathcal{P}_{pp,k+1} &= \mathcal{P}_{pp,k} + KH_i\mathcal{P}_{pp,k} \\ &= \mathcal{P}_{pp,k} + \mathcal{P}_{pz}\big(\mathcal{P}_{zz} + \mathcal{R}\big)^{-1}\mathcal{P}_{pz}^T. \end{aligned} \tag{2.22}$$

One of the most expensive calculations in the SPKF approach is the matrix square root operation in (2.16) – (2.17) used to construct the Sigma point set. To address this burden, a *square-root form* of the SPKF is implemented that propogates and updates the square root of the state covariance directly in Cholesky factored form, S_p, where $\mathcal{P}_{pp} = S_p S_p^T$ [110]. This technique also leads to better numerical stability because it ensures that the covariance is always positive definite. In general, the order complexity of the square root SPKF is $\mathcal{O}(L^3)$, which is identical to the complexity of the EKF. However, each component of the iterated filter has a slightly different complexity. The prediction step is $\mathcal{O}(L^3)$, however, the state and covariance update steps are $\mathcal{O}(LM^2)$, where M is the measurement dimension (Figure 2.1). Because $M \gg L$ in most tracking applications (i.e., many measurements are used to estimate a few parameters), the update step is typically the computational bottleneck in Kalman filter estimation.

Although the SPKF provides greater estimation accuracy than the EKF in the presence of nonlinear process and measurement equations, it assumes that the state variables follow normal distributions. This assumption is often violated in many visual tracking scenarios that contain multiple objects or occlusions. In this situation, sequential Monte Carlo (SMC) techniques (i.e., particle filters) [34] provide solutions for state estimation when the posterior distribution is multi-modal. Although particle filters are able to solve very general estimation problems, there exist many problems within visual tracking when the normal assumption holds, in which case Kalman filters provide accurate solutions and computational efficiency. This thesis demonstrates that the normal assumption holds for estimating the location of biological organisms within a constrained laboratory that contains few environmental occlusions, hence Kalman filtering is adopted. Further discussion will be provided in Chapter 7 about potential improvements to the nonlinear estimation techniques

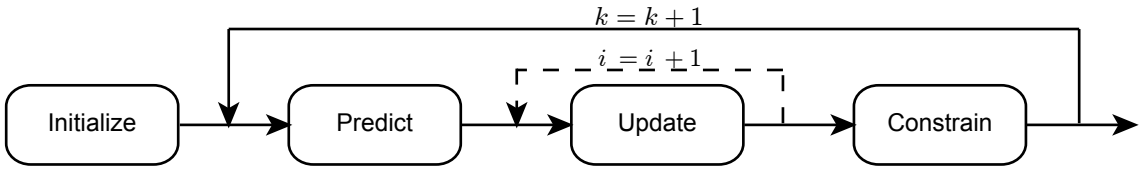


Figure 2.1: Constrained Filter — the iteration over i is based on the iterated Kalman filter approach presented in Section 2.2. The "Update" is an unconstrained minimization routine, thus the constraints are applied to its output. Each block in the filter contains nonlinear equations, and the Sigma point transform (Section 2.3) is applied to each one in order to accurately propagate the state mean and covariance through time.

applied in this study.

2.4 Incorporating Nonlinear Constraints

Many techniques for incorporating nonlinear constraints into the Kalman filter framework have been developed [28, 94, 116]. These methods typically utilize a pseudo-observation approach (i.e., the constraint is an observation with zero variance) or an operator to project the estimate onto the constraint surface (a more detailed review is provided in [55]). However, in these approaches, the estimate is not guaranteed to satisfy the constraint, and they often lead to singular covariance matrices. Recently, Julier and LaViola introduced a two-step approach utilizing the Sigma Point transform that first constrains the probability distribution and then constrains the conditional mean of the distribution [55]. A brief overview of their technique is provided here. An equality constraint between state variables can be written in the form

$$\mathbf{c}(p_k) = 0 \tag{2.23}$$

and presumes the existence of a projection function $\mathbf{w}(p_k)$ such that

$$\mathbf{c}\big(\mathbf{w}(p_k)\big) = 0 \qquad \forall p_k \in \mathbb{R}^n. \tag{2.24}$$

Assume the unconstrained estimate with mean $\hat{p}_k^*$ and covariance $\mathcal{P}_k^*$ has already been calculated (I eliminate the covariance subscript associated with the random variable for readability). First, the

projection operator is applied to every point in the distribution. This additional information reduces the uncertainty of the distribution and causes the covariance to decrease [55]. Let $p_k^\dagger = \mathbf{w}(p_k^*)$ be the constrained state, so that the mean and covariance of the constrained estimate are given by

$$\hat{p}_k^\dagger = E\big[\mathbf{w}(p_k^*)\big] \tag{2.25}$$

$$\mathcal{P}_k^\dagger = E\big[(p_k^\dagger - \hat{p}_k^\dagger)(p_k^\dagger - \hat{p}_k^\dagger)^T\big]. \tag{2.26}$$

Unless the constraints are linear, the expectation of a constrained distribution will not lie on the constraint surface itself (see [55] for a proof), so $\hat{p}_k^\dagger$ will not obey the constraint. Therefore, the projection operation is applied again to the mean of the constrained estimate. In addition, the covariance is increased to account for the fact that the minimum mean squared estimate is adjusted to a different value

$$\hat{p}_k = \mathbf{w}(\hat{p}_k^\dagger) \tag{2.27}$$

$$\mathcal{P}_k = \mathcal{P}_k^\dagger + (\hat{p}_k^\dagger - \hat{p}_k)(\hat{p}_k^\dagger - \hat{p}_k)^T. \tag{2.28}$$

In this example, the expectations in (2.25) and (2.26) are calculated using the Sigma Point transform presented in Section 2.3. This step in the algorithm is $\mathcal{O}(L^3)$, like the prediction step.

2.5 Level Set Segmentation

The use of geometric active contours, formulated in the level-set framework, has emerged as one of the most general and robust methods for image segmentation [90, 78]. One of the primary advantages of this approach is its ability, due to its implicit representation, to handle changes in contour topology. In order to determine the boundary that divides the image into foreground and background regions (i.e, the image boundary that divides the tracked animal's body from the background), a contour Γ is embedded into the zero level set of a signed distance function (SDF) $\phi : \Omega \to \mathbb{R}$ on the image

domain $\Omega \subset \mathbb{R}^2$:

$$\Gamma = \{\mathrm{x} = (x, y) \in \Omega \mid \phi(\mathrm{x}) = 0\} \tag{2.29}$$

$$\phi(\mathrm{x}) = \begin{cases} \min\limits_{\mathrm{x_I} \in \Gamma}(\|\mathrm{x} - \mathrm{x_I}\|) & x \in \Omega_1 \ (\textit{inside}) \\ -\min\limits_{\mathrm{x_I} \in \Gamma}(\|\mathrm{x} - \mathrm{x_I}\|) & x \in \Omega_2 \ (\textit{outside}). \end{cases} \tag{2.30}$$

The value of $\phi(\mathrm{x})$ is $\pm$ the distance between x and the closest point on the boundary, Γ. By minimizing an energy functional, Γ is deformed implicitly by evolving the function ϕ. Many different image-based energy functionals have been proposed, based on edge detection [64, 79] or region-based statistics [18, 118, 120], to drive the contour evolution. They typically impose three different criteria to extract the contour: (1) the pixels within each region (i.e., inside or outside) should be similar, (2) the pixels between regions should be dissimilar, and (3) the contour dividing the regions should be minimal. These criteria are all incorporated into the model

$$\mathrm{E}(p_1, p_2, \phi) = -\int_\Omega \mathrm{H}(\phi) \log p_1(I(\mathrm{x})) + (1 - \mathrm{H}(\phi)) \log p_2(I(\mathrm{x}))\, d\mathrm{x} + \nu \int_\Omega |\nabla \mathrm{H}(\phi)|\, d\mathrm{x} \tag{2.31}$$

where minimizing the first and second terms maximizes the posterior probability given by the densities p_1 and p_2, which measure the fit of an intensity value $I(\mathrm{x})$ to the corresponding region. $\mathrm{H}(\cdot)$ is a regularized Heaviside function and $\nu > 0$ is a scalar weighting parameter for the the third term, which penalizes the contour length. Although there are many ways to model the probabilities densities p_1 and p_2 (see [86] for a detailed overview), one approach is to assume they are normal densities with fixed standard deviation. This special case of (2.31) was introduced by Chan and Vese and has become one of the most widely used image functionals [18]. It has the form:

$$\mathrm{E_{cv}}(c_1, c_2, \phi) = \int_\Omega \mathrm{H}(\phi)\lambda_1 (I - c_1)^2 + (1 - \mathrm{H}(\phi))\lambda_2 (I - c_2)^2\, dx\, dy + \nu \int_\Omega |\nabla \mathrm{H}(\phi)|\, dx\, dy, \tag{2.32}$$

where c_1 and c_2 are the means of the normal densities and $\lambda_i = \frac{1}{2\sigma_i^2}$ $i = 1, 2$ are proportional to the variance. The means are defined as $c_1(\phi) = \frac{\int_\Omega I(x,y)\mathrm{H}(\phi) dx\, dy}{\int_\Omega \mathrm{H}(\phi) dx\, dy}$ and $c_2(\phi) = \frac{\int_\Omega I(x,y)(1-\mathrm{H}(\phi)) dx\, dy}{\int_\Omega (1-\mathrm{H}(\phi)) dx\, dy}$ and

are updated as the contour evolves to calculate the average grayscale intensity inside and outside the contour. In the Chan-Vese model, the maximum posterior probability is achieved by maximizing the grayscale homogeneity of the image I inside and outside the contour. Typically, $\lambda_1 = \lambda_2 = 1$ so that both regions are penalized equally (i.e., both regions assumed to have the same variance). I adopt the Chan-Vese model for my tracking algorithm because the high-contrast images produced in laboratory environments typically consist of a dark animal on a lighter background, or vice-versa.

When prior knowledge about the object being tracked is available, this information can be incorporated into the level set framework in two possible ways. The first approach introduces the shape prior at the variational level by adding to the energy functional an additional term that draws the contour closer to a prior shape,

$$\mathrm{E}(c_1, c_2, \phi, \chi) = \mathrm{E}_{\mathrm{cv}} + \lambda \int_{\Omega} (\phi - \phi_0(\chi))^2 \, dx \tag{2.33}$$

where χ are the model parameters of the shape prior that encode shape and pose information and λ is a scalar weighting term. A discussion on techniques that properly incorporate the dissimilarity measure of the shape prior is presented in [22]. The second approach explicitly parameterizes the signed distance function ϕ by the model parameters. This approach is utilized by Tsai et. al. to accurately segment MRI scans of the left ventricle and prostate gland [104, 105]. They redefine the Chan and Vese model as

$$\mathrm{E}_{\mathrm{cv}}(\chi) = \int_{\Omega} \mathrm{H}(\phi(\chi))(I - c_1(\phi(\chi)))^2 + (1 - \mathrm{H}(\phi(\chi)))(I - c_2(\phi(\chi)))^2 \, dx \, dy \tag{2.34}$$

and perform gradient descent directly on the shape and pose parameters, χ. The "length" term is omitted because the underlying parameterization of ϕ prevents the contour from growing arbitrarily. I also define ϕ as a function of the shape and pose parameters, however, instead of using principal component analysis to model the shape deformations as is done in [105], I use a generative model (2.1) to encode all of the appropriate variables. In the next section, I demonstrate how this explicit parameterization approach is used to incorporate the level set image functional into the observation

function of the dynamic state space model (2.3).

2.6 The Chan-Vese Model within an IKF Framework

Returning to the level set framework of Section 2.5, I assume that the segmenting contour Γ and associated signed distance function ϕ are parameterized by a state vector p. Following [104], I also eliminate the "length" term in the Chan and Vese model. Then (2.32) becomes

$$\mathrm{E_{cv}}(c_1, c_2, p) = \int_\Omega \mathrm{H}(\phi(p))(I - c_1)^2 + (1 - \mathrm{H}(\phi(p)))(I - c_2)^2 \, dx \, dy. \tag{2.35}$$

To show how the Chan-Vese segmentation model can formally fit into an IKF framework, I define the observation function of the state space model (2.3) as:

$$h(p) = \begin{cases} c_1 & \phi(p) \geq 0 \\ \\ c_2 & \phi(p) < 0 \end{cases}. \tag{2.36}$$

Note that the c_1, c_2 are functions of ϕ, which in turn is a function of p, and so $h(p)$ is differentiable, except at $\phi(p) = 0$. In the variational framework of (2.35), this non-differentiability is due to the Heaviside function, so a "smoothed" or regularized Heaviside is used instead (e.g., error function or hyperbolic tangent). In the Kalman filter framework, the use of statistical linearization (Section 2.3) differentiates $h(p)$ using a set of regression points, thus avoiding the non-differentiability at $\phi(p) = 0$. For implementation on images with a finite number of pixels, (2.35) is converted to a discrete representation:

$$\mathrm{E_{cv}}(c_1, c_2, p) = \sum_i \sum_j (I(x_i, y_j) - h(x_i, y_j \,; p))^2 \tag{2.37}$$

$$= \|(I - h(p))\|^2, \tag{2.38}$$

where $I(x_i, y_j)$ is the image intensity at the pixel with coordinates (x_i, y_j). Note that the Chan-Vese model tries to minimize the difference in pixel intensity between the observed image and a binary-valued image, defined by the observation function h of (2.36), across the image domain Ω. Assuming that the covariance of the measurement variable z (in (2.3)) takes the value $\mathcal{R} = \mathbb{I}$, the identity matrix[1], and rewriting the squared Mahalanobis distance as

$$\|y - p\|_\Sigma^2 := (y - p)^T \Sigma^{-1} (y - p) \tag{2.39}$$

the error function (2.7) can be rewritten as a MAP estimate:

$$\begin{aligned} \mathrm{E}(p) &= \frac{1}{2}\Big[\|z - h(p)\|_{\mathcal{R}}^2 + \|\hat{p} - p\|_{\mathcal{P}}^2\Big] && (2.40)\\ &= \frac{1}{2}\Big[\|I - h(p)\|_{\mathbb{I}}^2 + \|\hat{p} - p\|_{\mathcal{P}}^2\Big] && (2.41)\\ &= \frac{1}{2}\Big[\mathrm{E}_{\mathrm{cv}} + \|\hat{p} - p\|_{\mathcal{P}}^2\Big]. && (2.42) \end{aligned}$$

That is, the error function takes the nonlinear least squares form of (2.8) and (2.9), whose iterative minimization is equivalent to the IKF update equation. Thus, I have shown that the Chan and Vese segmentation model can be implemented as the observation term of an iterated Kalman filter when one defines the observation function according to (2.36). The second term in (2.42), $\|\hat{p} - p\|_{\mathcal{P}}^2$, is equivalent to a "shape prior" or "knowledge-driven" term. It draws the current state towards the prediction and weights it according to the statistical certainty, while the first term segments based solely on image intensity.

2.6.1 Discussion

The region-based tracking approach presented here has several advantages. First, instead of having to search for features in the image to associate with corresponding model locations, this tracker utilizes the distribution of intensity values across the entire image to search for the solution. Second, by

[1] From our derivation of the Chan-Vese model in Section 2.5, this is equivalent to weighting the inside and outside regions equally.

utilizing the statistical linearization methods, I eliminate the need to directly calculate the Jacobian matrices associated with our process and observation equations. Finally, it should extend to 3D tracking using multiple camera views with the appropriate combined observation model, although the formulation presented here is only set up for planar tracking. In practice, this approach is used in conjunction with a feature-based contour tracking method to benefit from the strengths of each (Section 5.4).

Finally, I note that by proper definition of an observation function, other image-based energy functionals can also be formally integrated into an IKF framework. For instance, the *Binary Mean Model* proposed by Yezzi et. al. [118] is equivalent to an observation equation of the form

$$h_{BM}(p) = c_1(p) - c_2(p) \tag{2.43}$$

and setting the Kalman gain $K_i = -K_i$ so that the difference between c_1 and c_2 is maximized instead of minimized. Also, the multiphase version of the Chan-Vese model [113] can be integrated using a similar approach. Here, m signed distance functions ϕ are used to calculate 2^m regions in the image. In the case of $m = 2$, the observation function has the form

$$h(p) = \begin{cases} c_1 & \phi_1(p) \geq 0 \quad \& \quad \phi_2(p) \geq 0 \\ c_2 & \phi_1(p) \geq 0 \quad \& \quad \phi_2(p) < 0 \\ c_3 & \phi_1(p) < 0 \quad \& \quad \phi_2(p) \geq 0 \\ c_4 & \phi_1(p) < 0 \quad \& \quad \phi_2(p) < 0 \end{cases} . \tag{2.44}$$

Although this extends for an arbitrary number of signed distance functions ($m > 2$), it forces the model to segment 2^m regions. Brox and Weickert present an alternative multiphase approach which does not require the number of regions to be a factor of two [13]. Thus, the framework I propose here is quite general beyond the binary Chan-Vese model.

2.7 Scaled Motion Dynamics

Like many tracking systems, I employ a priori knowledge of the animal's dynamics, which in the Kalman filter tracking framework effectively restricts the search space of possible pose configurations and achieves robustness to self-occlusions and misleading visual cues. Many techniques to incorporate a prior model of the motion rely upon dimensionality reduction. The idea here is that typical motion patterns (e.g., flapping fly wings) can be represented as a simple trajectory in a low-dimensional subspace. Since all the limb movements are coupled, the motion is modeled by the mapping between the original, high-dimensional space (i.e., joint angles) and the low-dimensional subspace. Successful applications of this technique exist within the human motion tracking literature, where the motion model is calculated using PCA [93] and Gaussian processes [108]. In [95], it was suggested to learn a Gaussian mixture of motion patterns directly in the nonlinear subspace. However, the disadvantage of dimensionality reduction is the subspace representation is not invariant with respect to velocity. Thus, to have a motion model that provides accurate predictions when images are captured at different frame rates, the training data must include the same pattern, but taken at different velocities. In addition, dimensionality reduction works well for a single motion pattern, but in the case of multiple different motion patterns, a mixture of regressors is needed, which is difficult to estimate [54]. An alternative to this is presented by Rosenhahn et al. who model the motion patterns in the original high-dimensional space [85]. This offers a convenient way to incorporate training data that can be rescaled to different velocities and consist of completely different motion patterns.

Here, I provide a review of the technique proposed by Rosenhahn [85] that was applied to human motion tracking. Assume a model represented by a kinematic chain. The consecutive evaluation of exponential functions and twists ξ_i with known joint locations will apply a rigid body transformation of an end effector point X_i

$$X_i' = \exp(\theta\hat{\xi})\big(\exp(\theta_1\hat{\xi}_1)\dots\exp(\theta_n\hat{\xi}_n)\big)X_i \tag{2.45}$$

where X_i is in homogeneous coordinates, $\theta\hat{\xi} = \left[\begin{smallmatrix} \hat{\omega} & v \\ 0 & 0 \end{smallmatrix}\right]$ is the matrix representation of a twist $\xi \in se(3) = \{(v, \omega) \mid v \in \mathbb{R}^3,\ \hat{\omega} \in so(3)\}$ (see [74] for more details). An additional rigid body motion is applied after the joint angles to transform the model in space with respect to a fixed reference frame. The pose of the kinematic chain is denoted by the $(6 + n)$ dimensional vector $p = (\xi, \theta_1, \ldots, \theta_n) = (\xi, \Theta)$ consisting of the 6 degree of freedom for the rigid body motion ξ and the joint angle vector Θ.

Now, assume a set of temporally ordered training samples is available

$$\{\widetilde{p}_i := (\widetilde{\xi}_i, \widetilde{\theta}_{1,i}, \ldots, \widetilde{\theta}_{n,i}) := (\widetilde{\xi}_i, \widetilde{\Theta}_i) | i = 0 \ldots N\} \tag{2.46}$$

with body transformations relative to the fixed observing frame given by the twist $\widetilde{\xi}_i$ and joint angle vectors $\widetilde{\Theta}_i$. I denote this list of training samples as $\mathcal{P} = \langle \widetilde{p}_i \ldots \widetilde{p}_N \rangle$, and the sublist in $\mathcal{P}$ of length m ending at time i by $\langle \widetilde{p}_{i-m+1} \ldots \widetilde{p}_i \rangle$. Assuming that m frames of a video sequence have already been tracked (I use $m = 5$), the goal is to predict the pose at the next time step $p_{k+1} = (\xi_{k+1}, \Theta_{k+1})$ given the list of previously computed states $\langle p_{k-m+1} \ldots p_k \rangle$. To realize this estimate, the sublist in $\mathcal{P}$ that best matches the previous poses $\langle p_{k-m+1} \ldots p_k \rangle$ is determined. For the matching to be invariant with respect to the velocity of the tracked organism, the comparison is performed at different scalings s of $\mathcal{P}$. The different scalings of the training data, denoted $\mathcal{P}^s$, are calculated using linear interpolation and resampling. The resulting scaled lists are given by $\mathcal{P}^s = \{\widetilde{p}_i^s := (\widetilde{\xi}_i^s, \widetilde{\theta}_{1,i}^s, \ldots, \widetilde{\theta}_{n,i}^s) | i = 0 \ldots sN\}$. Rosenhahn et al. scan the interval $s = [0.5 \ldots 2]$ with stepsize 0.1 for their human video, although other choices are possible. The best matching sublist in the training data is now calculated by

$$\underset{s,j}{\operatorname{argmin}} \sum_{v=0}^{m-1} \left(\sqrt{\sum_{t=1}^{n} (\theta_{t,k-v} - \widetilde{\theta}_{t,j-v}^s)^2} \right). \tag{2.47}$$

Only the joint angles are taken into account since their motion will be invariant with respect to the global orientation of the model. With the optimal scale and position in the prior set, the relative joint motion between the optimal location in the prior set and the subsequent orientation is calculated.

Then, this relative motion is applied to the current state

$$\Theta_{k+1} = \Theta_k + (\widetilde{\Theta}^s_{j+1} - \widetilde{\Theta}^s_j). \tag{2.48}$$

Similarly, the predicted body motion ξ_{k+1} is calculated from relative motion between the twists at the optimal scale and position $\widetilde{\xi}^s_j$, $\widetilde{\xi}^s_{j+1}$. This calculation is carried out using an adjoint transformation, and I refer the reader to [85] for more details. Thus, this motion model predicts the pose in the next frame by matching tracked motion patterns to patterns in a training set. This matching is performed at different scalings to make it invariant with respect to time. Also, one can easily combine training data from very different motion patterns in order to track video sequences consisting of two or more such patterns.

2.8 Summary

In this chapter I reviewed the result of Bell and Cathey [7], who showed equivalence between the update equation of an iterated Kalman filter (IKF) and the Gauss-Newton method for nonlinear least squares minimization. In addition, the technique of statistical linearization was presented, which offers an improved method for solving the IKF. Statistical linearization was also used to incorporate nonlinear equality constraints into the Kalman filter framework. After reviewing the level set method for image segmentation, I introduced a formulation to incorporate the Chan-Vese image functional into the IKF framework. After providing brief extensions to this formulation, I introduce the motion model of Rosenhahn which models the parameters in their original high-dimensional state. This type of motion model is used in Sections 4.2 and 6.4 to predict the motions of swimming zebrafish and flying *Drosophila*, respectively.

Chapter 3

Constructing Generative Models of Organisms for Visual Tracking

In [98], Snyder presents a range of generative surfaces and transformations that can be used to construct a wide array of objects from spoons to teddy bears. In this thesis, I make extensive use of a particular generative surface known as a *Frenet Tube*. This surface consists of a tube constructed around a parameterized curve in space using the local Frenet frame. In the following sections, I present a brief mathematical description of a Frenet tube and illustrate how this surface can be used to construct physically accurate models of nematodes, zebrafish, and fruit flies.

3.1 The Frenet Tube

A parameterized, differentiable curve is a map $\gamma : \mathbf{I} \rightarrow \mathbb{R}^3$ of an open interval $\mathbf{I} = (a, b)$ of the real line $\mathbb{R}$ into $\mathbb{R}^3$. The tangent vector to the curve is denoted $\gamma'(u) = \big(x'(u), y'(u), z'(u)\big) \in \mathbb{R}^3$. I assume the curve to be regular, meaning that $\gamma'(u) \neq 0 \quad \forall\ u \in \mathbf{I}$. The unit tangent vector is denoted $\mathbf{T}(u) = \frac{\gamma'(u)}{\|\gamma'(u)\|}$. Without loss of generality, I also assume that the curve γ is parameterized by arc length such that $\|\gamma'(u)\| = 1$. This restriction is not necessary, but simplifies derivation of the local Frenet frame. The second derivative of the curve, $\gamma''(u)$ is a vector that is normal to the tangent vector. This is shown by differentiating the dot product of the tangent vector:

$$\frac{\partial}{\partial u}\Big(\gamma'(u) \cdot \gamma'(u)\Big) = \frac{\partial}{\partial u}\Big(1\Big) \tag{3.1}$$

$$\gamma''(u) \cdot \gamma'(u) + \gamma'(u) \cdot \gamma''(u) = 0 \tag{3.2}$$

$$\gamma''(u) \cdot \gamma'(u) = 0. \tag{3.3}$$

Thus, $\gamma''(u)$ is perpendicular to the tangent vector. The magnitude of the second derivative is defined as the curvature, $\|\gamma''(u)\| = \kappa(u)$, so that the vector normal to the tangent can be written as $\gamma''(u) = \kappa(u)\mathbf{N}(u)$, where $\mathbf{N}(u)$ denotes the unit normal vector. The plane defined by the unit tangent and normal vector, $\mathbf{T}(u), \mathbf{N}(u)$ is called the *osculating plane* at u. The vector normal to this plane is called the binormal vector $\mathbf{B}(u) = \mathbf{T}(u) \times \mathbf{N}(u)$, which is calculated from the vector cross product of the tangent and normal vectors.

The vectors $\mathbf{T}(u), \mathbf{N}(u), \mathbf{B}(u)$ define a local orthonormal basis at each point $(x(u), y(u), z(u))$ along the curve $\gamma(u)$. This basis is called the *Frenet frame*. The parameterized surface, $H(u, v)$, of a tube of radius r and length L around the curve $\gamma(u)$ can be constructed using the Frenet frame by

$$H(u, v) = \gamma(u) + r\Big(\cos(v)\mathbf{N}(u) + \sin(v)\mathbf{B}(u)\Big), \qquad u \in [0, L],\ v \in [0, 2\pi]. \tag{3.4}$$

Within the generative modeling framework (Section 2.1), the curve $\gamma(u)$ is the *generator* of the model, while expanding it in the $\mathbf{N}, \mathbf{B}$ plane is the *transformation* that creates the parameterized surface. A more general form of (3.4) is given when the the radius of the tube is allowed to vary smoothly along the length of the curve according to the function $R(u)$,

$$H(u, v) = \gamma(u) + R(u)\Big(\cos(v)\mathbf{N}(u) + \sin(v)\mathbf{B}(u)\Big), \qquad u \in [0, L],\ v \in [0, 2\pi]. \tag{3.5}$$

In addition to tubes, a *Frenet ribbon* is another parameterized surface that is constructed in a similar manner by only expanding the curve in the direction of the normal vector. This surface is given by

$$H(u, v) = \gamma(u) + R(u)\Big(v\mathbf{N}(u)\Big), \qquad u \in [0, L],\ v \in [-1, 1]. \tag{3.6}$$

When implemented within a model-based visual tracking system, the continuous parametric surfaces

defined in (3.5) and (3.6) must be evaluated on a discrete parametrized mesh. Using (3.6) as an example, the construction of the mesh involves sampling u and v at a fixed number of samples in their respective domains

$$H(u_i, v_j) = \gamma(u_i) + R(u_i)\Big(v_j \mathbf{N}(u_i)\Big) \tag{3.7}$$

$$u_i \in [0, L],\ i = 1, \ldots, N_u \qquad v_j \in [-1, 1],\ j = 1, \ldots, N_v. \tag{3.8}$$

Now, assume that the curve $\gamma(u)$ is additionally parameterized by a parameter p that controls its shape, $\gamma(u; p)$. If fixed values are chosen for the number of samples in the discrete parameter space (e.g., $N_u = 30$, $N_v = 3$), then the surface can be redefined as a function of only the shape parameters p

$$H(p) = \gamma(u_i; p) + R(u_i)\Big(v_j \mathbf{N}(u_i)\Big), \qquad u \in [0, L],\ v \in [-1, 1]. \tag{3.9}$$

This will now define the continuous parametric surface as a discrete mesh whose shape is changed according to p. The parametric surface can also be rendered to the image plane by projecting points along the surface according to a known calibrated camera model (see Appendix A for details on camera calibration). Let $X_{ij} = \begin{bmatrix} H(u_i, v_j) \\ 1 \end{bmatrix}$ denote a point on the parametrized surface in homogeneous coordinates, then its projection into the camera reference frame is given by

$$\lambda \begin{bmatrix} u_{ij} \\ v_{ij} \\ 1 \end{bmatrix} = P X_{ij} \tag{3.10}$$

where $P \in \mathbb{R}^{3\times 4}$ is the camera projection matrix and u_{ij}, v_{ij} are the pixel coordinates of the point

X_{ij}. The binary image I_H of the projected model points is given by

$$I_H(u,v) = \begin{cases} 1 & (u,v) = (u_{ij}, v_{ij}) \ \forall i,j \\ 0 & \text{otherwise} \end{cases} \tag{3.11}$$

$$= \text{Render}\Big(H(u,v)\Big). \tag{3.12}$$

In the case of a discrete parameterized surface, as assumed above, one has to "fill in the gaps" in the projected image to obtain a continuous representation of the projected silhouette. This is achieved in hardware using a graphics rendering program such as OpenGL or in Matlab$^{\text{TM}}$ using the *roipoly* function.

In the following sections, I use this framework of a Frenet tube/ribbon to construct parameterized models of nematodes, zebrafish, and fruit flies. Given this framework, the parameterized surface will be completely defined once the curve $\gamma(u)$ and radius profile $R(u)$ have been defined. The tubes and ribbons are able to accurately reconstruct the shape of the organisms to facilitate detailed visual tracking.

3.2 The Worm Model

The motion of *C. elegans* on a typical microscope slide is largely planar, so I restrict the modeling of the worm to two dimensions and assume orthographic projection in the camera model. To a good approximation, *C. elegans* maintains a constant length and width during locomotion. Although a worm may undergo elongations and contractions during mating behavior, the magnitude of these effects is small enough to ignore. Therefore, the centerline of each worm is modeled as a planar, inextensible curve. The body posture can be effectively described by the bend angle of the worm as a function of distance along its length. This function, $\Theta(u)$, becomes the *generator* of the model

and is finitely parameterized using a linear combination of known basis functions, $\Phi_j^k(u)$:

$$\Theta(u) = \sum_{j=1}^{N_\Theta} \alpha_j \Phi_j^k(u). \tag{3.13}$$

Although other choices are possible, the current implementation uses a second order ($k = 2$) periodic B-spline basis with $N_\Theta = 8$ (See Appendix B for details on B-splines). The organism's centerline is constructed by integrating the unit tangent vector $\mathbf{T}$ to the centerline curve, and the width is created by expanding the surface in the direction of the normal vector $\mathbf{N}$. This *transformation* creates the parameterized generative model $H(p)$:

$$\mathbf{T}(u) = \begin{bmatrix} \cos(\Theta(u)) \\ \sin(\Theta(u)) \end{bmatrix}, \qquad \mathbf{N}(u) = \begin{bmatrix} -\sin(\Theta(u)) \\ \cos(\Theta(u)) \end{bmatrix} \tag{3.14}$$

$$H(p) = \beta\Big(\int_0^u \mathbf{T}(\hat{u})d\hat{u} + v\, R(u)\mathbf{N}(u) \Big) + \vec{T} \qquad u \in [-\tfrac{L}{2}, \tfrac{L}{2}], \quad v \in [-1, 1] \tag{3.15}$$

where β is a constant scaling term with units $\frac{pixels}{mm}$, $\vec{T}$ is a global translation vector to the worm's center of mass, $p = \begin{bmatrix} \alpha_j & \vec{T} \end{bmatrix}^T$, L is the organism's length, and $R(u)$ is a continuous function for the organism's width as a function of distance along its length. Currently, $R(u)$ is defined as a fourth order open B-spline using 20 control points and is calculated offline using a semi-automatic initialization routine (Section 5.3). For the experiments involving actual video, the scaling term β is calculated from a test image using a stage micrometer. An overview of this modeling approach is illustrated in Figure 3.1. Thus, the worm model is a planar Frenet ribbon whose midline bends according to α_j and is translated in the plane according to $\vec{T}$.

This approach to modeling *C. elegans* offers several advantages. Since each B-spline basis function is only defined over a subset of the u domain (see Appendix B), the control points, α_j, have local control over the shape. This property is analogous to the worm's anatomy, where groups of muscles contract over local body regions. In addition, the parametrization of the centerline offers a natural

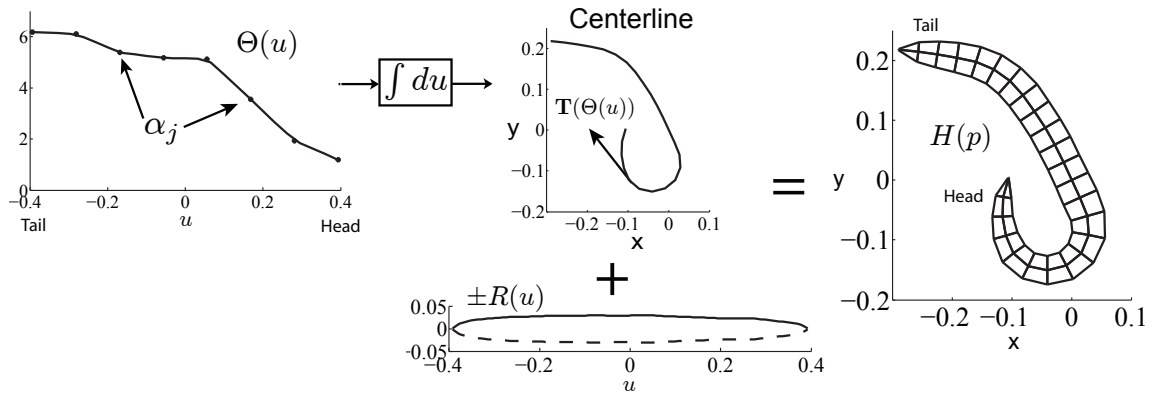


Figure 3.1: *C. elegans* generative model. α_j are control points that control the shape of $\Theta(u)$. The centerline is constructed by integrating the tangent vector $\mathbf{T}$ and the worm's width profle; $R(u)$ is added in the normal direction to create the complete model $H(p)$. $N_u = 25$ and $N_v = 3$ define the discretization of the continuous surface.

means to constrain the worm's length. The model can allow for elongations and contractions via a *rate of length* function, $K(u)$: $\vec{x}(u) = \int_0^u K(\hat{u})\mathbf{T}(\Theta(\hat{u}))d\hat{u}$. In addition, the integral in (3.15) must be calculated efficiently because evaluation of the model location will occur many times within any iterative algorithm. I use Romberg integration [81] to achieve over an order of magnitude decrease in computation time over the standard Simpson quadrature.

Another subtle feature about this model is that the global orientation of the worm is encoded into the shape parameters α_j. This is a non-intuitive representation because typically, one would imagine the shape of the worm to be invariant with respect to a body fixed frame. In this way, the generative model would be defined as

$$H(p) = \beta\Big(\mathrm{R}(\theta) \underbrace{\int_0^u \begin{bmatrix}\cos(\Theta(\hat{u}))\\ \sin(\Theta(\hat{u}))\end{bmatrix} d\hat{u} + v\, R(u)\mathbf{N}(u)}_{\text{worm shape}}\Big) + \vec{T} \tag{3.16}$$

where $\mathrm{R}(\theta) = \begin{pmatrix}\cos(\theta) & -\sin(\theta)\\ \sin(\theta) & \cos(\theta)\end{pmatrix}$, $\left(\mathrm{R}(\theta), \vec{T}\right)$ defines a rigid body transformation in $\mathbb{R}^2$, and the new state is given by $p = \begin{bmatrix}\alpha_j & \theta & \vec{T}\end{bmatrix}^T$. However, this is a poor parameterization for estimation because of the *redundancy* between the parameters α_j and θ. This redundancy causes a many to one mapping between the state and the organism's pose and will likely result in poor estimates of θ. Therefore, my choice to directly model the organism's bend angle function in a global coordinate system provides a

compact parameter space for estimation and eliminates the need for a separate rigid body rotation. An invariant representation, α_j, of the worm's shape can be constructed by directly modeling the curvature function as a B-spline

$$\kappa(u) = \sum_{j=1}^{N_\kappa} \alpha_j \Phi_j^k(u). \tag{3.17}$$

However, this function would have to be integrated twice to construct the centerline, resulting in greater computation.

The choice of $k = 2$ for the order of the B-spline basis implies that the worm's centerline consists of piecewise continuous circular arcs with local discontinuities in the curvature (in Figure 3.1, $\Theta(u)$ is piecewise linear, thus the curvature is discontinuous). This design parameter was chosen to behave as a psuedo-constraint, by forcing the model to have constant curvature over each subregion of the centerline. Originally, I had chosen $k = 4$ such that the curvature basis functions were parabolas, but I found that the unconstrained Kalman filter estimation would drive the worm model into anatomically infeasible local minima because the endpoints of the worm model could take on arbitrary curvature values. The discontinuities in curvature of the model's centerline did not adversely affect tracking, rather they improved it. To account for the fact that the actual worm has continuous curvature, I would post-process the estimates to smooth out these discontinuities for any kinematic measurement of the worm. However, given the techniques for incorporating nonlinear constraints into the Kalman filter framework of Section 2.4, I can now use a model with $k = 4$ and incorporate constraints based on the anatomy of the worm so that misleading visual cues do not cause the algorithm to solve for an infeasible configuration.

3.3 The Fish Model

When fish, such as the zebrafish *Danio rerio* studied in this thesis, swim in shallow water, their motions are largely planar. Thus, a planar Frenet ribbon that is nearly identical to the worm model described in the previous section can be used to model the visual geometry of the swimming

creature. Unlike the worm model that is able to bend over the entire length of its body, the fish model assumes that fish have a stiff head. This assumption prevented the tracker from creating unrealistic bending deformations in the head region, and is based on the experimental results of Müller [75], whose data shows that the head region of freely swimming zebrafish undergoes negligible bending (zero local curvature). This curvature assumption corresponds to the bend angle function, $\Theta(u)$, remaining constant in the head region, which I define as the anterior 20% of the fish's body length, although other choices are possible. This constraint on curvature in the head region was implemented by defining the origin of u to occur at a distance of $0.2L$ behind the snout, such that the head region is described by positive u values ($0 \leq u \leq 0.2L$) and the tail region by negative u values ($-0.8L \leq u \leq 0$) (Figure 3.2B). This approach simplifies the formulation of the bend angle function, $\Theta(u)$:

$$\Theta(u) = \begin{cases} \sum_{j=1}^{N_\Theta} \alpha_j \Phi_j^k(u) & u \leq 0 \\ \sum_{j=1}^{N_\Theta} \alpha_j \Phi_j^k(0) & u > 0. \end{cases} \tag{3.18}$$

Figure 3.2 illustrates these concepts — the definition of u as well as how the B-spline basis functions describe the body wave. The spline bases have local maxima in the tail region, but become constant in the head region.

The complete fish model is then calculated according to Equation (3.15) using (3.18) as the bend angle function. The parameter domain is redefined as $u \in [-(1-\gamma)L,\ \gamma L],\ N_u = 30$, with $\gamma = 0.2$ and $v \in [-1,\ 1],\ N_v = 3$. The width function $R(u)$ is defined identically as a fourth-order open B-spline function using 20 basis functions, and its value is calculated from a chosen frame of the video recording (see Section 4.3) and held fixed during tracking. This process is illustrated by Figure 3.3. I denote the complete fish model as $H(p)$ where $p = \begin{bmatrix} \alpha_j & \vec{T} \end{bmatrix}^T$ are the fish parameters which include the bend angle amplitudes $\vec{\alpha}$ defined earlier and, $\vec{T}$, the global translation vector of the entire fish.

Creating deformable models based on medial profiles has been used in segmentation problems in medical imaging [47] and for tracking multiple *C. elegans* from microscopy images [37, 87]. Because zebrafish are laterally symmetric about their body axis, the medial profile representation offers

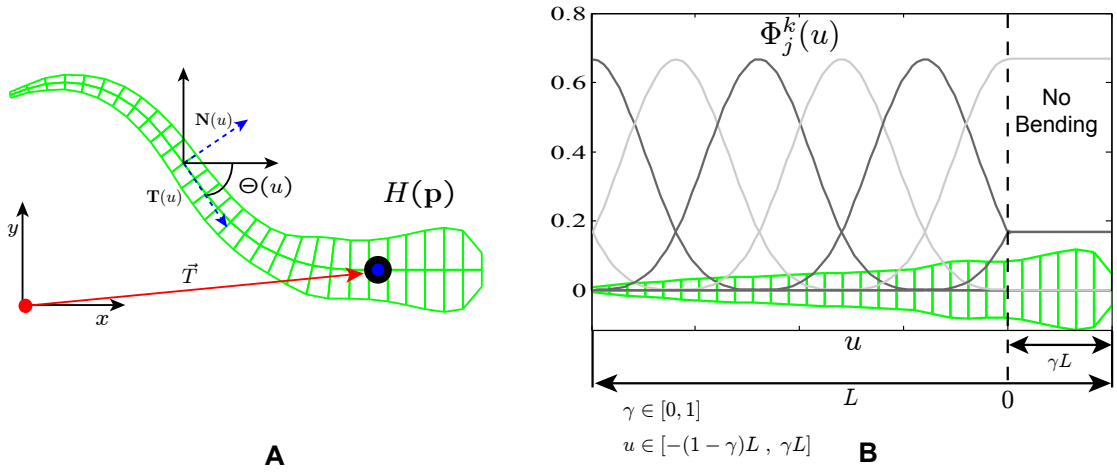


Figure 3.2: Illustration of the modeling approach used for zebrafish: (A) Geometric mesh $H(p)$ (green) with local tangent ($\mathbf{T}$) and normal ($\mathbf{N}$) vectors used to construct the mesh. The parameter $\vec{T}$ and the function $\Theta(u)$ define the position and shape of the model and are estimated during tracking; (B) Head region of length γL is designated as rigid (I set $\gamma = 0.2$ for all experiments), while tail region bends according to linear combination of eight B-spline bases $\Phi_j^k(u)$.

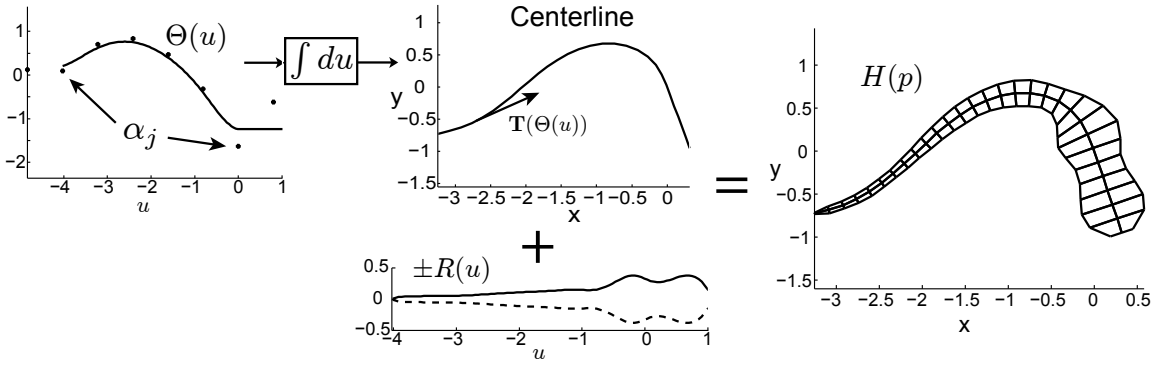


Figure 3.3: Method for constructing zebrafish model used in this analysis. The tangent vector associated with the function $\Theta(u)$ is integrated to create the fish centerline. This centerline is combined with the fish's width profile $R(u)$ to create the complete model $H(p)$. $R(u)$ remains fixed during tracking, and the bending of the model is modulated by changing the values of α_j, which control the shape of $\Theta(u)$.

several advantages for the tracking framework. Each B-spline basis function is defined only over a subregion of the fish body. Therefore, the local bend amplitudes, α_j, have local control over the degree of bending in the body. This property is analogous to the fish anatomy, where contractions of individual muscles affect the bending over subregions of the fish's body. In summary, this parameterization of the centerline has few degrees of freedom, requires no training data, and offers a natural and anatomically sound way to constrain the fish's length and designate certain regions as stiff.

3.4 The Fly Model

The model of the fruit fly consists of three different body parts that are constructed using parametric surfaces: the thorax/abdomen (which I henceforth refer to as "body"), head, and wing. The body of the fly is a Frenet tube with an elliptical cross-section that is constructed according to:

$$H(u,v) = C(u) + R(u)\Big(1.2\cos(v)\mathbf{B}(u) + \sin(v)\mathbf{N}(u)\Big), \qquad u \in [-\tfrac{L_b}{2}, \tfrac{L_b}{2}],\ v \in [0, 2\pi]. \tag{3.19}$$

The centerline $C(u)$ is modeled as a 3D B-spline curve with 5 control points, and the width profile $R(u)$ is defined as a fourth order open B-spline using 20 control points. The fly is assumed to have a cross section that is 20% wider in the lateral direction than the dorsal/ventral direction and length equal to L_b. The construction of this parametric surface is illustrated in Figure 3.4. Although the illustration here has a curved centerline, in the actual experiments I assume that the centerline $C(u)$ is a straight line aligned with the x- axis (see Figure 3.7). This simplifying assumption provided accurate model fitting for the image resolution of the video. Although the model is able to capture more detailed body deformations according to 3.19, it was not necessary for the experiments carried out in Chapter 6. The head of the fly is constructed identically to the body, except a different centerline and width profile are used (Figure 3.5). The head model is placed at the anterior portion

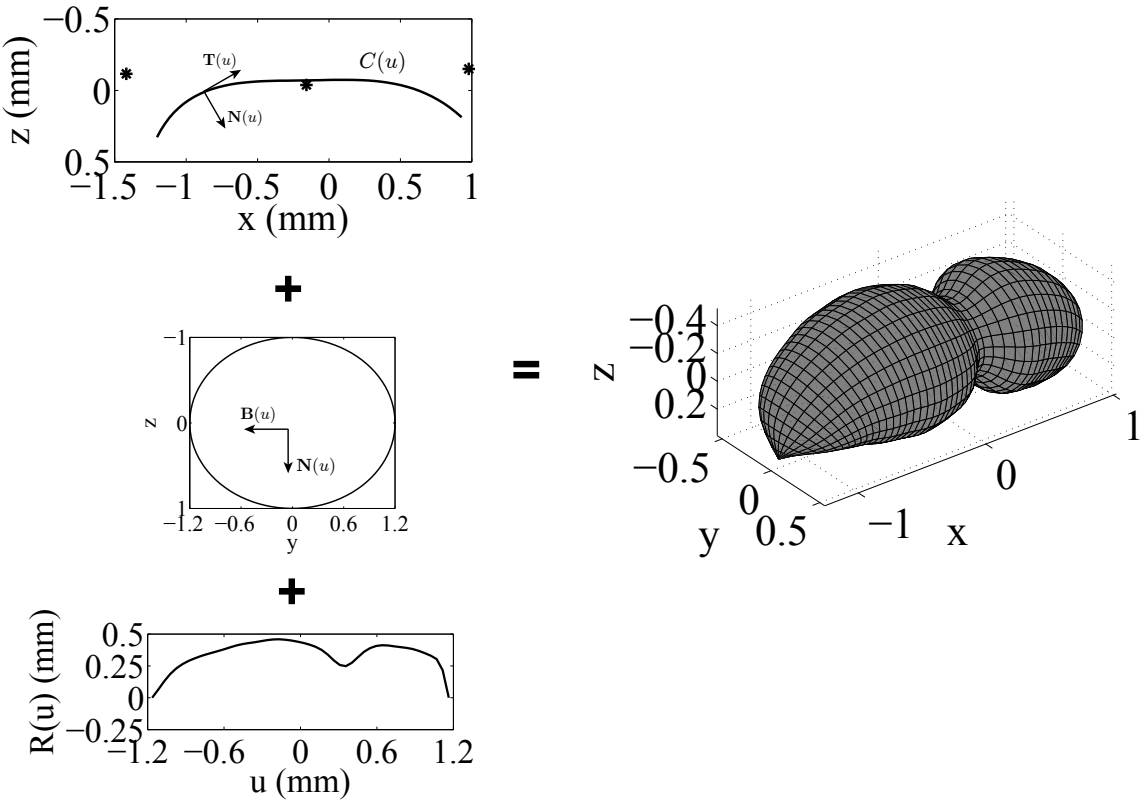
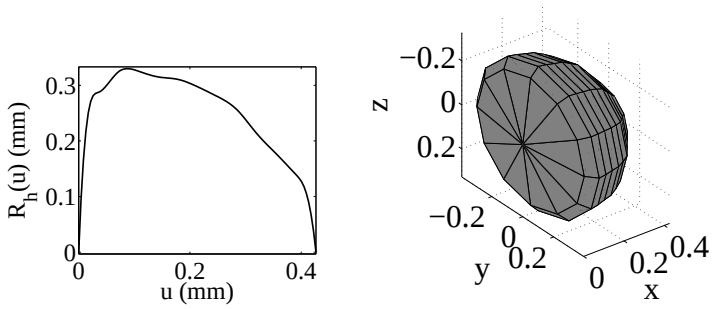


Figure 3.4: Method for constructing the body (i.e., thorax/abdomen) of the fruit fly model used in this analysis. The centerline $C(u)$ is a 3D B-spline curve with 5 control points (only 3 of them are visible in the axes). The curve of the centerline lies completely in the x-z plane. The width profile, $R(u)$, is revolved around $C(u)$ using a elliptical cross section where the lateral direction is 20% wider than the dorsal/ventral direction.

of the body model using the local Frenet frame of the abdomen's centerline:

$$X'_{head} = \begin{bmatrix} \mathrm{R}_{bh} & C(\frac{L_b}{2}) \\ 0_{3\times 1} & 1 \end{bmatrix} X_{head} \tag{3.20}$$

where X'_{head} and X_{head} are points on the head model with respect to the body and local fixed frame, respectively in homogeneous coordinates. The rotation matrix is given by $\mathrm{R}_{bh} = \begin{bmatrix} \mathbf{T}(\frac{L_b}{2}) & -\mathbf{B}(\frac{L_b}{2}) & \mathbf{N}(\frac{L_b}{2}) \end{bmatrix}$ (i.e., the local Frenet frame at the end of the centerline curve). Additional parameters can be introduced to model the relative motion between the head and the body, but for this study, I assume that the head remains fixed at the end of the body.



(a) (b)

Figure 3.5: (a) Width profile of the fly head model. (b) Complete head model of the fly constructed identically to Figure 3.4, except (a) is used as the profile curve and the centerline is just the x-axis

The wing of the fly is assumed to have negligible thickness, thus is modeled as a planar surface. The outline of the wing, $\gamma(u)$ is constructed from a closed fourth order B-spline curve with 20 control points. The origin of the closed curve is located at its centroid, so the parametric surface is constructed by scaling the curve down to its origin

$$H(u,v) = v\,\gamma(u) + t_w \qquad u \in [0,1],\ v \in [1,0]. \tag{3.21}$$

The origin of the wing's reference coordinate system is moved from its centroid to the joint location at t_w because this is a more intuitive location for the origin when constructing a kinematic chain (Section 2.7). An illustration of the model is provided in Figure 3.6. The wings are transformed

into the body fixed frame according to

$$X'_{wing} = \begin{bmatrix} \mathbb{I} & T_{bw} \\ 0_{3\times 1} & 1 \end{bmatrix} X_{wing} \tag{3.22}$$

where X'_{wing} and X_{wing} are points in the wing model with respect to the body and local fixed frame, respectively and $\mathbb{I}$ is the identity matrix. The translation T_{bw} from the body fixed frame to wing fixed frame is given by

$$T_{bw} = \begin{bmatrix} 0.2021 \\ \pm 0.1055 \\ -0.1477 \end{bmatrix} L_{bh} \tag{3.23}$$

where $\pm$ denotes the right and left wings, respectively, and $L_{bh} = L_b + L_h$ is the combined length of the body and head models. The complete fly geometric model is shown in Figure 3.7 with all of the parts fully assembled in the reference configuration. The parameterization of the model's pose is discussed in Section 6.3.

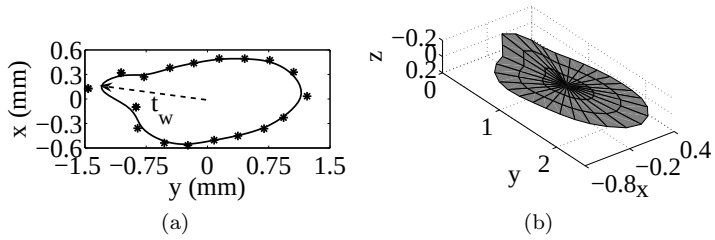


Figure 3.6: (a) Outline profile of the fly wing model constructed from from a closed planar B-spline curve with 20 control points. (b) Complete wing model of the fly constructed by scaling and translating the profile curve in (a), with $N_u = 30$ and $N_v = 4$

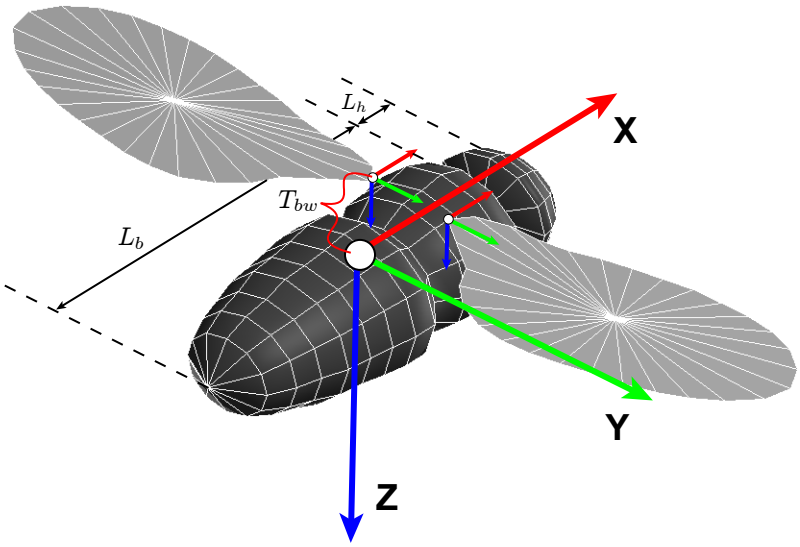


Figure 3.7: Geometric model of *Drosophila* used in our algorithm. L_b and L_h represent the length of the fly's body and head, respectively. T_{bw} is the translation vector that transforms the body-centered reference frame to the wing joint reference frame. The coordinate frame orientation follows the convention common to aeronautics where rotations about the x, y, and z axes are known as *roll*, *pitch*, and *yaw*, respectively. Downward pointing z axis is chosen so that positive pitch angles correspond to pitching upwards.

Chapter 4

Automated Visual Tracking of Zebrafish Swimming

The results of this chapter were accomplished in collaboration with the David Lentink, Sander Kranenbarg, Ulrike Müller, and Johan L. van Leeuwen of the Experimental Zoology Group of Wageningen University. Ansa Wasim recorded the movie sequences that were used in my analysis (Figure 4.7), and Henk Schipper captured the photographs used in the center of volume calculations (Figure 4.13d). B. Walderich, H. M. Maischein and J. Odenthal of the Hubrecht laboratory permitted use of the *stocksteif* mutant for demonstrating the ability of the tracker system. The *stocksteif* mutation in zebrafish is characterized by an overossification of the notochord. The axial skeleton of these mutants is a stiff, bony rod, contrasting the flexible series of articulating vertebrae in their wildtype siblings. In order to analyze the influence of vertebra development on the swimming capabilities of zebrafish, we quantified changes in swimming performance during all stages of development.

4.1 Introduction

The zebrafish is a key genetic model organism that has long served as a convenient model to study the various aspects of fish swimming [41, 103]. Automated tracking and analysis systems have been previously developed for zebrafish [5, 9]. However, these systems track the fish only as a point and cannot quantify body wave kinematics of swimming. Other studies of zebrafish swimming have manually tracked the fish to quantify the body posture, a method that is both time-consuming and

potentially prone to subjective errors [14, 69, 75]. Tytell and Lauder used a semi-automated method to estimate the fish midline by manually indentifying the snout and tail and automatically estimating the midline from the extracted silhouette [107]. Other authors have relied on "skeletonizing" algorithms that dissolve a binary image representing the animal's silhouette down to its midline [23, 43, 70]. Although this approach is automated, it will not estimate the correct midline if other objects are present in the binary image because it cannot distinguish between pixels that belong to the animal's silhouette and those that belong to a different object. As a result, these algorithms will not correctly estimate the fish's body posture when there is environmental clutter such as other fish, plants, or a hair used to initiate behavioral responses, as was done in the experiments of Section 4.5. Automated kinematic analysis of multiple zebrafish larvae was recently demonstrated. However, this particular analysis technique utilizes an image filter that is customized for the appearance of zebrafish larvae of a specific age [15]. This technique does not extend nicely for zebrafish of different ages, other fish species, or when environmental clutter is present.

Here, we present a complete method for accurately and efficiently quantifying the body posture of zebrafish and other organisms with symmetric medial profiles. This approach directly models the shape of the animal and utilizes locations of high contrast in the image to estimate its posture. The posture estimate is calculated using techniques that remain robust to clutter. The detailed swimming motion is estimated based on dorsal images of the fish recorded at sufficiently high frame rate [48], which enables a quantitative evaluation of the animal's kinematics and dynamics (provided the mass distribution of the animal is known). The next section develops a detailed geometric model for the zebrafish. Subsequently, I describe the appropriate motion and measurement models that use information from the previous and current frame to estimate the fish's current position and posture. Finally, the capabilities of the tracking approach are demonstrated on zebrafish performing an escape response at three stages during their development (from larvae to juvenile).

4.2 Motion Model

The motion model of the fish predicts the fish parameters at the current time step based on the parameters calculated at the previous time step. This is a computational approach to implementing $f(p, \xi)$, the dynamic motion model of the state space framework (Section 2.1). By predicting the motion, it provides a better initial estimate of the fish parameters before they are updated using the measurement model. It is assumed that the fish's movement is a combination of undulatory motion along its body and a displacement of the whole body in the direction of the centerline axis tangent vector at the head location (see Figure 4.1). The undulatory motion of steady swimming in fish consists of a traveling wave of increasing amplitude from head to tail, while the fast "C-start" behavior of fish resembles a standing wave.

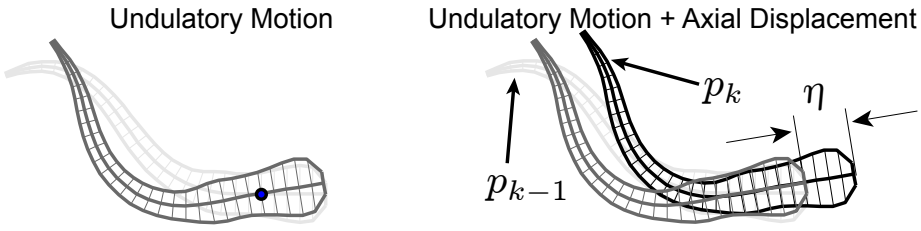


Figure 4.1: Illustration of motion model of the fish. We assume that the total motion between frames $k-1$ and k can be decomposed into undulatory motion and axial displacement. Note that figure displacements are exaggerated for illustration purposes. Actual motion between frames is much smaller due to high frame rate of camera.

Given the model of the zebrafish geometry presented in Section 3.3, these motions are expressed by the change in local bend amplitudes, $\vec{\alpha}$, from the previous time steps to the current one. To predict the time evolution of $\vec{\alpha}$, I utilize the method of matching scaled motion patterns described in Section 2.7. However, this technique requires that the motion parameters are invariant with respect to the global orientation of the organism (e.g., the knee joint angle of a human running is invariant with respect to that person's global running direction). $\vec{\alpha}$ does not currently satisfy this requirement because it encodes the global orientation of the fish. To rectify this, I construct a spatially invariant representation, $\bar{\alpha}$, that assumes the solution in the previous time step was a fish with its head aligned with the positive x-axis (see Figures 4.2 and 4.3). Let $\Lambda(u) \in \mathbb{R}^{N_u \times N_\Theta}$ be a matrix of B-spline bases evaluated at N_u sampled grid points in u. The invariant shape parameters are calculated by first

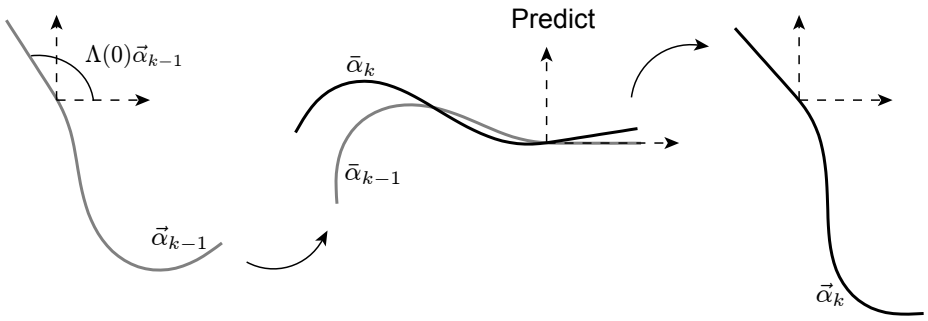


Figure 4.2: (Left) The shape parameters $\vec{\alpha}$ encode the global orientation of the fish. (Right) In order to create a shape representation that is invariant with respect to the global orientation, I assume that the solution from the previous time step is a fish with its head aligned with the positive x-axis (i.e., the bend angle is zero). The prediction is performed in the invariant representation $\bar{\alpha}$ according to the method described in Section 2.7, and then transformed back.

subtracting the head bend angle at the previous time step from the bend angle function at the current time step, and then projecting this new bend angle function back onto the B-spline bases:

$$\bar{\alpha}_k = \Lambda^{-1}(u)\Big(\underbrace{\Lambda(u)\vec{\alpha}_k}_{\text{bend angle function at time } k} - \underbrace{\Lambda(0)\vec{\alpha}_{k-1}}_{\text{head bend angle at time } k-1} \Big). \tag{4.1}$$

The matching of the motion pattern with the prior training set is done with respect to $\bar{\alpha}$ by calculating the optimal scale and location in the prior training set (Section 2.7, Figure 4.3a) according to

$$\underset{s,j}{\operatorname{argmin}} \sum_{v=0}^{m-1} \Big(||\bar{\alpha}_{k-v} - \widetilde{\alpha}^s_{j-v}|| \Big), \tag{4.2}$$

and the relative change in the shape parameters at the optimal location in the prior data is used to predict the shape parameters at the next time step

$$\bar{\alpha}_{k+1} = \bar{\alpha}_k + (\widetilde{\alpha}^s_{j+1} - \widetilde{\alpha}^s_j). \tag{4.3}$$

Finally, the predicted shape parameters are transformed back to the body's original orientation,

$$\vec{\alpha}_{k+1} = \Lambda^{-1}(u)\Big(\Lambda(u)\bar{\alpha}_{k+1} + \Lambda(0)\vec{\alpha}_{k-1} \Big). \tag{4.4}$$

The matrix inverse, $\Lambda^{-1}(u) = \left(\Lambda^T(u)\Lambda(u)\right)^{-1}\Lambda^T(u)$ is chosen as the Moore-Penrose psuedo-inverse, which minimizes the norm in the bend angle functions. However, the residual errors introduced by these calculations is negligible.

To model the axial displacement, I assume that the fish has constant velocity, η_1, between frames that is corrupted by acceleration noise, η_2. Thus, the state vector and process noise vector for a single organism are:

$$p = \begin{bmatrix} \vec{\alpha} & \vec{T} & \eta_1 \end{bmatrix}^T \qquad \vec{\alpha} \in \mathbb{R}^{N_\Theta},\ \vec{T} \in \mathbb{R}^2,\ \eta_1 \in \mathbb{R} \qquad \xi = \begin{bmatrix} \Delta\vec{\alpha} & \Delta\vec{T} & \eta_2 \end{bmatrix}^T. \tag{4.5}$$

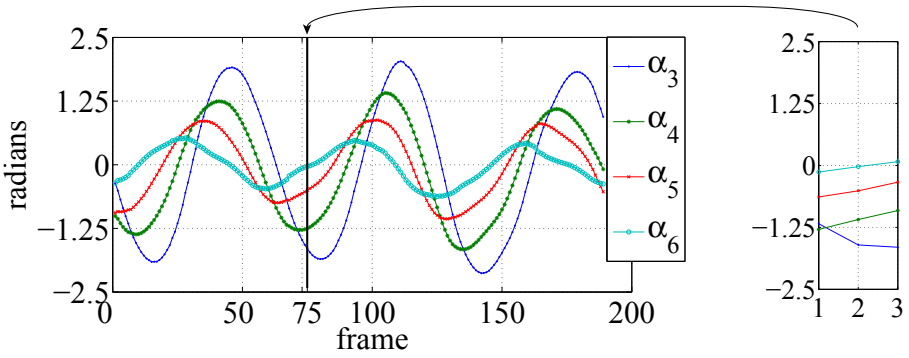
The complete motion model calculates the predicted state vector after the organism has undergone a total axial displacement of $\eta = \eta_1\Delta t + \eta_2\frac{\Delta t^2}{2}$ in the direction of the head's centerline tangent vector, where Δt is the inverse of the camera frame rate (Figure 4.1). These dynamic equations take the form:

$$\begin{aligned} p_{k+1} &= f(p_k, \xi_k) \\ \begin{bmatrix} \vec{\alpha}_{k+1} \\ \vec{T}_{k+1} \\ \eta_{1,k+1} \end{bmatrix} &= \begin{bmatrix} \Lambda^{-1}(u)\Big(\Lambda(u)\bar{\alpha}_{k+1} + \Lambda(0)\vec{\alpha}_{k-1}\Big) \\ \vec{T}_k + \int_0^\eta \begin{bmatrix} \cos(\Phi(\hat{u})\vec{\alpha}_{k+1}) \\ \sin(\Phi(\hat{u})\vec{\alpha}_{k+1}) \end{bmatrix} d\hat{u} \\ \eta_{1,k} \end{bmatrix} + \begin{bmatrix} \Delta\vec{\alpha}_k \\ \Delta\vec{T}_k \\ \eta_{2,k}\Delta t \end{bmatrix} \end{aligned} \tag{4.6}$$

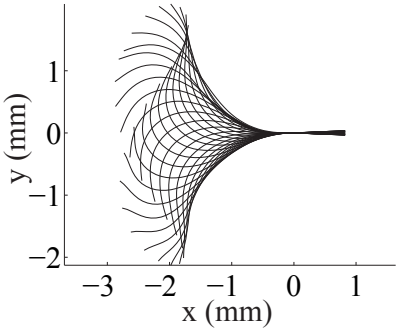
where $\Phi(u) \in \mathbb{R}^{1\times N_\Theta}$ is a row vector of B-spline bases. The predicted shape parameters are calculated using the method decribed above, and the predicted translation is calculated by integrating the tangent vector defined by the predicted shape parameters, $\begin{bmatrix} \cos(\Phi(u)\vec{\alpha}_{k+1}) \\ \sin(\Phi(u)\vec{\alpha}_{k+1}) \end{bmatrix}$, over the axial displacement, η.

4.3 Initial Detection of Zebrafish

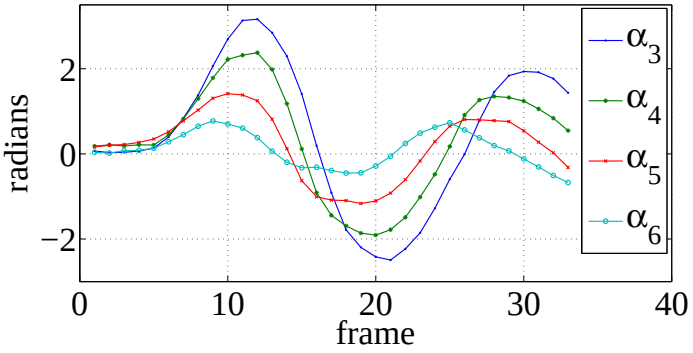
Any tracking algorithm relies on an initial estimate of the object location. To achieve this, I have developed a semi-automated initialization routine that operates on a chosen movie frame (typically



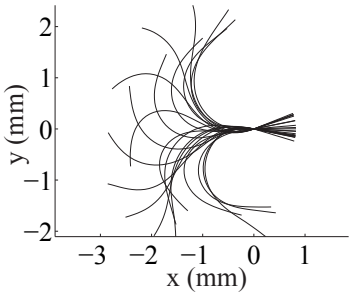
(a) Continuous swimming



(b) Fish centerlines calculated from (a)



(c) "C start" — startle response



(d) Fish centerlines calculated from (c)

Figure 4.3: (a) and (c) Spatially invariant representation of shape parameters, $\bar{\alpha}$, used to predict the undulatory motion (only 4 out of 8 parameters from the middle region of the fish body are shown). In (a), a query of shape parameters is matched with frame 75 in the prior database. The relative change in $\bar{\alpha}$ to frame 76 is used to calculate the predicted shape.

the first) and extracts three important pieces of information for tracking: 1) an estimate of the background image, 2) the initial fish state p_0, and 3) the width function, $R(u)$. The background image is estimated by selecting a region around the fish and erasing it using the built-in Matlab^{TM} function *roifill*, which smoothly interpolates inward from the pixel values on the boundary of the user-defined region. Next, the background image is used to segment the movie frame, and the Matlab^{TM} function *bwboundaries* calculates the fish boundary from the resulting binary image. The user is then requested to click on the snout and tail locations of the fish, which allows us to divide the boundary into the left and right discrete boundaries of the fish denoted $B_L(i)$ and $B_R(i)$, respectively.

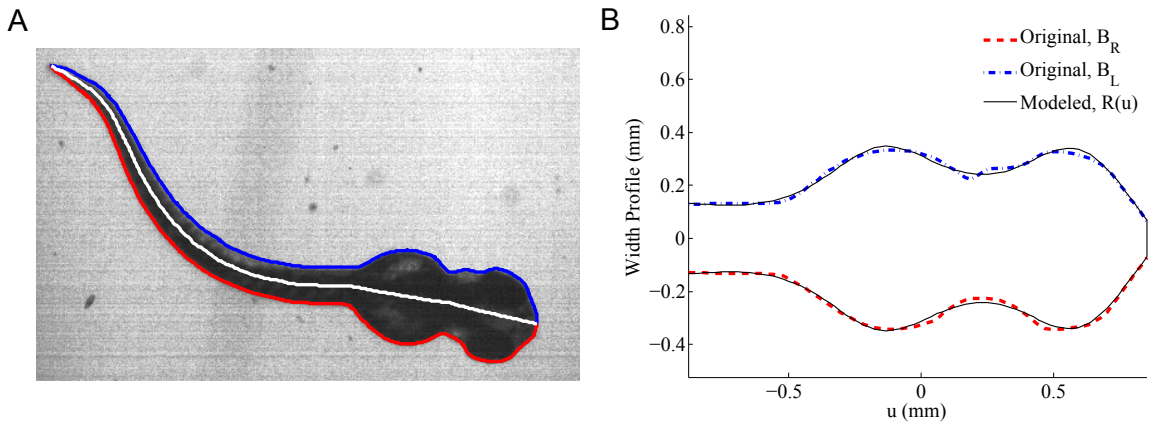


Figure 4.4: Illustration of the initialization process used in the fish tracker: (A) The initial fish centerline (white), $C(u)$, is estimated from the left (blue) and right (red) fish outlines. (B) This is used to estimate the width profile from the raw pixel data, B_R and B_L. The modeling approach assumes a symmetric fish. Figure is zoomed into the head region because $R(u)$ and pixel data are indistinguishable in the tail region.

The initial centerline of the fish will be the curve that is equidistant from the left and right boundaries. To determine this centerline, we use a modification of the integral area distance used in [87] which iterates and converges to the fish centerline. The method works by first finding the closest point in B_L to each point in B_R and vice versa. Then, I calculate the median locations between each set of corresponding points in B_L and B_R. These median locations are assigned as the new left and right boundaries and this process is repeated until the boundaries converge onto the true discrete centerline $C(i)$. This is described by the following psuedo-code:

while $\|B_L^j - B_R^j\| > \epsilon$

$$C_L(i) = \Big(B_L^j(i) + \lambda_{B_R^j(i)}\big(B_L^j(i)\big)\Big)/2$$

$$C_L(i) = \Big(B_L^j(i) + \lambda_{B_R^j(i)}\big(B_L^j(i)\big)\Big)/2$$

$$B_L^{j+1} = C_L$$

$$B_R^{j+1} = C_R$$

$$j = j + 1$$

end

The initial left and right boundaries B_L^0 and B_R^0 are calculated in the semi-automated fashion described earlier and λ is the nearest neighbor function, where $\lambda_A(B)$ is the element in A that is closest to B. The estimated centerline curve $C(u)$ is determined by fitting a B-spline through the points calculated from $C(i)$, and is illustrated in Figure 4.4 along with the original left and right boundaries it was derived from. The bend angle function $\Theta(u)$ is calculated from $C(u)$, and initial bending amplitudes are calculated by projecting the bend angle function onto the basis functions from (3.18) and illustrated in Figure 3.2. Once the centerline of the fish has been calculated, the optimal radius function is found by minimizing the squared normal displacement between the extracted image boundary $B_{L,R}$ and the model boundary $M_{L,R}$ dictated by the radius function while constraining the width profile to positive values. Let $R(u) = \Lambda_R(u)S$ where $S \in \mathbb{R}^{20\times 1}$ is a vector of control points and Λ_R is the matrix of B-spline bases, and $H(u_i, v_j) = C(u_i) + \Lambda_R(u_i)S\Big(v_j\,\mathbf{N}(u_i)\Big)$ is the Frenet ribbon of the fish model. I solve

$$\min_S \sum_{j=1}^{N_v}\sum_{i=1}^{N_u}\Big(\mathbf{N}^T(u_i)\Big(H(u_i,v_j) - \lambda_{B_L,B_R}(H(u_i,v_j))\Big)\Big)^2 \tag{4.7}$$

$$\text{subject to } \Lambda_R(u_i)S \geq 0 \tag{4.8}$$

where $N_v = 2$, so that only the points on the boundary of the model are chosen, while the number of samples chosen along the length of the fish is chosen large enough to accurately estimate S, (e.g., $N_u = 100$). The result of this minimization is shown in Figure 4.4 and demonstrates that the width profile of the fish can be accurately reconstructed.

4.4 Observation Model

The present tracker system assumes that the location of the fish boundary can be measured. Thus, the measurement model consists of the boundary points, $q_i = \binom{x_i}{y_i}$, along with their outward normal vectors, $n_i = \binom{n_i^x}{n_i^y}$ along the fish's outline. In order to fit the geometric model to the image at each frame, the appropriate image measurements must be matched to the corresponding locations in the model. To achieve this match, the image is segmented using background subtraction, which produces a binary image that is used to search for edges. Next, a one-dimensional edge-detector filter is applied in the direction normal to the boundary at each of the boundary points in the fish model (see [8] for details or the contour model in Figure 5.4 for an illustration). The distance between the model boundary point, q_i, and the corresponding detected edge point, r_i, is projected onto n_i so that the error minimized by the SPKF is the normal displacement between the edge points and model points

$$E(p) = \frac{1}{2}\|\mathbf{n}^T(\mathbf{r}-\mathbf{q})\|^2 \tag{4.9}$$

$$= \frac{1}{2}\Big[(\mathbf{n}^T\mathbf{r}-\mathbf{n}^T\mathbf{q})^{\mathbf{T}}\,(\mathbf{n}^T\mathbf{r}-\mathbf{n}^T\mathbf{q})\Big] \tag{4.10}$$

$$= \frac{1}{2}\Big[(z-h(p))^{\mathbf{T}}\,(z-h(p))\Big] \tag{4.11}$$

where the bold letters indicate the concatenation of all the sampled boundary points (e.g., $\mathbf{q} = \begin{bmatrix} q_1 \\ \vdots \\ q_i \end{bmatrix}$) and (4.11) is identical to the measurement term of the Kalman filter update equation discussed in Section 2.2, modulo a covariance matrix. This process is illustrated in Figure 4.5 where the initial estimate, edge points, and final solution are overlaid on an actual image. By using the SPKF, the error in (4.11), along with an additional term corresponding to the predicted estimate, are minimized to obtain an updated estimate of the fish's state. At age 28 days, the zebrafish has fully developed pectoral and caudal fins. These fins can cause incorrect tracking because the lighting conditions can make them appear as solid as the fish's body. However, by modifying the fish model to not take edge measurements in the pectoral and caudal regions, the body posture of the juvenile fish is still accurately estimated (Figure 4.6).

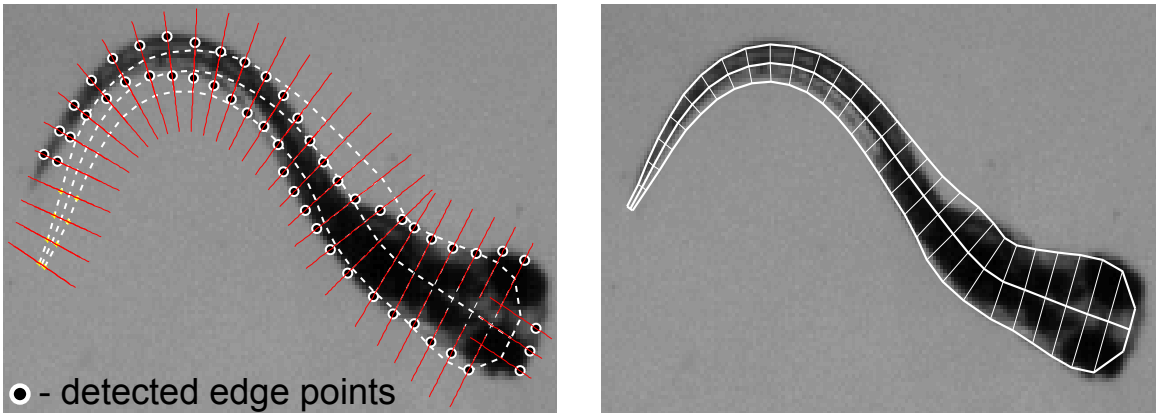


Figure 4.5: Measurement model for matching zebrafish images. Left: initial estimate of the model location (white-dashed line) with matching edge feature points, (black filled white circles). Red lines denote the 1D search regions for edge points. Note the tail is initially not matched to the boundary. Right: Final estimate of the model after 4 iterations. Although some error is present between the outline of the model and the actual fish, the centerline is accurately estimated based on visual inspection. Errors in the outline are due to small out of plane motions of the fish.

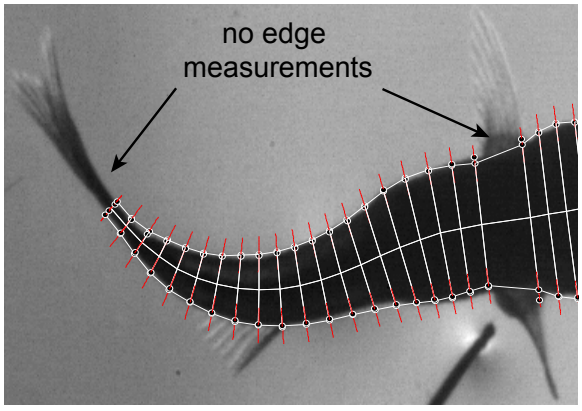


Figure 4.6: At age 28 days, the fish has fully developed pectoral and caudal fins, which can cause incorrect model fitting if they are mistakenly classified as part of the boundary. To address this, the juvenile fish model is modified to not incorporate edge measurements in the pectoral and caudal regions.

4.5 Tracking Results

Zebrafish (*Danio rerio*) eggs were collected after mating one *stocksteif* heterozygous female with one *stocksteif* heterozygous male. The batch of eggs contained both *stocksteif* mutant and wildtype embryos, but the mutant phenotype does not become apparent until five days post fertilization. The embryos were reared at the optimal rearing temperature of 28 °C. After hatching, the embryos were fed Paramecium (5 and 6 days post fertilization) and Artemia (from day 7 onwards). A fast startle response was recorded at 5, 15, and 28 days post fertilization for both wildtype and *stocksteif* mutant animals using a high-speed video camera (Photron, APX RS, 1500 frames/s, 1024×1024 pixels, exposure time 1/8000 s) fitted with a 105 mm Nikon lens. The startle responses were elicited by touching the animals with a horse hair. I analyzed recorded sequences from the initiation of the escape response to the moment when the fish either leaves the field of view or ceases active swimming. The sequences therefore include stage 1 and 2 of an escape response, and usually several tail beats that are part of stage 3 [115].

Here, I present automated tracking results for wildtype and *stocksteif* mutant zebrafish at 5, 15, and 28 days post fertilization. Figure 4.7 shows the raw centerlines of the zebrafish estimated by the tracker at fixed time intervals and demonstrates the quality of the proposed method. The method successfully tracks fast escape responses of fish larvae (Figure 4.7A,D) despite occasional partial occlusions (in this case, the hair used to induce the escape response (Figure 4.7E)). Furthermore, tracking takes an average computation time of $5.5 \pm 1.7 \frac{sec}{frame}$ on a 3.0GHz Intel®Xeon processor, which enables the quick analysis of large datasets.

To calculate an upper bound on the accuracy of the tracking approach, I created a synthetic movie sequence by rendering our model along a known trajectory. I then tracked this synthetic sequence and measured the average error between the known and estimated centerlines over time (Figure 4.8). The algorithm is able to localize the synthetic data within 0.5% of the body length on average; errors in estimating the real movie sequences will be slightly larger because they contain additional noise sources not present in the synthetic one.

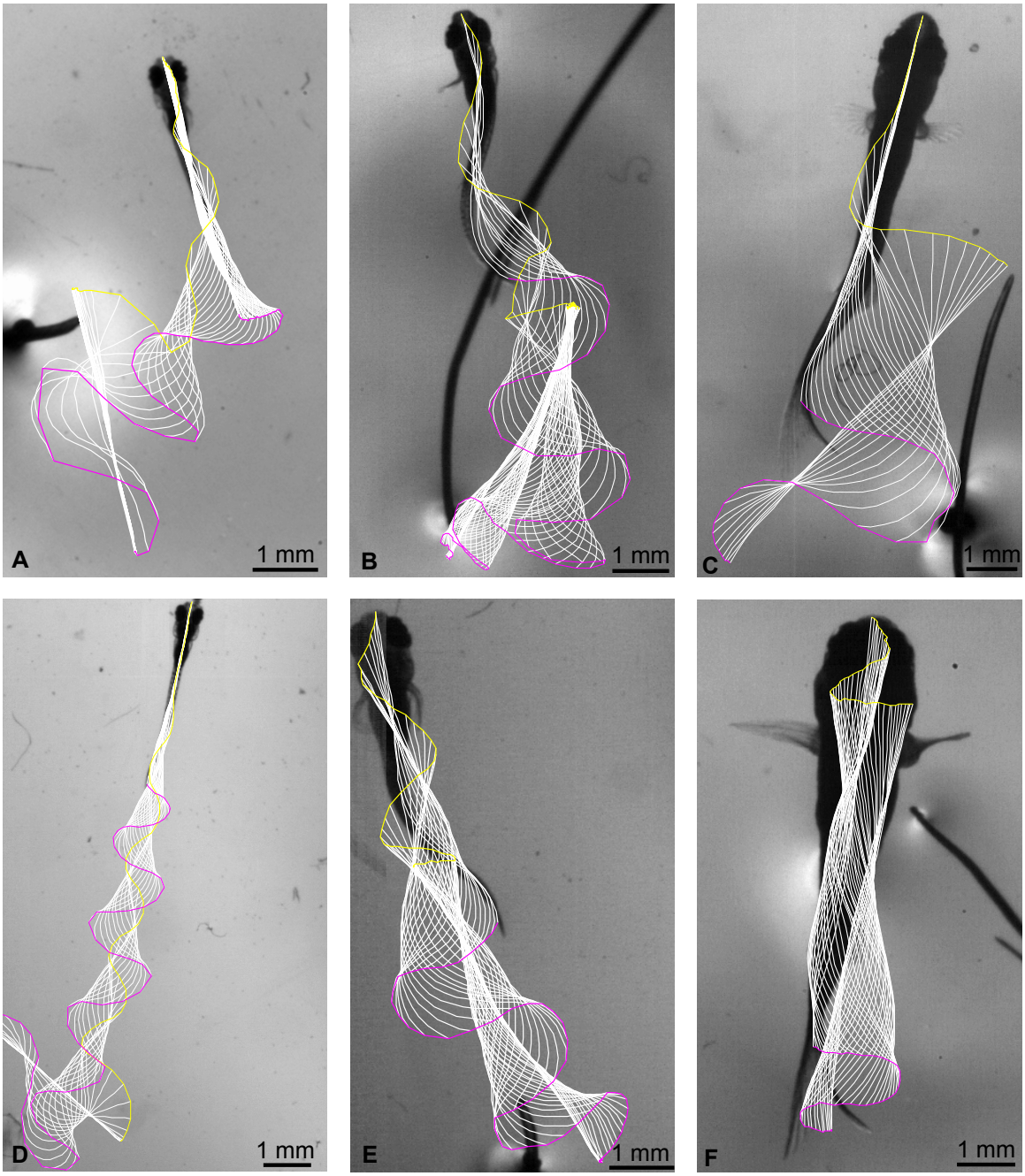


Figure 4.7: Tracking results for zebrafish at (A,D) 5 days, (B,E) 15 days, and (C,F) 28 days post fertilization. The first row are wildtype and the second row are *stocksteif* mutants. The raw centerlines estimated by the tracker are plotted at 1.3 ms intervals for age 5 and 15 days and 2.7 ms intervals for age 28 days. Magenta and yellow trajectories indicate the paths of the tail and snout, respectively. Note in (C,F) that the caudal fin is not modeled in our current approach, so its motion is disregarded.

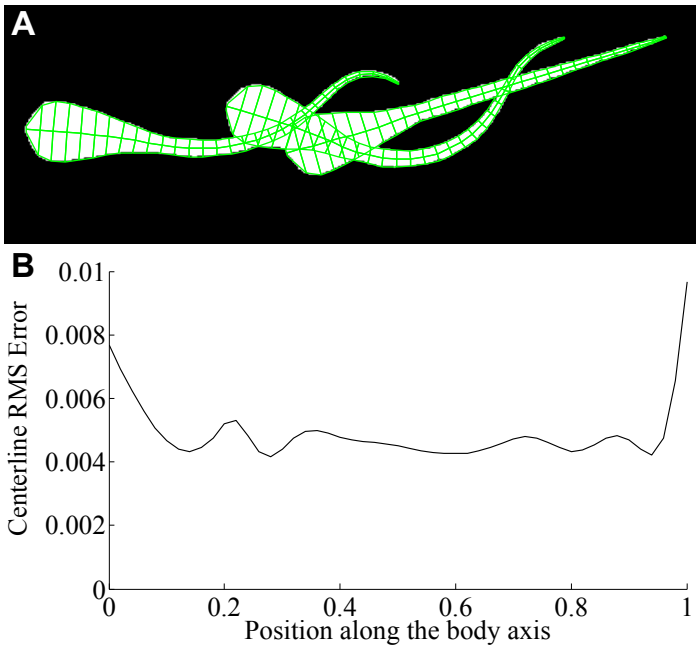


Figure 4.8: Error estimates from tracking synthetic images generated with our model (A). This provides an upper bound on the accuracy that we can achieve with the current implementation. (B) Given noiseless images, we can localize the centerline of the fish to within 0.5% of its body length on average. Actual errors on real data will be slightly larger than this.

4.6 Kinematic Data

From these centerlines, I wish to extract important kinematic parameters to gain insight into developmental influences on the propulsion mechanisms of swimming fish, and, vice versa, the mechanical influences on the development of the fish. One of these parameters is body axis curvature, which provides information about the muscle strains the fish undergoes. To measure curvature, I apply spatial smoothing to the extracted centerlines and then apply temporal smoothing directly to the curvature values. Spatial smoothing is performed by fitting a quartic (k=5) B-spline curve to the extracted centerlines (I constrain the fitted curve to have zero curvature at the endpoints). The fitted planar B-spline curve has the form $\gamma(u) = \Lambda(u)X$ and is calculated by solving

$$\min_X \sum_{i=1}^{N_u} \frac{1}{2} \left\| \Lambda(u_i)X - H(u_i, 0; p_k) \right\|^2 \tag{4.12}$$

$$\text{subject to } \kappa(u_i) = 0, \qquad i = 1, N_u \tag{4.13}$$

where $H(u_i, 0; p_k)$ are the centerline points from the estimated model at time k, $\Lambda(u) \in \mathbb{R}^{N_u \times m}$ denotes the matrix of B-spline bases evaluated at N_u sampled grid points in u, and $X \in \mathbb{R}^{m \times 2}$ denotes the matrix of control points. Similar to the width profile calculation of Section 4.3, the number of samples chosen along the length of the fish, N_u is chosen to be large enough ($N_u = 100$) such that X is accurately estimated. The curvature, $\kappa(u)$, is calculated directly from the B-spline bases. Let $\mathbf{T}(u)$ and $\mathbf{N}(u)$ denote the unit tangent and normal vectors to the curve $\gamma(u)$, then $\mathbf{T}'(u) = \kappa(u) \cdot \mathbf{N}(u)$ from the geometry of planar curves. Then the curvature is calculated as

$$\gamma'(u) = \Lambda'(u)X \tag{4.14}$$

$$\frac{\partial}{\partial u}\Big(\|\gamma'(u)\|\,\mathbf{T}(u)\Big) = \frac{\partial}{\partial u}\Big(\Lambda'(u)X\Big) \tag{4.15}$$

$$\|\gamma'(u)\|\,\kappa(u)\,\mathbf{N}(u) = \Lambda''(u)X \tag{4.16}$$

$$\kappa(u) = \frac{\Big\langle \Lambda''(u)X\,,\,\mathbf{N}(u)\Big\rangle}{\|\gamma'(u)\|}. \tag{4.17}$$

The fish's curvature is calculated from the smoothed centerlines instead of deriving it from the raw centerline of the geometric model because the model contains a discontinuity in the curvature at the location where it becomes stiff. The units of curvature are mm^{-1}, so I multiply the values by the body length of the fish to produce a non-dimensional value known as the *specific curvature*, used for all subsequent measurements.

Temporal smoothing is performed by applying a low-pass filter to the curvature values across time at fixed locations (51 uniformly spaced points) along the fish's body. The cutoff frequency for the filter is chosen based on visual inspection of the magnitude response of the curvature's Fourier transform at each of the body locations. Figure 4.9 plots the error between the filtered and unfiltered centerlines and curvature values for both wildtype and *stocksteif* zebrafish. With average errors around 0.1% of body length and 0.1 for the centerline and curvature, respectively, the filtering process does not compromise the accuracy achieved by the automated tracker. The resulting curvature profiles after filtering are shown in Figure 4.10. Comparing *stocksteif* with wild type at age 15 and 28 days (Fig. 4.10B,E, Fig. 4.10C,F) it can be seen that the peak curvature

values of *stocksteif* are smaller than those of wildtype.

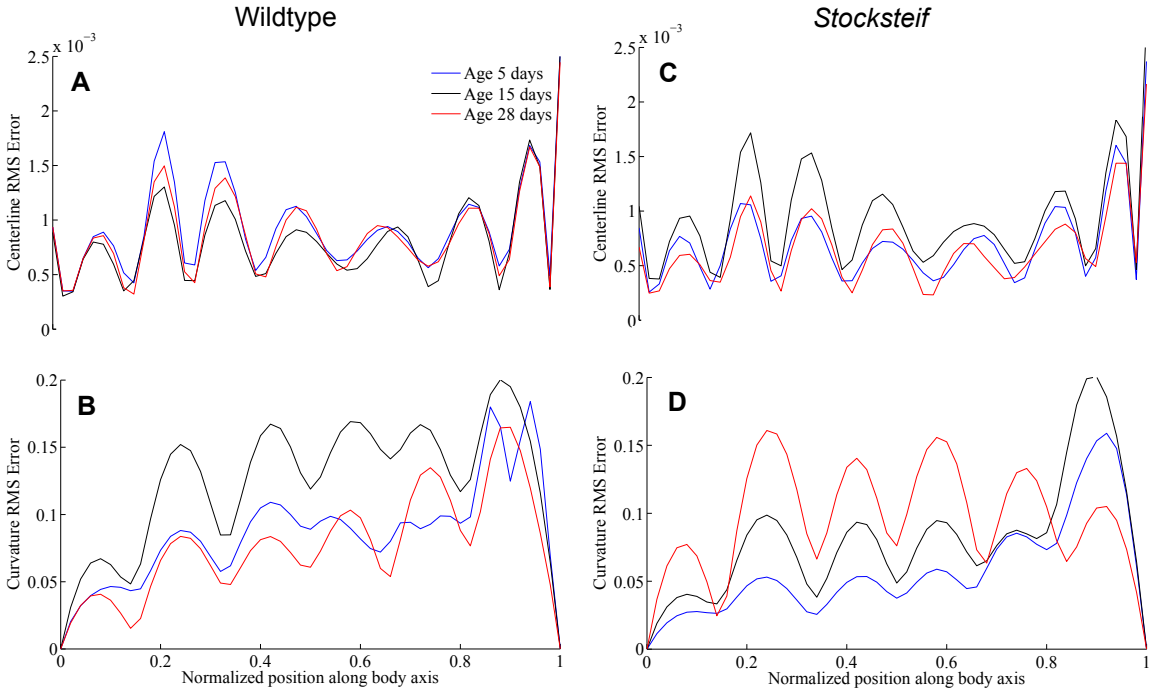


Figure 4.9: Average error between the filtered and unfiltered data as a function of body axis position. The error for the centerline location (A,C) and curvature (B,D) are normalized to body length and measured at 51 uniformly spaced locations along the fish body. This provides an average deviation over time between the filtered and unfiltered data at particular locations along the fish. Small error values are achieved for both wildtype and *stocksteif* fish, illustrating that the post-processing filtering technique retains most of the original information.

To analyze the performance of the fish, additional kinematic parameters were measured. The angular acceleration is the second temporal derivative of the fish bend angle function, (i.e., $\frac{\partial^2}{\partial t^2}\Theta(u,t)$). In Figure 4.11, similar peak angular accelerations are observed between wildtype and *stocksteif* at age 5 days, when the mutant phenotype has just become manifest. However, as the fish age, large discrepancies appear in the angular accelerations. At age 28 days, the peak angular accelerations of the wildtype are two orders of magnitude larger than that of the *stocksteif*. This trend is also present in the tail beat frequency of the fish. To develop an objective method for estimating the tail beat frequency of the fish, the Fourier transform of the curvature values during continuous swimming was calculated at equally spaced locations along the fish's body. The time period of continuous swimming was manually determined by inspecting the curvature profiles for regions where wavespeed remained relatively constant (see Fig. 4.10). Figure 4.12 plots the magnitude of the frequency response along

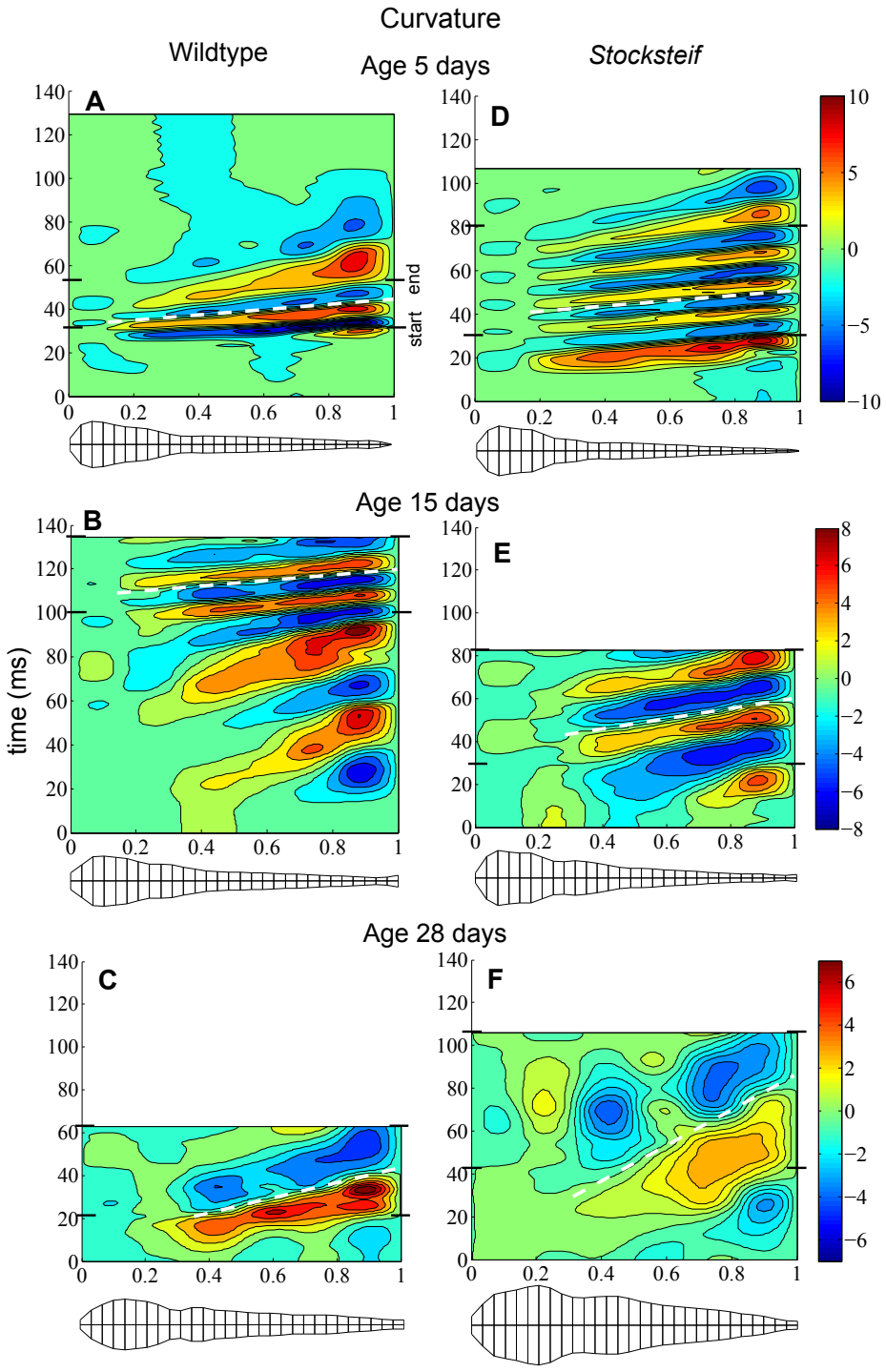


Figure 4.10: Specific curvature profiles for wildtype and mutant zebrafish at 5, 15, and 28 days post fertilization. Black tick marks indicate the regions of approximate continuous swimming that are used in the frequency analysis of Figure 4.12. Dotted white lines indicate approximate linear fit of zero curvature contour used to calculate wave speed.

the body axis of the fish. For all fish, the body axis position at approximately 90% posterior to the snout has the largest frequency response. The tail beat frequency, f, is calculated in the following manner. Let $\mathcal{F}[\cdot]$ denote the Fourier transform of a function, and $\kappa(u,t)$ the curvature as a function of position along body axis and time. Then $K_i(\omega) = \left|\mathcal{F}[\kappa(u,t)|_{u=u_i}]\right|$ is the magnitude of the Fourier transform of the curvature function at the i^{th} location along the fish's body, and $\omega^i_{max} = \underset{\omega}{\operatorname{argmax}}\, K_i(\omega)$ is the frequency that maximizes the magnitude response at the i^{th} location. The tail beat frequency, f, is calculated as

$$f = \frac{\sum_i^{N_u} K_i(\omega^i_{max}) \cdot \omega^i_{max}}{\sum_i^{N_u} K_i(\omega^i_{max})} \tag{4.18}$$

using N_u locations sampled along the fish's body. This is an average frequency over the length of the body that is weighted by the magnitude response. This approach does not rely on determining the tail's lateral displacement from a mean path of motion [75], and is therefore invariant to the spatial trajectory of the fish. Again, similar tail beat frequencies are observed at age 5 days. However, the 15 and 28 day old *stocksteif* have smaller tail beat frequencies than the wildtype. The curvature wave speed is estimated by performing a linear fit to the points of zero curvature during continuous swimming (see Fig. 4.10), and then calculate the resulting wavelength given the tail beat frequency provided by the Fourier analysis using (4.18). A summary of these values is provided in Table 4.1.

Typical measures of escape swimming performance include displacement, speed, and acceleration of the fish center of mass (COM) when stretched straight [33, 114]. To estimate the location of the COM, I reconstructed the fish volume from dorsal and lateral photographs assuming an elliptic cross-section [70] (see Figures 4.13a–4.13c). The center of volume (COV) and center of area from the dorsal view (COA) were calculated. The COV will correspond to the COM assuming the fish has uniform density. COV calculations were only performed on wildtype zebrafish for comparison with published literature on escape responses. However, COV calculations do not lend themselves to high-throughput analysis because separate photographs of each individual fish must be acquired to account for the variation in morphometry for fish at a given age (very large variation exists for

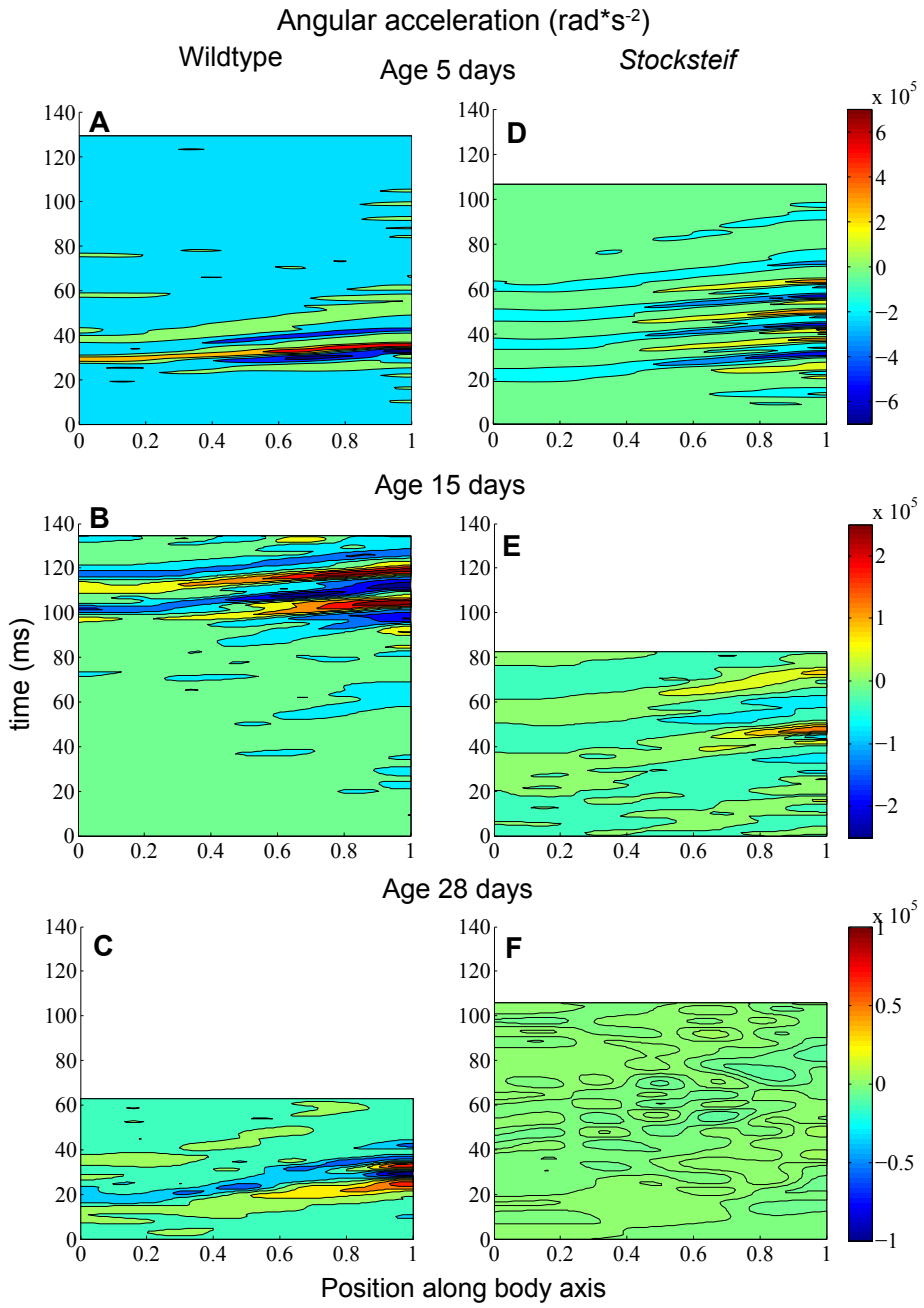


Figure 4.11: Angular acceleration of wildtype and *stocksteif* zebrafish at 5, 15, and 28 days post fertilization. The largest accelerations are present near the tail tip where the body's moment of inertia is smallest. The largest accelerations occur during the initial tail beats when the fish is starting from rest. There is a significant difference in magnitude between the wildtype and *stocksteif* accelerations at age 15 and 28 days, however, similar values are achieved at age 5 days.

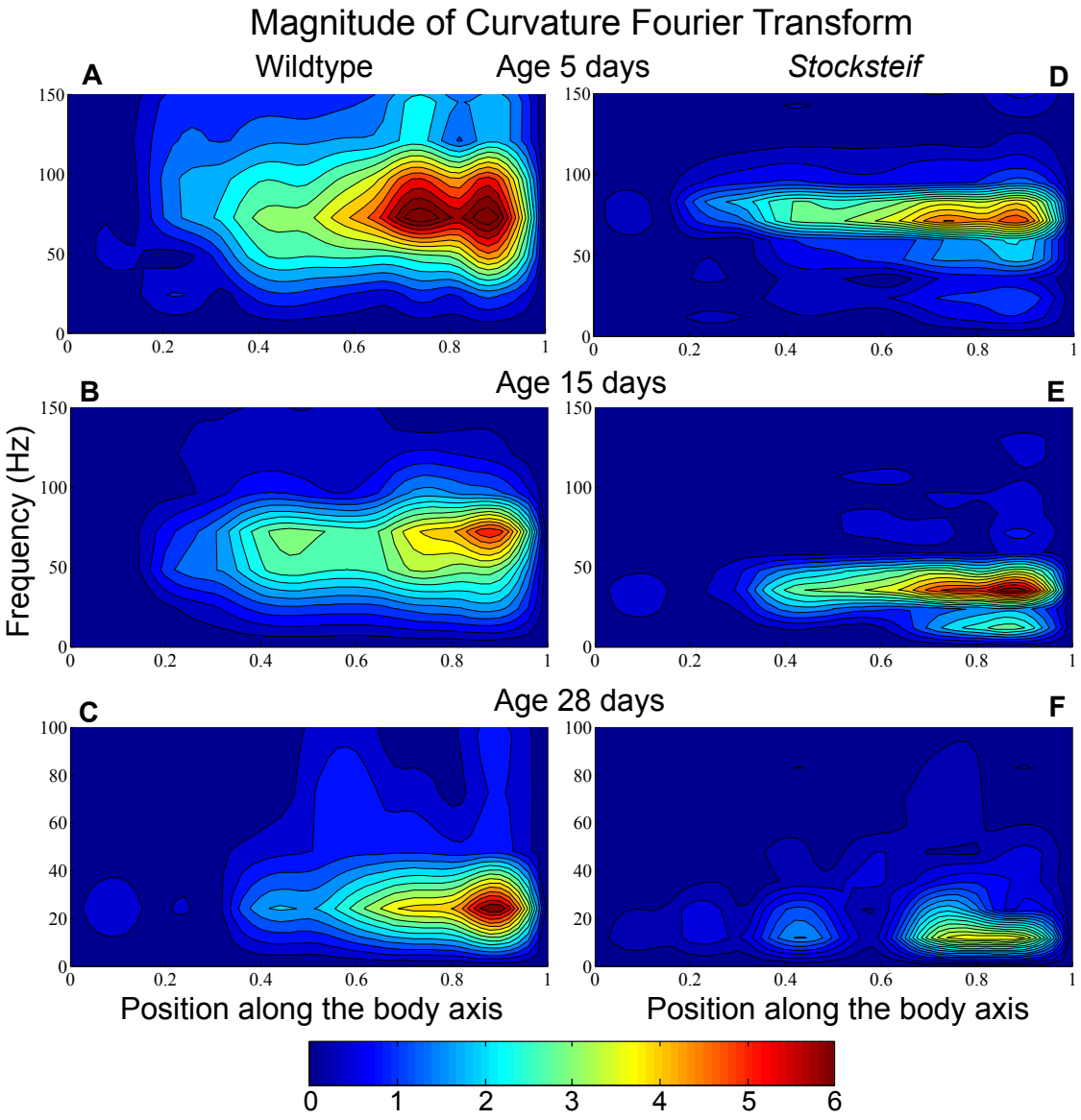


Figure 4.12: The magnitude of the curvature's Fourier transform during continuous swimming. The characteristic swimming frequency for each fish is calculated by taking a weighted average of the maximum frequency responses along the length of the fish. At age 5 days, the fish have similar swimming frequencies. However, at age 15 and 28 days, the *stocksteif* have slower swimming frequencies than the wildtype. In addition, the 28 days *stocksteif* primarily has undulations in the posterior 40% of its body due to its stiffer vertebrae.

Age (days)	L (mm)	f (s^{-1})	c (s^{-1}) (R^2, # of points)	λ
5 wt	3.4	73	115.0 (0.98, 51) 83.2 (0.98, 52) 55.2 (0.99, 60)	1.15
5 stkf	3.4	74	87.5 (0.99, 55) 87.5 (0.99, 55) 92.9 (0.98, 53)	1.21
15 wt	5.0	71	80.1 (0.99, 56) 79.3 (0.96, 57) 81.1 (0.98, 54)	1.13
15 stkf	5.4	36	42.9 (0.99, 59) 42.1 (0.98, 56)	1.18
28 wt	9.7	24	28.8 (0.98, 62) 24.6 (0.99, 58)	1.12
28 stkf	9.4	12	15.7 (0.95, 85) 13.6 (0.98, 91)	1.22

Table 4.1: Summary of kinematic parameters for wildtype (wt) and *stocksteif* (stkf) zebrafish at different ages. L, bodylength; f, swimming frequency; c, wave speed (R^2, number of points); λ, average wavelength. Each wave speed calculation represents a different linear fit performed in the region designated as continuous swimming in Figure 4.10.

the *stocksteif* at a specific age). Instead, I propose to use the COA as a location for comparison between wildtype and *stocksteif* because it is easily measured from the video sequence and has similar speed and acceleration profiles to the COV (see Figure 4.14). Figure 4.13d demonstrates that the COA location has a maximum deviation of 5–6% of bodylength from the COV at age 5 days and the deviation decreases as the fish ages, thus, COA serves as a good proxy for the COV. Figure 4.14 illustrates the results of these measurements. Speed and acceleration are calculated using the MSE spline method of Walker [114], which fits a quintic spline to the displacement data based on the expected error of a given camera resolution and frame rate. The *stocksteif* consistently exhibits lower peak acceleration and speed compared with wildtype at each age. These preliminary measurements indicate consistent discrepancies between the wildtype and *stocksteif* due to the stiffer vertebral column present in the mutant.

4.7 Discussion

I presented a method for estimating the body posture of zebrafish within laboratory environments using flexible geometric models and nonlinear estimation. Given the generalized mathematical frame-

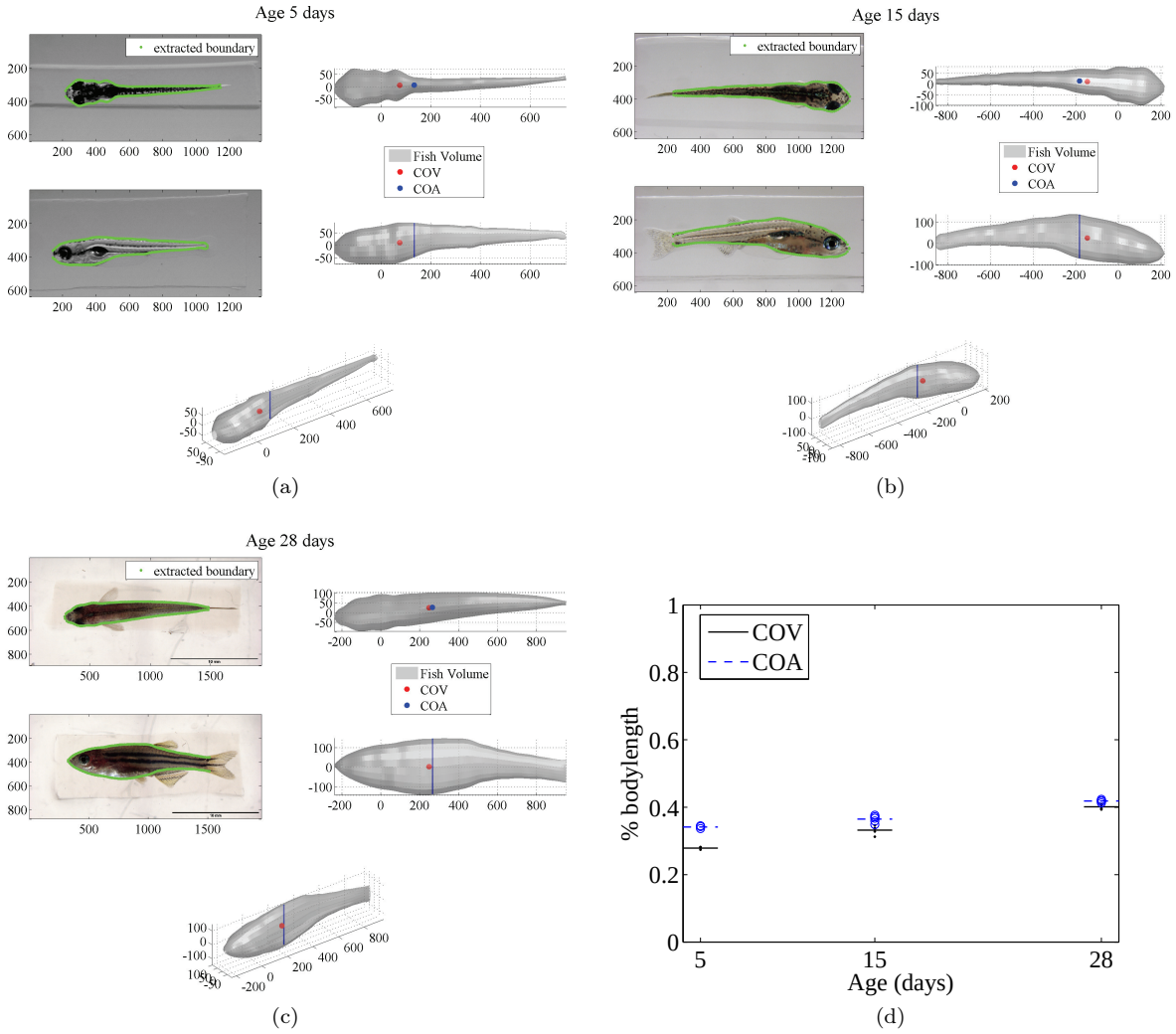


Figure 4.13: (a)–(c) Example of center of volume (COV) and center of area (COA) calculation for zebrafish at age 5, 15, and 28 days. Volume assumes an elliptical cross-section of the fish. COV will coincide with the COM if a uniform density is assumed. (d) Location of COV and COA measured posterior from the snout of the fish. For ages 5, 15, and 28 days, we measured data from 3, 5, and 4 fish, respectively. Horizontal bars indicate the mean value. As the fish get older, the COV and COA become closer to the same location. COA serves as a good proxy for COV and is easier to calculate in a high-throughput manner.

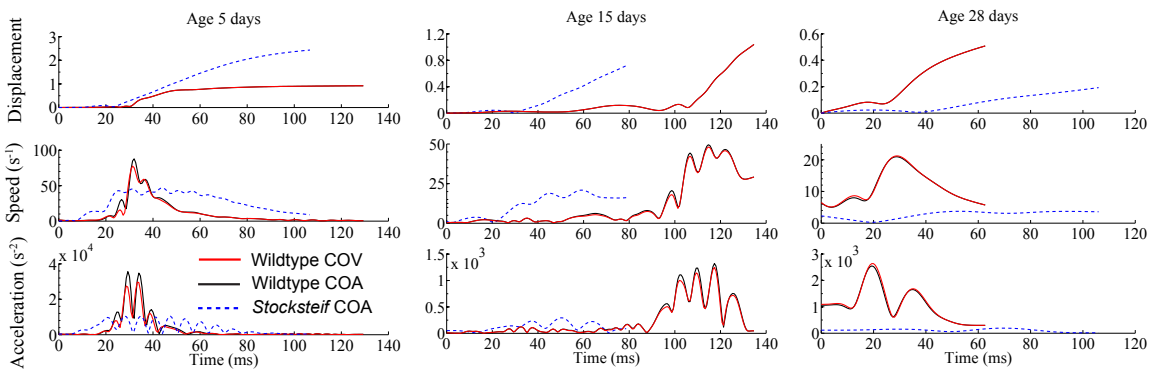


Figure 4.14: Displacement, speed, and acceleration plots for the fish measured at the center of area (COA) of the dorsal view. Zero time indicates the onset of stimulus, and the MSE quintic spline method of [114] is used to calculate speed and acceleration from the positions estimated by the model. Profiles measured at wildtype center of volume (COV) is determined using the method described by Figures 4.13a–4.13c

work used to model the fish's appearance, this method should track any fish species with a symmetric medial profile and that swims by undulating the body (see Figure 4.15 for an application to lampreys). The discrete time dynamic state space model also provides a general framework for performing statistical inference that is robust to outliers and enables tracking during partial occlusions. This is a strong improvement over previously developed methods which are either manual [14, 69, 75], require a perfectly segmented image with no environmental clutter [23, 43, 70], or are customized for fish of a specific size and appearance [15].

The assumption of planar motion is a limitation in the proposed method, which arises directly from the recorded material—top-view monocular video. For behaviors that contain large out of plane motions, the geometric model will not accurately represent the appearance of the fish. For such behaviors, a 3D version of the model-based tracker would be required; the principle is sound, but should be extended (see Chapter 6 for the extension to 3D tracking of *Drosophila*). The tracker may also fail if a significant portion of the fish becomes completely occluded by environmental clutter (e.g., the entire head). Nevertheless, this shortcoming could be improved by extending the algorithm to use a more advanced nonlinear estimator, a more advanced motion model, or a direct model of the occlusions. However, in laboratory settings, where many environmental parameters are controlled, this technique represents an accurate and fully automatic approach to quantify behavior and will

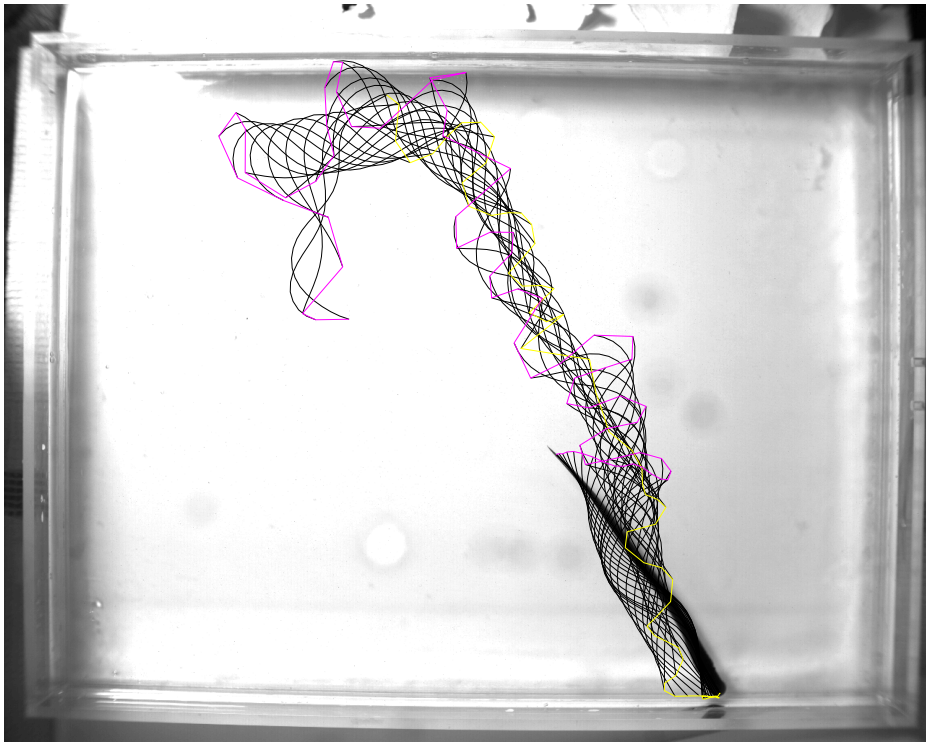


Figure 4.15: Tracking results for lamprey filmed at 25 fps. The raw centerlines estimated by the tracker are plotted at 120 ms intervals. Magenta and yellow trajectories indicate the paths of the tail and snout, respectively. The zebrafish model was used with a different width profile $R(u)$ calculated according to Section 4.3. This demonstrates that the algorithm is applicable across different organisms.

facilitate studies requiring the analysis of many and long image sequences.

In addition to traditional swimming performance indicators, I explored two new ways of analyzing the kinematics data; by plotting the angular acceleration as a function of time and the frequency response along the body. I also used more objective mathematical definitions and corresponding algorithms to quantify standard variables such as tail beat frequency, wave speed, and length. The center of area (COA) of the fish dorsal view was proposed as a valid location for comparison between wildtype and *stocksteif* fish because its ease of measurement lends itself to high-throughput analysis. This preliminary comparison between wildtype and stocksteif swimming performance indicators already suggest significant differences. Hence, this preliminary data analysis of swimming fish illustrates the capabilities of the automatic fish tracker and bodes well for gaining a complete understanding of how stiffness of the vertebral column affects swimming performance.

Chapter 5

Automated Visual Tracking for *C. elegans* Mating Behavior Analysis

The work carried out in this chapter was done in collaboration with Allyson Whittaker and Paul Sternberg of the Sternberg Laboratory at the California Institute of Technology. The video sequences used in Section 5.5 were recorded by either Christopher J. Cronin or Allyson Whittaker, and both Christopher and Allyson provided extensive help and feedback during the analysis of these sequences.

5.1 Introduction

The small nematode Caenorhabditis elegans (*C. elegans*) is a widely used model organism in the study of genetics and developmental biology. Since the exact position and cell lineage of all of its 959 cells and 302 neurons is known, *C. elegans* offers a convenient platform to understand how different behaviors are modulated by the nervous system, sensory input, and genetic modifications. However, to study the relationship between behavior and genes in *C. elegans*, for example, requires the *screening* of an enormous number of individual *C. elegans* specimens to establish the gene-behavior relationship. Each screen consists of a visual observation of the specimen's movements over a time period, and the extraction of key movement and behavior parameters from the observations. The number and complexity of the observations that are needed to support current research into the gene-behavior linkage argues for highly automated screening systems. In this particular study, I focus on applying my visual tracking algorithm to the mating behavior of male *C. elegans* (Figure

5.1), which represents arguably the most complex behavior the organism exhibits.

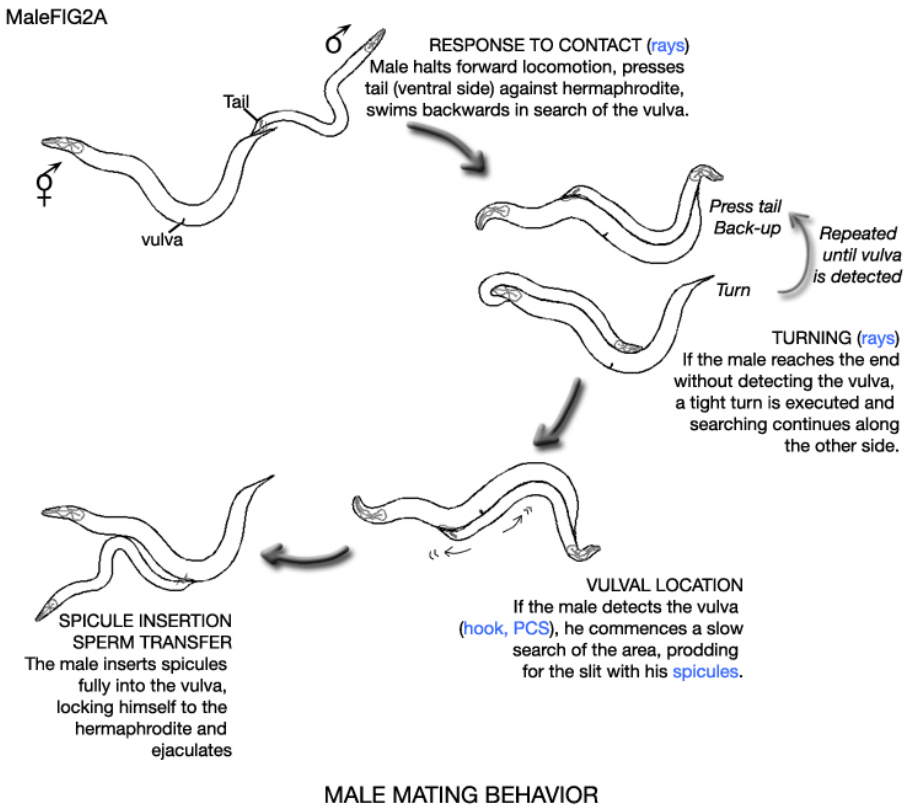


Figure 5.1: Illustration of basic steps involved in male mating behavior (Taken from [66]). Various mutants exhibit deficiencies in different steps of the behavior. For instance, some fail to respond after making contact, while others fail to turn at the end of the hermaphrodite. To understand how genes control the neuronal and sensory function of the worms during mating, the detailed kinematics of the organisms must be captured during interactions. However, mating presents the challenge of tracking two worms that severely occlude each other through long periods of time.

Several automated tracking and analysis systems have been previously developed for *C. elegans*. These tracking systems can be roughly divided into two categories. The first class of systems track multiple worms at a low magnification, and they are able to capture the animals' gross motion characteristics, such as velocity and frequency of reversals [27]. Systems in the second category track single worms at a higher magnification and utilize a motorized stage to keep the worm in the camera field of view [23], [43]. These systems can quantify the detailed posture of the individual worms, and subsequently perform phenotype classification. However, limitations in the underlying implementations of these systems prevent them from tracking detailed worm posture for multiple

organisms that may occlude each other.

Recently, several automated systems have been developed for tracking multiple *C. elegans* nematodes [87, 37, 36, 50]. In [87] and [50], the authors both use a region-based approach that tries to maximize the overlap between their worm model and worm pixels. However, they do not take advantage of contour information to refine the pose estimation and achieve greater accuracy. In [37], I proposed an edge-based tracker that achieved high accuracy, however, manual intervention was required when the head and tail of the worms remained occluded for more than a few frames. In [36] I proposed an extension of the algorithm that incorporated a region model based on level sets in addition to the edge-based cues.

This chapter presents my method for tracking multiple *C. elegans* specimens during mating. The tracking algorithm combines detailed geometric models with nonlinear estimation techniques to quantify the shape and motion of the worms. Sections 3.2 and 5.2 respectively describe the geometric modeling approach and the motion model used in the dynamic state space framework, while Section 5.4 reviews the details of the observation model. Finally, Section 5.5 presents tracking results and data analysis for several challenging video sequences.

5.2 Motion Model

The motion model assumes that the organism undergoes axial progression along its length with a constant wave velocity, η_1, corrupted by acceleration noise, η_2. Thus, the state vector and process noise vector for a single nematode are identical to the zebrafish (Section 4.2):

$$p = \begin{bmatrix} \vec{\alpha} & \vec{T} & \eta_1 \end{bmatrix}^T \qquad \xi = \begin{bmatrix} \Delta\vec{\alpha} & \Delta\vec{T} & \eta_2 \end{bmatrix}^T \tag{5.1}$$

The equations of motion calculate the predicted state vector after the nematode has undergone a total axial displacement of $\eta = \eta_1 \Delta t + \eta_2 \frac{\Delta t^2}{2}$, where Δt is the inverse of the camera frame rate

(Figure 5.2). These equations take the form:

$$
\begin{aligned}
p_k &= f(p_{k-1}, \xi_{k-1}) \\
\begin{bmatrix} \vec{\alpha_k} \\ \vec{T_k} \\ \eta_{1,k} \end{bmatrix} &= \begin{bmatrix} \Lambda(u)^{-1}\Lambda(u+\eta)\vec{\alpha}_{k-1} \\ \vec{T}_{k-1} + \int_0^{\eta} \begin{bmatrix} \cos(\Phi(\hat{u})\vec{\alpha}_{k-1}) \\ \sin(\Phi(\hat{u})\vec{\alpha}_{k-1}) \end{bmatrix} d\hat{u} \\ \eta_{1_{k-1}} \end{bmatrix} + \begin{bmatrix} \Delta\vec{\alpha}_{k-1} \\ \Delta\vec{T}_{k-1} \\ \eta_{2,k-1}\Delta t \end{bmatrix}.
\end{aligned} \tag{5.2}
$$

This equation is similar to the prediction model of zebrafish (Equation (4.6)), except the predicted shape parameters are calculated differently. The interpolated bend angle function, $\Theta(u+\eta) = \Lambda(u+\eta)\vec{\alpha}_{k-1}$ is calculated by projecting the previous shape parameters, $\vec{\alpha}_{k-1}$, onto the B-spline basis in the domain of displacement, $\Lambda(u+\eta)$. Then, the new shape parameters, $\vec{\alpha}_k$, are calculated by projecting back onto the original basis, $\Lambda(u)$. Likewise, the predicted translation is calculated by integrating the tangent vector over the axial displacement, η.

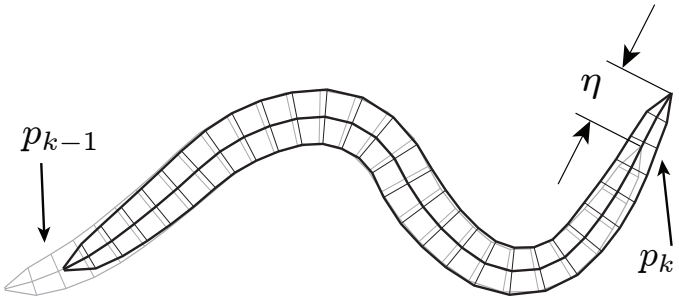


Figure 5.2: Motion model of *C. elegans*. The model is based on the assumption that the worm undergoes axial displacement η along its length between frames.

C. elegans nematodes exhibit a wide array of body deformations during their different behavioral modes. Although their typical form of motion during foraging behavior is a traveling sinusoidal wave, other behaviors, such as mating, exhibit highly erratic and irregular motions. For this reason, I do not adopt the motion model used in zebrafish for nematodes. Since fish exhibit stereotyped body undulations to propel themselves in water, matching this motion pattern to a database provides accurate prediction (Section 4.2). However, such stereotyped motions are not present in nematode mating behavior, so a representative database in unlikely to exist. Instead, this current approach offers a simpler prediction model given the small inter-frame motion in the video sequences.

Instead of extrapolating the bend angle function to calculate the predicted shape parameters, the centerline itself can be extrapolated to calculate the predicted shape parameters. Let

$$\vec{x}_{k-1}(u) = \int_0^u \begin{bmatrix} \cos(\Phi(\hat{u})\vec{\alpha}_{k-1}) \\ \sin(\Phi(\hat{u})\vec{\alpha}_{k-1}) \end{bmatrix} d\hat{u} \tag{5.3}$$

denote the centerline of the model as calculated in the previous time step. A spline is fit to these points such that

$$\vec{x}_{k-1}(u) = \sum_{j=1}^{N_{\vec{x}}} a_j \Phi_j(u). \tag{5.4}$$

This spline is used to extrapolate to the worm's centerline, $\vec{x}_k = \sum_{j=1}^{N_{\vec{x}}} a_j \Phi_j(u+\eta)$. Let $\mathbf{T}(u)$ be the local tangent vector of the extrapolated curve $\vec{x}_k$. The tangent vectors are use to estimate the local bend angles, and the bend angle is projected back onto the B-spline basis to get the predicted shape parameters:

$$\Theta_k(u) = \text{atan2}(\mathbf{T}_y, \mathbf{T}_x) \tag{5.5}$$

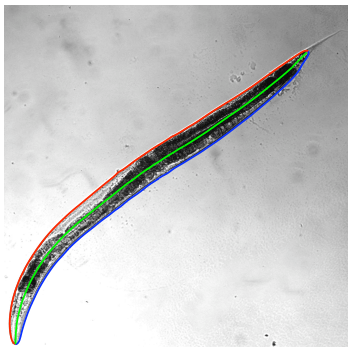
$$\vec{\alpha}_k = \Lambda(u)^{-1}\Theta_k(u). \tag{5.6}$$

Both methods provide adequate prediction when the interframe displacement is not too large, however, the second method is more accurate when η is larger. In the implementation, the basis functions $\Phi_j(u)$ used to extrapolate the centerline are piecewise cubic polynomials.

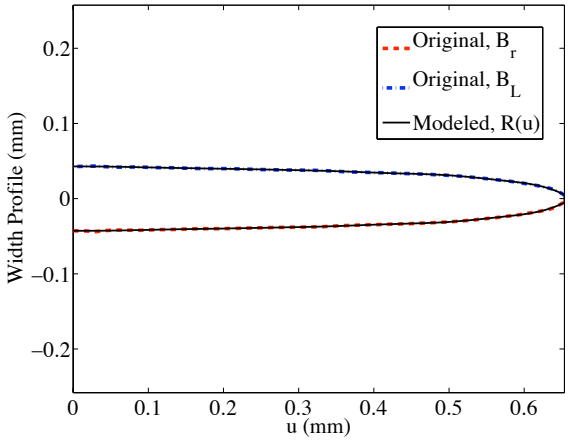
5.3 Model Initialization

I assume that all worms of a particular sex have the same width profile, $R(u)$. This function is estimated from high-resolution images of a male and a hermaphrodite example specimen (see Figure 5.3a), where a semi-automated routine extracts the boundary and estimates $R(u)$ (see Section 4.3 for details). Thus any worm in the video sequences is assumed to have a scaled version of the width

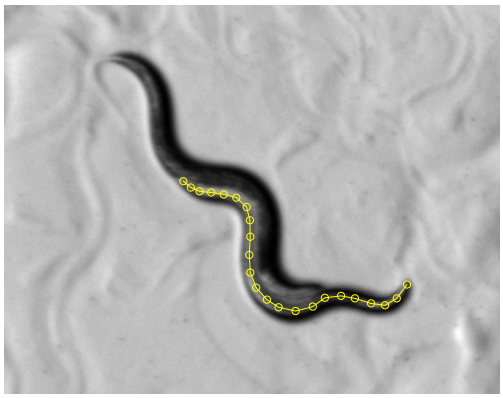
(a) High-resolution images of the hermaphrodite (left) and male (right) *C. elegans* used to estimate the width profile of the worms



(b) Extracted boundaries (red,blue) and estimated centerline (green) of the worm



(c) Estimated $R(u)$ with original extracted boundary illustrating quality of shape estimation



(d)

Figure 5.3: (a)–(c) Estimation of width profile in worm model. A separate calculation is performed for both males and hermaphrodites. Details of calculation are provided in Section 4.3. (d) I manually estimate the initial state (i.e., pose) of the worm model by clicking centerline points in the image. An automated routine was not explored since the worms are always in contact for most of our video sequences.

profiles estimated from these images. To estimate the initial state p_0 of the model, I utilize a manual approach where the user clicks on points along the centerline in the image (see Figure 5.3d) and the shape parameters are estimated according to (5.6). Because the mating worms were often in contact throughout the entire video sequence, I could not implement a fully automated initialization routine as in [87], which assumes the worms are not touching at the initialization frame.

5.4 Observation Model

The observation model I designed for tracking multiple *C. elegans* consists of two components, a region model and a contour model. The region model was designed to improve tracking performance in situations like Figure 5.4 where, despite strong edges, it is difficult to localize the worm in the axial direction during the partial occlusions common to mating behavior. The region model tries to maximize the overlap between the geometric models and worm pixels. To complement the region-based approach, a contour model that uses local edge feature points, similar to the one presented in Section 4.4, is also utilized. Here I explain the details of each approach, while Figure 5.4 provides an illustration of the complete observation model.

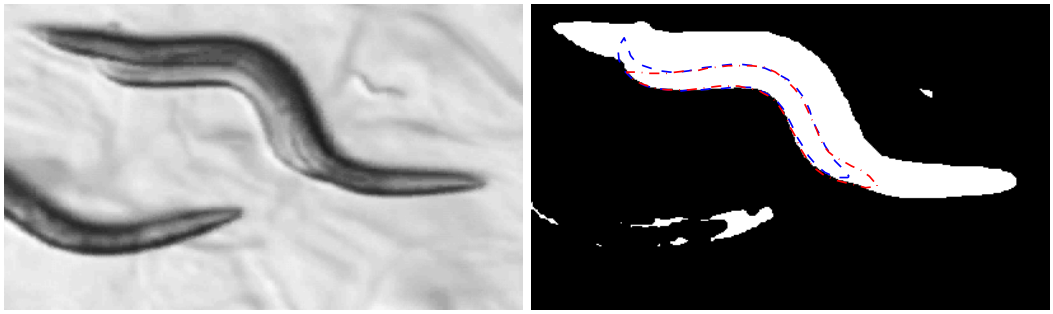


Figure 5.4: Despite the presence of strong edges, it is difficult to localize the worm in the axial direction during partial occlusions present in mating.

5.4.1 Region Model

Although the Chan and Vese model (Section 2.5) can be applied directly to grayscale images, I apply it to binary images that are calculated using a simple background subtraction model (Figure

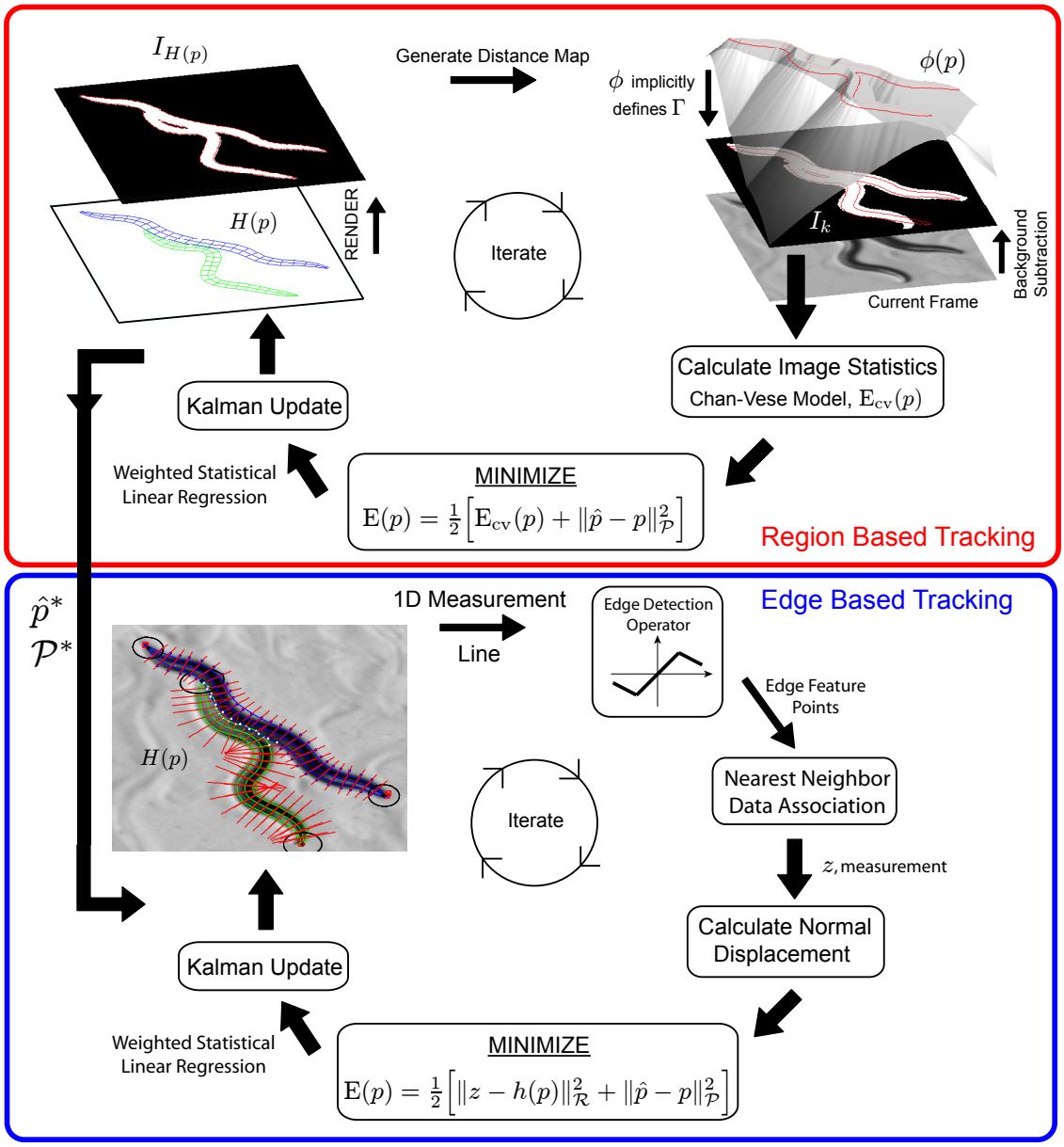


Figure 5.5: Overview of observation model used in multi-worm tracking. The worm models are rendered to a binary image using the desired camera model. The extracted boundaries are used to create a distance map $\phi(p)$ that implicitly defines the contour. The Chan-Vese functional is incorporated into the error metric and minimized using a SPKF. The output of the region model is used as an initial estimate that is refined using local edge feature points. Feature points are detected by performing 1D search along the normal to the model contour. The distance between the points is also minimized using the SPKF.

Figure 5.6: Because the background of the petri dish remains largely static, accurate segmentation of the observed data images I_k is achieved using background subtraction.

5.6). Because microscope lighting effects may cause regions of the worm to be nearly camouflaged in the background, I determine the pixels belonging to the worm by background subtraction instead of relying on the image functional to partition the image based on the distribution of grayscale intensities. This approach is primarily used to provide improved tracking of multiple organisms, as I do not currently constrain the shape priors to prevent intersection of different worms because this occurs frequently in the intended applications.

The binary image serves as the data observation at the k^{th} frame, $z_k = I_k$. An overview of the worm observation model is illustrated in Figure 5.7. The rendered image of each organism model defined by (3.15) is joined to create the total image $I_{H(p)} = \bigcup_{i=1}^{K} \text{Render}(H(p^i))$. The boundaries of this image are extracted using the built in Matlab$^{\text{TM}}$ function *bwboundaries*. Once the boundary locations $\partial\Omega$ have been found, the values of c_1 and c_2 can be calculated directly. I first determine the domains:

$$\Omega^I = \big\{(x_i, y_i) \mid I_{H(p)}(x_i, y_i) = 1, (x_i, y_i) \notin \partial\Omega\big\} \tag{5.7}$$

$$\Omega^O = \big\{(x_i, y_i) \mid I_{H(p)}(x_i, y_i) = 0\big\}, \tag{5.8}$$

which determine the sets of pixels inside and outside the worm boundary, respectively. The observation model $h(p)$ is then calculated as follows:

$$h(p) = \begin{cases} c_1 = \frac{\sum_i^N I_k(x_i, y_i)}{N} & (x_i, y_i) \in \Omega^I \bigcup \partial\Omega \\ c_2 = \frac{\sum_i^M I_k(x_i, y_i)}{M} & (x_i, y_i) \in \Omega^O. \end{cases} \tag{5.9}$$

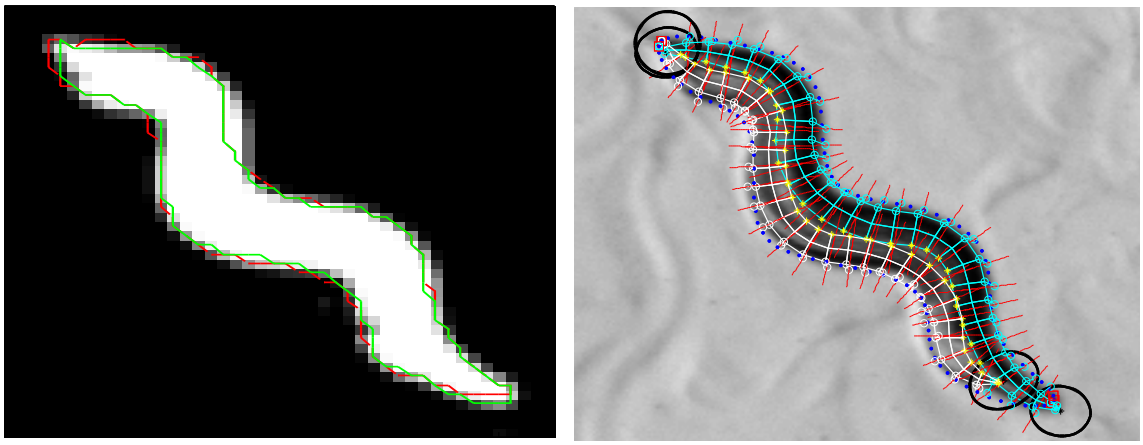
I assume that the image pixels are corrupted by additive noise with unit variance, $\omega_k \sim \mathcal{N}(0, \mathbb{I})$.

Using this model, the observation dimension is equal to the number of pixels in the image. To decrease the computational cost associated with inverting the observation covariance in the Kalman filter update step, I make two modifications. First, I_k is trimmed to the smallest window that contains the closed boundaries that belong to the worms. I also subsample the images I_k and $I_{F(p)}$ using a standard Gaussian pyramid with unit variance. The observation update is then performed at a level 2 of the pyramid where level 0 is the original resolution (see Fig. 5.7a).

5.4.2 Contour Model

The contour observation model is illustrated in Figure 5.7b and is identical to that used for zebrafish tracking (Section 4.4). It consists of the model boundary points, $q_i = \binom{x_i}{y_i}$, along with their outward normal vectors, $n_i = \binom{n_i^x}{n_i^y}$ To find corresponding edge-feature points, the approach of Blake [8] is utilized by applying a 1D edge detector filter to I_k in the normal direction at each of the boundary points in the model. The nearest neighbor detected edge points, r_i, are also projected onto n_i so that the error minimized by the KF has the form $\mathbf{n^T(q-r)}$. I also make two additions to this contour model. A closed B-Spline curve is fit to the boundaries of I_k and locations of high negative curvature are calculated. These feature points likely correspond to head and tail locations and are matched with the model if they pass a Mahalanobis test [1] (i.e., $\|q_i - r_i\|_{\Sigma_i} \leq \gamma$, from Eq. (2.39)) and have a similar normal vector orientation. Here, $\Sigma_i = \begin{bmatrix} \sigma_x^2 & \sigma_{xy}^2 \\ \sigma_{xy}^2 & \sigma_y^2 \end{bmatrix}$ is the covariance matrix associated with the model point q_i. Also, due to their cylindrical shape, the worms never share a visible boundary while they interact (i.e., their centerlines are almost never coincident and parallel in any region). This constraint is incorporated by rejecting any detected edge points for one worm where the measurement lines cross over the model location of other worms. This idea is also modeled in [87] using a "worm overlap energy" and is a simpler notion of the exclusion principle presented in [68].

[1] For normally distributed measurements, the Mahalanobis distance is chi-squared distributed with number of degrees of freedom equal to the dimension of measurement vector—2 in this case. The probability that the distance is less than the parameter γ can, therefore, be obtained from χ^2 distribution tables. For example, $\gamma = 5.99$ corresponds to a 95% probability that the feature r_i is associated with the model point q_i.



(a) (b)

Figure 5.7: Observation models, $h(p_k, \omega_k)$ used in the proposed tracker: (a) Illustration of the observation model for region-based tracking. Initial condition (red) and final solution (green) are plotted over the subsampled data image, I_k. Contours represent the zero level set of $\phi(p_k)$. The contour has to be evolved little because the motion model provides a good initial estimate and the motion between frames is small; (b) Observation model for contour-based tracking. Edge features are detected along 1D measurement lines. Model locations are labeled as occluded (yellow stars) if no match is detected. High curvature points (red squares) are matched with head/tail locations.

5.5 Tracking Results

The tracking algorithm is demonstrated on 4 worm sequences. All videos are acquired using a stereographic microscope outfitted with a 30 Hz camera at 720x480 or 640x480 resolution. For each sequence, the model pose parameters are manually initialized by clicking points along the centerline of each organism to match the configuration at the first frame (Section 5.3). The algorithm is written in MATLAB$^{\text{TM}}$, uses the state estimation toolbox of [112], and operates in a batch off-line mode.

In the multiple worm sequences, the hermaphrodite (larger worm) is partially paralyzed to keep the mating behavior within the camera's field of view. Research into the genetic basis of mating behavior mainly involves studying the male motion. The male worm has 41 distinct muscles that are used in mating and not present in hermaphrodites [66]. Mating consists of two behavior modes that are dominated by motion: backing and turning. Quantifying the body posture of the male during backing and turning is an important metric for understanding the genetic and neuronal basis of mating. For instance, certain mutants either cannot turn, or hesitate a lot before turning. Figure 5.7 shows four sequences where the tracking algorithm is able to successfully track both worms while

the male backs and turns. The first row illustrates the failed tracking result when just the contour observation model is utilized. Because the male model is only using the lower edge information to localize itself, the tracker cannot detect when it crawls underneath the hermaphrodite. However, the partial occlusion is successfully handled by incorporating the region-based module.

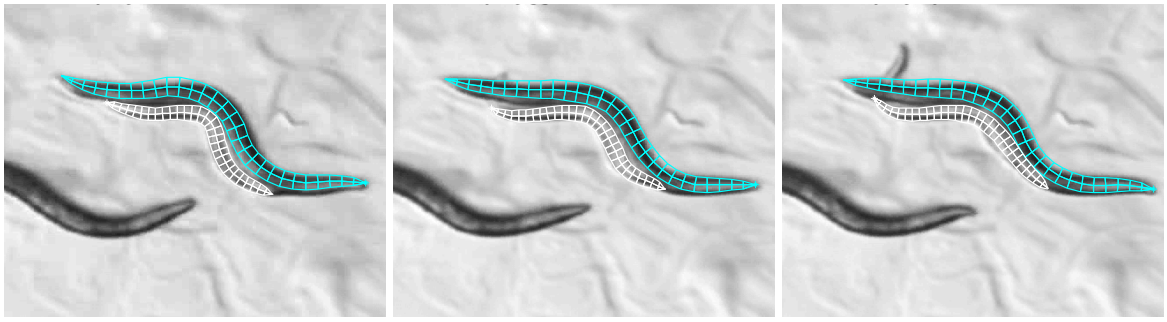
5.6 Kinematic Data

The data presented in this section is meant to illustrate the range of information provided by the worm tracking algorithm and demonstrate important kinematic parameters that were previously inaccessible using manual methods. It is not meant to answer any particular biological question. However, it does provide a foundation for future algorithms designed for automated behavior recognition.

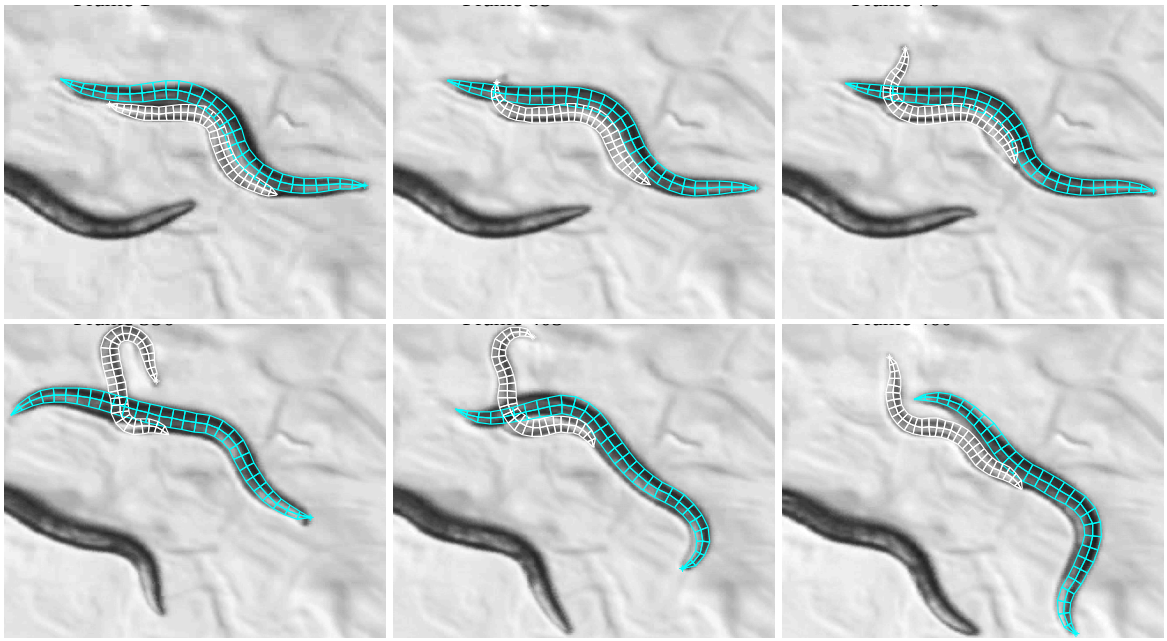
Fig. 5.8 illustrates the specific curvature profile (curvature normalized by body length) of a wildtype and tph1 mutant male worm. This mutant has a defect in the gene used to produce the enzyme *tryptophan hydroxylase*, the key catalyst in producing serotonin, which regulates male mating behavior. The estimated centerlines of the worm are smoothed and the curvature is calculated according to Section 4.6. This particular measurement provides insight into the turning mechanisms of the male. A successful turn is indicated by the anterior propogation of ventral bending as seen in the wildtype. In the unsuccessful turns of the mutant, the ventral bending does not propogate anteriorly. This representation provides information about how the male is configuring his body, but not about its location relative to the hermaphrodite.

To visualize this information, consider Figure 5.9. We define Ψ as the relative angle between the male tale and the hermaphrodite body and $\bar{u}$ as the normalized distance along the hermaphrodite where the male tail is located ($\bar{u} = 0$ is middle of worm and $\bar{u} = 1$ is tip). This convention was chosen because the male will often turn around both ends of the hermaphrodite, so the natural place to put the origin is in the middle of the worm. When the male is backing, Ψ is a small angle ($\Psi \leq \frac{\pi}{2}$), but once he begins to turn, Ψ increases significantly.

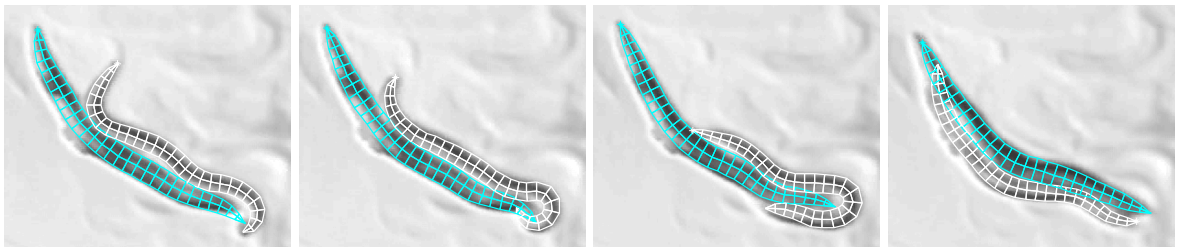
Figure 5.10 provides preliminary data using this representation. The trajectories in the polar plot



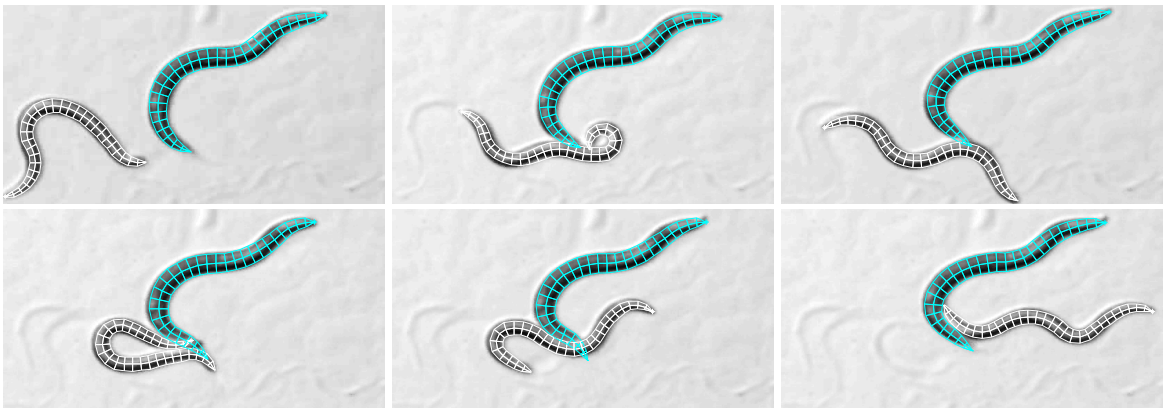
(a) Using only edge information, tracking of the male worm fails when it is partially occluded.



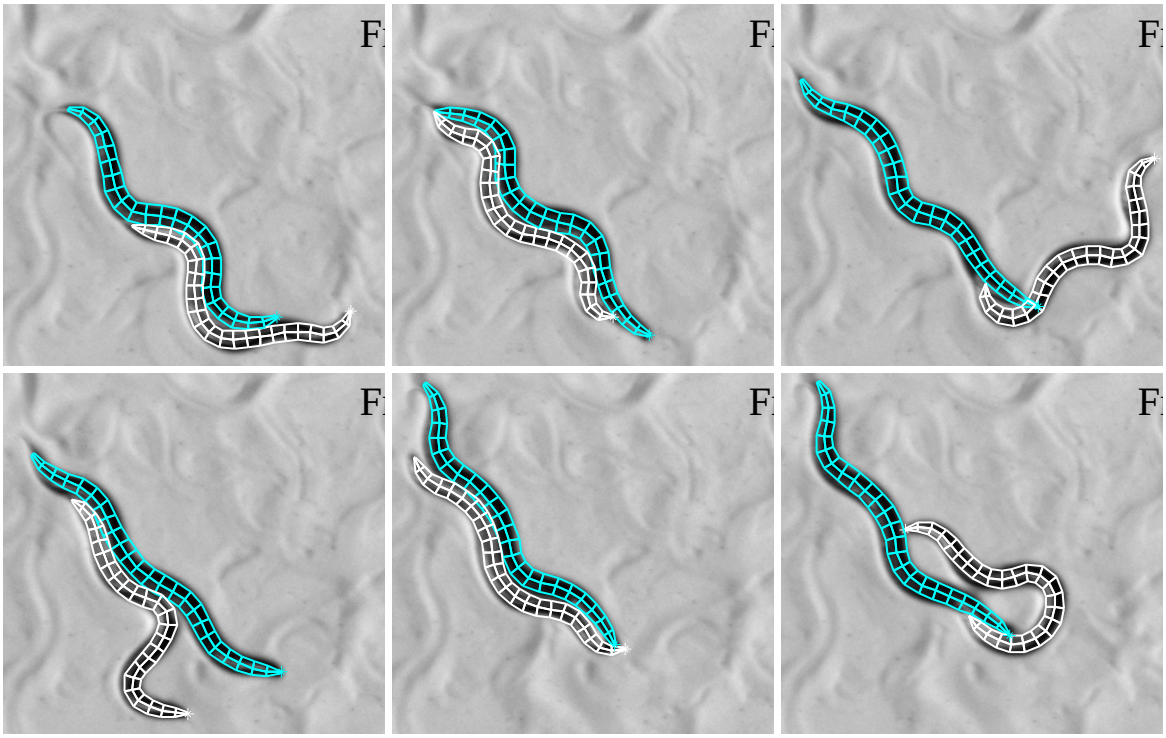
(c) Tracking using region and edge based cues successfully handles the partial occlusion.



(d) This mutant hesitates but successfully completes a turn.



(f) The implicit representation of ϕ handles the changing topology of the image contour in the second and fourth still frames when the male worm curls its body.



(h) Tracking mutant male that performs backing along hermaphrodite but hesitates and fails to complete a turn.

Figure 5.7: Tracking results of four different *C. elegans* mating sequences. Model estimate of male and hermaphrodite appear in white and cyan, respectively.

indicate the relative orientation between the male's tail and the hermaphrodite body. The wildtype trajectories demonstrate consistent behavior where the male keeps contact with the hermaphrodite ($\bar{u} \leq 1$) and relative angle goes from $\Psi \leq \frac{\pi}{2}$ to $\pi \leq \Psi \leq \frac{3\pi}{2}$, meaning his tail went from one side of the hermaphrodite to the other. In contrast, the mutant trajectories are highly variable. For instance, one worm moves off the end of the hermaphrodite and loses contact ($\bar{u} > 1$) indicating a "missed turn", while another curls around the end of the hermaphrodite, but fails to continue backing and "over-curls" on itself ($\Psi > \frac{3\pi}{2}$). The "stuttering" turn illustrates when the male remains moving back and forth near the end of the end of the hermaphrodite, but does not initiate a turn to the other side. The "good", "sloppy", and "missed" turns were described by Loer using qualitative human observation [67]. The data in Figure 5.10 provide quantitative metrics that enable the distinguishing behaviors to be more precisely classified than is possible by human observation.

5.7 Discussion

The algorithm presented in this chapter presents a good foundation for tracking multiple *C. elegans*. However, there are several shortcomings that should be addressed in future work. First, the algorithm tends to loose track of the male when he turns over top or underneath of the hermaphrodite. This partial occlusion is unlike that seen in Figure 5.8c because after the male reaches the opposite side of the hermaphrodite, he curls his tail tightly against her body leaving little visual information (i.e., pixels) that the turn has occurred (Figure 5.11). This 3D motion from a monocular view clearly causes the posterior distribution to become multi modal (i.e., there are multiple hypotheses about where the worm could be located. As a result, our SPKF solver, which assumes a unimodal state distribution, fails to track. Sequential Monte Carlo methods or multiple hypothesis tracking techniques would probably solve this problem. Another possibility to improve this problem is the design of a much stronger motion model (i.e., prior) that better predicts the motion of the worm during mating. The one described in Section 5.2 works well for backing and foraging behavior, but because deformations are not predicted, it does not work well for turning. The motion of *C. elegans* is highly erratic, however, so care must be taken to incorporate the appropriate amount of

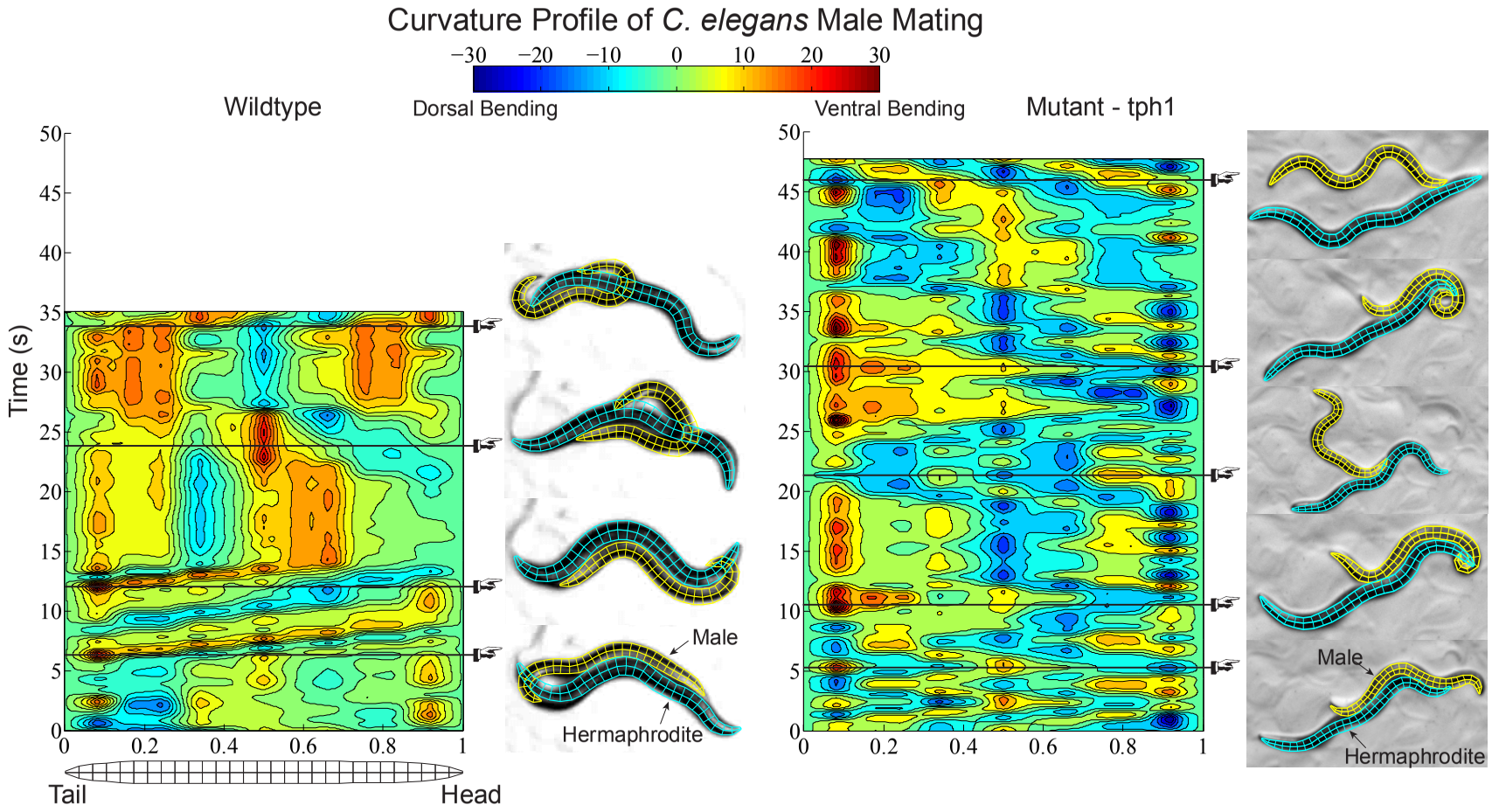


Figure 5.8: Specific curvature profiles of wildtype and tph1 mutant *C. elegans* during backing and turning steps of mating. The ventral curling (red regions) indicates the male is trying to curl around the hermaphrodite. A successful turn is indicated by the anterior propogation of ventral bending as seen in the wildtype. In the unsuccessful turns of the mutant, the ventral bending does not propogate anteriorly.

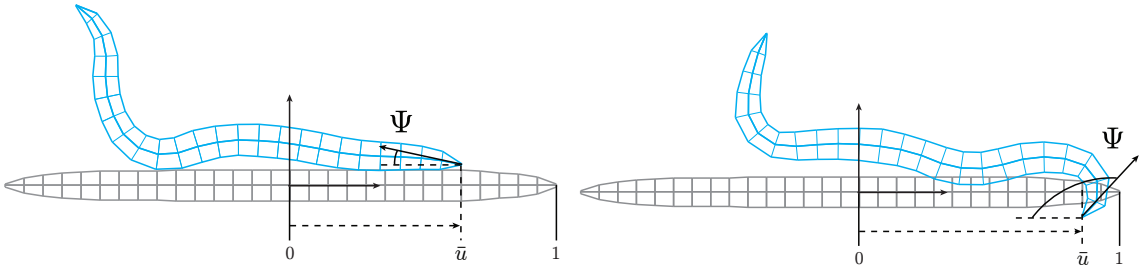


Figure 5.9: Illustration of male turning angles. The relative angle between the male tail and the hermaphrodite body is small during backing (left) but increases greatly during turning (right). The variable $\bar{u}$ measures the relative location of male's tail along the hermaphrodite body.

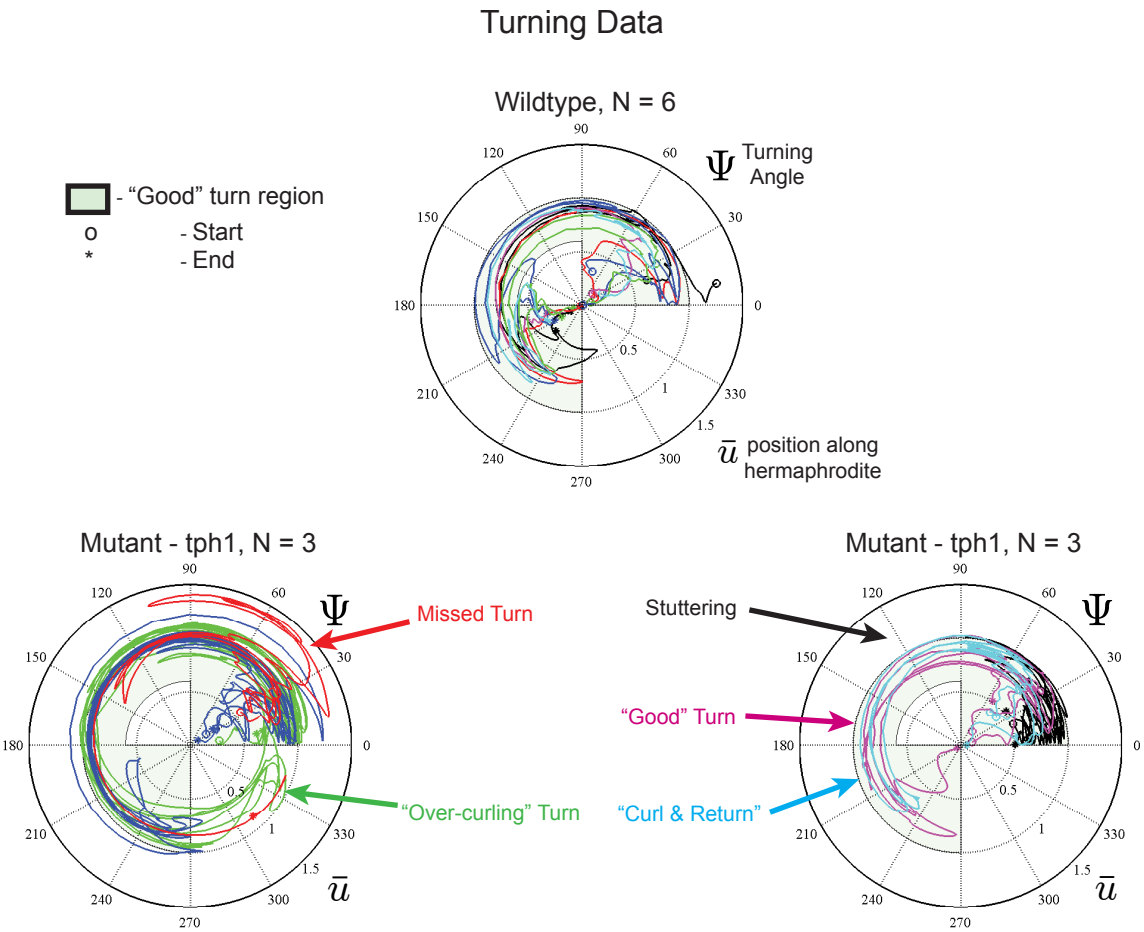
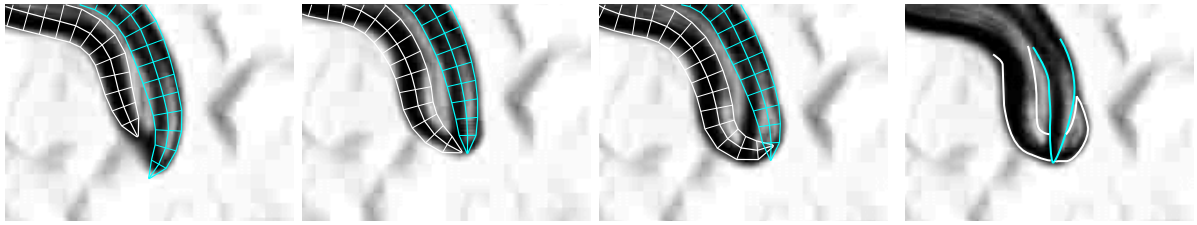


Figure 5.10: Polar plot of $(\bar{u}, \Psi)$ for 6 wildtype males (top) and 6 tph1 mutant males (bottom). I have arbitrarily indicated a "good" region of turning, where the successful turns of the wildtype worms occur.



(a) (b)

Figure 5.11: Failure mode of current multi-worm tracking algorithm. The turning behavior of wild-type male worms involves a tight turn where he flips his tail to the other side of the hermaphrodite (right). This maneuver typically involves 3D motion (e.g., the male crawls under the hermaphrodite causing it to rise out of the plane (middle)). This 3D motion within a monocular video clearly causes the state distribution to become multi-modal. Hence, the Kalman-filter based local optimizer loses track. (b) The true configuration of the worms

uncertainty.

In addition, my focus on mating behavior caused this analysis to ignore tracking more than two worms simultaneously. The model representation used 11 parameters per worm, thus tracking 10 worms would involve estimating 110 parameters simultaneously, requiring a greater computational burden. A more compact representation of the state space could mollify this problem. Despite these shortcomings, our algorithm demonstrates promising results that can be expanded with further refinements. In addition, with the use of a microscope with stereo cameras, depth information can be measured from the images. By modeling the worms as 3D, the out of plane motion present in *C. elegans* mating can be estimated directly.

Chapter 6

3D Visual Tracking of *Drosophila* Flight Maneuvers

The experiments carried out in this chapter were done in collaboration with Gwyneth Card, Will Dickson, and Michael Dickinson of the Dickinson Laboratory at the California Institute of Technology. Will Dickson provided the digitized surface points of a *Drosophila* body and wings that were used to construct the generative model. In particular, Gwyneth Card recorded all video illustrated in Section 6.8 and made her customized manual tracking software and body kinematic data from [16] readily available.

6.1 Introduction

The fruit fly is one of the most important model organisms used in modern biology. It offers an ever-increasing and widely accessible set of methods for altering genes and controlling their temporal and spatial expression. For example, these methods now make it possible to manipulate the activity of neurons in intact flies using pulses of light [65]. Although their nervous system contains only 300,000 neurons, flies display an array of complex behaviors. One group of behaviors that represents an exciting area of research is the flight maneuvers. The sensory-motor response time of *Drosophila* is on the order of a few milliseconds, making it one of nature's fastest solutions to the problem of flight control. In order to characterize *Drosophila's* sensory-motor control system and to understand how different behaviors are modulated by the nervous system, sensory input, and genetic modifications,

scientists utilize high-speed cameras to record the complex behaviors. Previous studies in insect flight maneuvers have required laborious manual methods to capture the body and wing kinematics [3, 16, 39, 40] from these high-speed videos. The time-consuming nature of this approach prohibits further experiments to characterize other flight behaviors. Thus, an automated tracking technique that estimates the complete body posture of the fly is necessary to produce the number of complex observations needed for current research.

To address this concern, I develop an automated model-based tracking technique to capture the 3D body and wing motion of *Drosophila*. Previously, many studies in *Drosophila* flight control measured the relative wing motion during tethered flight by shining an infrared light upon the fly and measuring the resulting shadow with a photodiode receptor [31, 45]. Here, the 3D wing motion is reduced to a 1D voltage signal on the photodiode. Recently, Graetzel et al. developed a real-time computer vision system to measure the wing motion of a tethered fly [46]. Here, a single camera view is used to track the angular position of the wing's leading and trailing edge in the projected camera view. In [119], Zanker measured the full 3D motion of flies during tethered flight using stroboscopic video and mirrors to capture multiple views. However, the 3D reconstruction relied on manual digitization of six keypoints on the wing in each camera view. Later, Fry developed customized software to manually fit 3D wing models to free-flight *Drosophila* in multiple camera views [39]. This technique was expanded to analyze hovering and take off behaviors in fruit flies and honey bees [3, 16, 40]. The proposed algorithm extends the work of [39] by developing visual tracking techniques to *automatically* fit a known 3D fly model to images captured from multiple calibrated camera views.

In the computer vision literature, many encouraging techniques can be found for estimating the 3D rigid motion of a human from multiple calibrated camera views (see Section 1.1.2 for a detailed review) [73]. Typically, a 3D human model containing kinematic chains is given, and the goal is to estimate the body posture and joint angles using image measurements (e.g., silhouettes, appearance textures, optical flow). Although my approach to tracking *Drosophila* flight maneuvers builds upon several key ideas from the human motion tracking literature, there are several challenges

peculiar to *Drosophila* that require special attention. For instance, I incorporate the motion model of Rosenhahn (Section 2.7) to accurately predict the pose of the fly, given its pose from the previous time steps. However, this motion model must be extended to include a quaternion representation of rotations. *Drosophila* exhibit complex wing rotations during flapping flight that requires a global parameterization of $SO(3)$. Although a local parameterization using joint angles is sufficient for Rosenhahn to predict human motions, this same parameterization would undergo singularities for wing motion and result in incorrect prediction.

Another challenge specific to *Drosophila* is that the near-cylindrical shape of the its body makes it difficult to estimate the roll angle about the body axis (i.e., the head to tail axis exhibits strong rotational symmetry). Estimating unobservable states has been addressed in the human tracking literature (e.g., depth ambiguities and rotations about axes of symmetry in limbs) primarily within the context of monocular video [96, 97]. However, given our intended application of precise motion tracking and use of multiple cameras, these techniques are outside the scope of our problem. Recently, the authors in [63] demonstrated the ability to track rotations of spherical objects and solids of revolution by integrating the displacement of texture features into the update procedure of their CAD model. Unfortunately, our video of *Drosophila* is void of any robust features except the silhouette (see Figure 6.1 for an example). The extremely high frame rate (6000 fps) needed to capture the wing kinematics prevents foreground lighting in many experimental setups, making their texture feature method inapplicable. Also, high-intensity lights generate excessive heat that can damage the fly and/or alter its behavior. Instead, I rely on the gross symmetric motion of the wing beats to provide a cue for the location of the body's dorsal edge. This biomechanical constraint allows us to estimate the fly's roll angle given the location of its wings.

The quaternion representation and unobservability of the body roll angle requires me to impose nonlinear equality constraints upon the state variables (i.e., model pose parameters) within the estimation algorithm. I adopt a Sigma Point Kalman filter based estimator [111, 92] that provides accurate solutions and computational efficiency. Although this is a local minimization scheme, our accurate foreground segmentation, multiple camera views, and prior motion model based on [85]

Figure 6.1: Zoomed images of *Drosophila* synchronously captured from 3 camera views within laboratory environment. The high-speed video offers no strong visual features except the silhouette. Even with three camera views, the complex wing beat motion is difficult to capture due to low observability of the wings at certain postures (left, middle) and motion out of the camera's focal field (right).

allows us to keep the prediction close to the true solution so that the unimodal distribution of the state variables remains a valid assumption. Many techniques for incorporating nonlinear constraints into the Kalman filter framework have been described in the literature [28, 94, 116]. They typically utilize a pseudo-observation approach (i.e., the constraint is an observation with zero variance) or a projection operator to project the estimate onto the constraint surface. However, in these approaches, the estimate is not guaranteed to satisfy the constraint, and these methods often lead to singular covariance matrices. Recently, the authors in [55] introduce a two-step approach utilizing the Sigma Point transform that first constrains the probability distribution and then constrains the conditional mean of the distribution (Section 2.4). We adopt this approach to accurately incorporate the nonlinear constraints imposed by the quaternion representation and unobservable roll angle.

Thus our algorithm is constructed to address the challenges of a particular tracking problem. The geometric generative model used to represent the fly is embedded with the degrees of freedom relevant to our particular application so that the state estimation procedure performs inference on physical quantities relevant to experimental goals. Section 6.2 reviews the quaternion representation of spatial rotations, and Section 6.3 explains the construction of an accurate fly model and the initialization of the model parameters from a video frame. Section 6.4 describes an extension to the motion model of Rosenhahn. Sections 6.5 and 6.6 discuss the foreground segmentation and model fitting techniques. Section 6.8 demonstrates results obtained by applying this method on several

video sequences involving jumping, tumbling, and flight stabilization, in addition to performance validation against manually tracked data.

6.2 Quaternion Rotations

Unit quaternions provide a global parameterization of $SO(3)$ which do not suffer from singularities and are accurate for integrating incremental changes in orientation over time. They are quantities of the form:

$$Q = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}; \qquad \|Q\| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1 \tag{6.1}$$

where q_0 and $\mathbf{q} = (q_1, q_2, q_3)$ are termed the scalar and vector parts, respectively, and the quaternion basis elements satisfy

$$\mathbf{i} \cdot \mathbf{i} = \mathbf{j} \cdot \mathbf{j} = \mathbf{k} \cdot \mathbf{k} = \mathbf{i} \cdot \mathbf{j} \cdot \mathbf{k} = -1$$

$$\mathbf{i} \cdot \mathbf{j} = -\mathbf{j} \cdot \mathbf{i} = \mathbf{k} \qquad \mathbf{j} \cdot \mathbf{k} = -\mathbf{k} \cdot \mathbf{j} = \mathbf{i} \qquad \mathbf{k} \cdot \mathbf{i} = -\mathbf{i} \cdot \mathbf{k} = \mathbf{j}.$$

The conjugate of a quaternion $Q = (q_0, \mathbf{q})$ is given by $Q^* = (q_0, -\mathbf{q})$, and the inverse is denoted $Q^{-1} = \frac{Q^*}{\|Q\|}$. A vector $X = (x, y, z) \in \mathbb{R}^3$ is identified with the 4D quaternion vector space as a *vector quaternion*,

$$\mathbf{x} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k}. \tag{6.2}$$

In the quaternion algebra, rigid body rotations of the vector $\mathbf{x}$ are given by the equation

$$\mathbf{x}' = Q \cdot \mathbf{x} \cdot Q^* \tag{6.3}$$

where $\cdot$ denotes quaternion multiplication. This quaternion product can be conveniently applied using a matrix form. Denoting the quaternion as the 4-tuple $Q = \begin{bmatrix} q_1 & q_2 & q_3 & q_0 \end{bmatrix}^T \in \mathbb{R}^4$, the

product of two quaternions Q and P is equivalently given by the matrix product

$$Q \cdot P = [Q^+]P \qquad \text{where} \quad [Q^+] = \begin{bmatrix} q_0 & -q_3 & q_2 & q_1 \\ q_3 & q_0 & -q_1 & q_2 \\ -q_2 & q_1 & q_0 & q_3 \\ -q_1 & -q_2 & -q_3 & q_0 \end{bmatrix} \tag{6.4}$$

or (6.5)

$$Q \cdot P = [P^-]Q \qquad \text{where} \quad [P^-] = \begin{bmatrix} p_0 & p_3 & -p_2 & p_1 \\ -p_3 & p_0 & p_1 & p_2 \\ p_2 & -p_1 & p_0 & p_3 \\ -p_1 & -p_2 & -p_3 & p_0 \end{bmatrix}. \tag{6.6}$$

Therefore, the rotational transformation in (6.3) can be written as

$$\mathbf{x}' = Q \cdot \mathbf{x} \cdot Q^* \tag{6.7}$$

$$\mathbf{x}' = [Q^+][Q^-]^*\mathbf{x} \tag{6.8}$$

$$\begin{bmatrix} X' \\ 0 \end{bmatrix} = \begin{bmatrix} \mathrm{R}_Q & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} X \\ 0 \end{bmatrix} \tag{6.9}$$

$$X' = \mathrm{R}_Q X, \tag{6.10}$$

which has the familiar form of a rotation matrix applied to spatial points after ignoring the 0 values in the 4^{th} dimension.

6.3 Geometric Model and Its Initialization

In [32], Dickson et al. construct a polygonal model of the fruit fly from multiple calibrated images of the body and wing. This polygonal model consists of a triangle mesh (Figure 6.2a) that is integrated into a physics engine to simulate the dynamics of flapping flight. I use this polygonal

model to construct a parameterized generative model of the fly that contains 3 primitive shapes: the body, head, and wing (see Section 3.4 for details).

The coordinate transformation $\mathbf{M} \in SE(3)$ that defines the position of a body-fixed reference frame relative to a world-fixed frame is given by $\mathbf{M} = \left(\begin{smallmatrix} \mathrm{R}_Q & T \\ \mathbf{0}^T & 1 \end{smallmatrix}\right)$. A kinematic chain of an articulated body is represented as the consecutive application of coordinate transforms. In our particular application, the wing joint of the fly is modeled as a spherical joint at a known location, so the kinematic chain has the form

$$X_j' = \begin{bmatrix} \mathrm{R}_{Q^{body}} & T \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathrm{R}_{Q^{wing}} & T_{bw} \\ \mathbf{0}^T & 1 \end{bmatrix} X_j \tag{6.11}$$

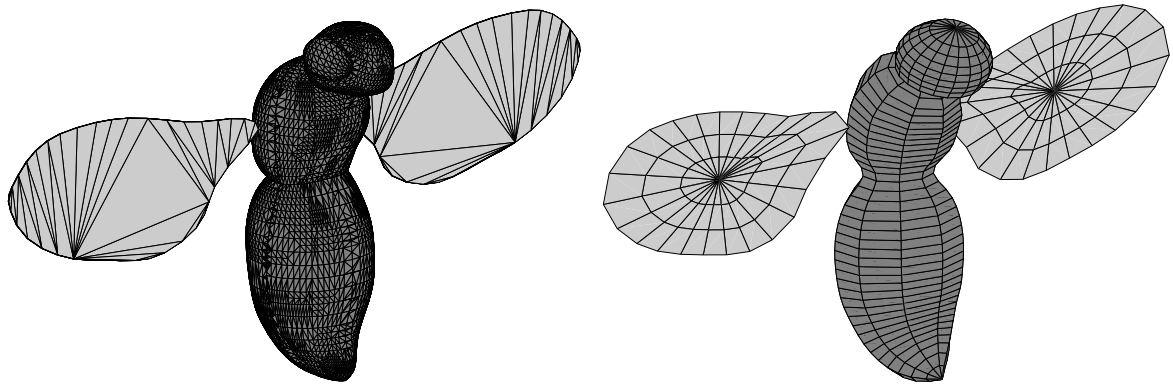
$$X_j' = \mathbf{M}(p) X_j \tag{6.12}$$

where X_j is a 4D vector in homogeneous coordinates that defines the model coordinates in the local frame. The state of the fly model is $p = \begin{bmatrix} T & Q^b & Q^{lw} & Q^{rw} \end{bmatrix}^T$, where the superscripts refer to the body, left wing, and right wing, respectively. Although, the wing joint is modeled as rotations about a fixed joint, that actual location of the joint moves. This could be modeled by extending the the state to $p = \begin{bmatrix} T & Q^b & Q^{lw} & Q^{rw} & \Delta^{lw} & \Delta^{rw} \end{bmatrix}^T$ where $\Delta^{lw,rw}$denotes a displacement of the left/right wing joint locations from their nominal location. The coordinate transformation would be redefined as

$$\mathbf{M}(p) = \begin{bmatrix} \mathrm{R}_{Q^{body}} & T \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathrm{R}_{Q^{wing}} & T_{bw} + \Delta \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{6.13}$$

and the displacement would be constrained to be less than a specified value, $\|\Delta\| \leq \epsilon$.

To initialize the geometric fly model, a customized software package is used to estimate the initial state of the fly, p_0 [16]. This software allows the user to click on six locations on the fly's body in two out of three camera views to localize its 3D position. The six locations include the head, tail, and joint/tip locations of both wings. A frame model is adjusted about its rotational axis until it matches



(a) (b)

Figure 6.2: (a) Triangle mesh of *Drosophila* calculated from multiple calibrated images (courtesy W. Dickson [32]). (b) Generative model constructed according to Section 3.4 based on the data points provided by (a)

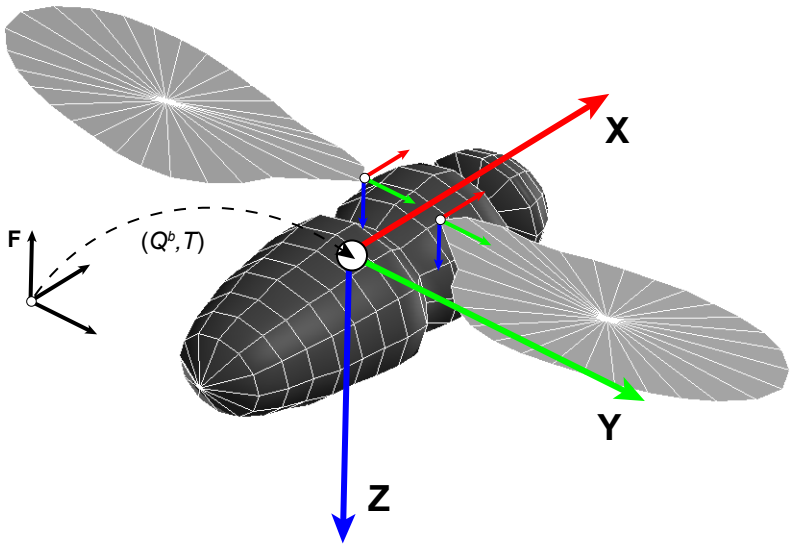
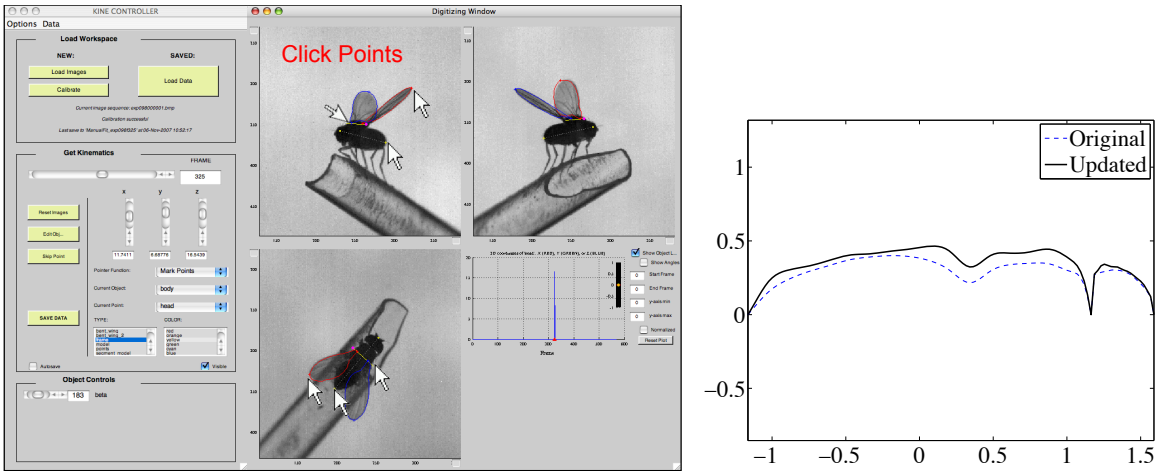


Figure 6.3: Geometric generative model of *Drosophila*. Coordinate frame orientation follows convention common to aeronautics where rotations about the $x, y,$ and z axes are known as *roll*, *pitch*, and *yaw*, respectively. Downward pointing z axis is chosen so that positive pitch angles correspond to pitching upwards.



(a) (b)

Figure 6.4: (a) Customized software for manual digitization of *Drosophila* body kinematics from [16]. Points are clicked at the head, tail, wing joint, and wing tip in multiple camera views to manually fit a geometric model to the images. Manually estimated pose is used as an initial guess for the automated algorithm. (b) At the initial frame, the profile of the body is refined, while holding the pose parameters fixed, to more closely match the actual shape of the *Drosophila* by minimizing the error described in Section 6.6.

the images based on visual inspection (see Figure 6.4a). Next, the body transformation, (Q^b, T), associated with the manual initialization are refined using the registration procedure described in Section 6.6 applied to the body only segmented images, I_{body}, shown in Figure 6.6e. Finally, the shape profile of the body is adjusted, while holding the pose, (Q^b, T), fixed to further match the body only segmented images. The entire width profile is modeled as the combination of two B-spline curves representing the head and body, respectively. The control points of the B-spline curves are adjusted to best match the images, similar to the width profile calculation for zebrafish in Section 4.3. Figure 6.4b illustrates the results of this shape refinement for a particular *Drosophila* that provides a more accurate representation of the appearance.

6.4 Scaled Motion Dynamics

Here, I extend the motion model of Rosenhahn presented in Section 2.7 to incorporate a quaternion representation of rotations. The set of temporally ordered training samples is given by

$$\{\widetilde{p}_i := (\widetilde{T}_i, \widetilde{Q}_i^b, \widetilde{Q}_i^{lw}, \widetilde{Q}_i^{rw}) := (\widetilde{T}_i, \widetilde{Q}_i^b, \widetilde{\mathbf{Q}}_i) | i = 0 \dots N\} \tag{6.14}$$

with body transformations relative to the fixed observing frame given by the translation $\widetilde{T}_i$, rotation (represented in quaternion form) $\widetilde{Q}_i^b$, and joint angle vectors $\widetilde{\mathbf{Q}}_i$ (details on collecting the training samples are given in Section 6.8). We denote this list of temporally ordered training samples as $\mathcal{P} = \langle \widetilde{p}_i \dots \widetilde{p}_N \rangle$, and the sublist in $\mathcal{P}$ of length m ending at time i by $\langle \widetilde{p}_{i-m+1} \dots \widetilde{p}_i \rangle$. In order to predict the state p_{k+1} at the next time step, the training list searched to find the location in the list that best matches the sublist of previous tracked states, $\langle p_{k-m+1} \dots p_k \rangle$. For the matching to be invariant with respect to the velocity of the tracked fly, the matching is performed at different scalings s of $\mathcal{P}$. The different scalings of the training data, denoted $\mathcal{P}^s$, are calculated using two different techniques. The scaled body translations are obtained using linear interpolation and resampling. However, linear interpolation in the 4D space of unit quaternions ignores the 3D spherical subspace where quaternions exist and would result in interpolated values with non-uniform velocity that are not elements of $SO(3)$. To remain on the correct constraint surface and produce valid rotations, the technique of *spherical linear interpolation* (Slerp) is employed [91]. To interpolate between two rotations given by Q_1 and Q_2 on the interval $u \in [0,\ 1]$, calculate

$$\text{Slerp}(Q_1, Q_2; u) = \frac{\sin(1-u)\Omega}{\sin \Omega} Q_1 + \frac{\sin u\Omega}{\sin \Omega} Q_2 \tag{6.15}$$

where $\cos \Omega = Q_1^T Q_2$. The resulting scaled lists are given by $\mathcal{P}^s = \{\widetilde{p}_i^s := (\widetilde{T}_i^s, \widetilde{Q}_i^{b,s}, \widetilde{\mathbf{Q}}_i^s) | i = 0 \dots sN\}$. For the *Drosophila* flight initiation videos that we analyzed, we scan the interval $s = [0.5,\ 1]$ with stepsize 0.1 as the scaling factors because the *Drosophila* were typically observed to undergo equivalent or faster wing motion than our database, which was captured at 6000 fps. The best matching sublist of the training data is then calculated by

$$\underset{s,j}{\text{argmin}} \sum_{v=0}^{m-1} \Big(\|\mathbf{Q}_{k-v} - \widetilde{\mathbf{Q}}_{j-v}^s\| \Big). \tag{6.16}$$

Only the quaternion representation of the wing joint angles are taken into account since their motion will be invariant with respect to the global orientation of the fly. An illustration of this technique is given in Figure 6.5. In order to calculate the predicted wing configuration, the relative rotation

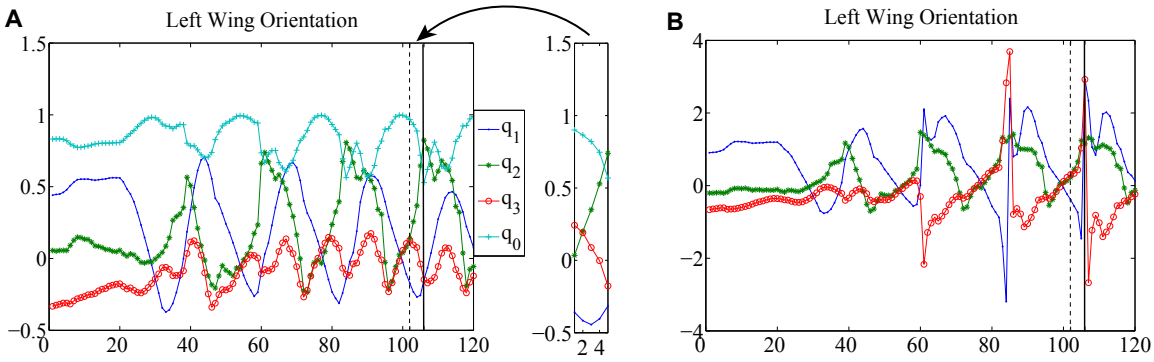


Figure 6.5: (A) Rotational motion of *Drosophila* left wing motion during take off (120 out of 380 samples shown). Motion is parameterized by quaternions which vary smoothly with time. The query of $m = 5$ previously calculated poses is matched with position 106 of the prior database. The relative motion to position 107 is used to calculate the prediction. (B) The same motion as (A) but parameterized by joint angles about constant twists. Because there is no 3-dimensional, global parameterization of $SO(3)$, the wing motion parameters undergo discontinuities as the wing passes through a singularity, resulting in inaccurate prediction of the motion.

between the optimal location in the prior set, j, and its subsequent orientation, $j+1$ is calculated. Then, this relative motion is applied to the current state to generate the prediction of the state variables at the next time frame

$$\partial \widetilde{Q}^{lw,s}_{j+1} = \widetilde{Q}^{lw,s}_{j+1} \cdot (\widetilde{Q}^{lw,s}_{j})^{-1} \tag{6.17}$$

$$Q^{lw}_{k+1} = \partial \widetilde{Q}^{lw,s}_{j+1} \cdot Q^{lw}_{k}. \tag{6.18}$$

An identical calculation to (6.17) is also performed for the right wing and the body orientation. The predicted body translation is given by

$$T_{k+1} = T_k + (\widetilde{T}^{s}_{j+1} - \widetilde{T}^{s}_{j}) \tag{6.19}$$

so the entire predicted state consists of $p_{k+1} = \begin{bmatrix} T_{k+1} & Q^{b}_{k+1} & Q^{lw}_{k+1} & Q^{rw}_{k+1} \end{bmatrix}$. I must note, however, that this prediction assumes a correlation between the joint angle velocity and the body's spatial velocity. In [85], Rosenhahn wisely notes that this assumption is not valid for humans because a larger person runs faster than a smaller person with the same changes in joint angles. Their use of the twist representation for the global body transformation allows them rescale the body motion using

the velocity of the previously calculated state, and not that determined from the prior database. For *Drosophila*, the assumption of correlation between joint velocity and body velocity is valid because there is less variation in body size across the populations used in experimental studies. Nevertheless, in order to track blowflies or other members of the order *Diptera* that are larger than *Drosophila*, the body motion can still be scaled conveniently using the quaternion representation. Let $\widetilde{\theta} = 2\cos^{-1} q_0^b$ denote the body velocity of the predicted motion calculated from the scalar part of Q_{k+1}^b, and θ is the average value of the last m frames calculated from $\langle p_{k-m+1} \dots p_k \rangle$. First, we calculate the twist, $\xi = \binom{v}{\omega}$, associated with the predicted body motion

$$\omega = \frac{1}{\sin\frac{\widetilde{\theta}}{2}}\, \mathbf{q}_{k+1}^b \tag{6.20}$$

$$v = \big((I - e^{\widehat{\omega}\widetilde{\theta}})\widehat{\omega} + \omega\omega^T\widetilde{\theta}\big)^{-1} T_{k+1} \tag{6.21}$$

where $\widehat{\omega}$ denotes the skew-symmetric matrix associated with the vector ω and $e^{\widehat{\omega}\theta} = I + \widehat{\omega}\sin\theta + \widehat{\omega}^2(1-\cos\theta)$ by Rodrigues' formula. The rescaled body transformation $(\bar{T}_{k+1}, \bar{Q}_{k+1}^b)$ is now given by

$$\bar{Q}_{k+1}^b = \begin{bmatrix} \omega \sin\frac{\theta}{2} \\ \\ \cos\frac{\theta}{2} \end{bmatrix} \tag{6.22}$$

$$\bar{T}_{k+1} = \big((I - e^{\widehat{\omega}\theta})\widehat{\omega} + \omega\omega^T\theta\big)v. \tag{6.23}$$

In this way, the kind of motion the body undergoes is determined by the prior data, but the velocity is determined by the previous frames.

6.5 Foreground Segmentation

The *Drosophila* are filmed in a laboratory environment that provides nearly constant background illumination during flight manuevers. Hence, we adopt background subtraction to segment the pixels belonging to the fly. In addition, the appearance of *Drosophila* is very consistent during the video

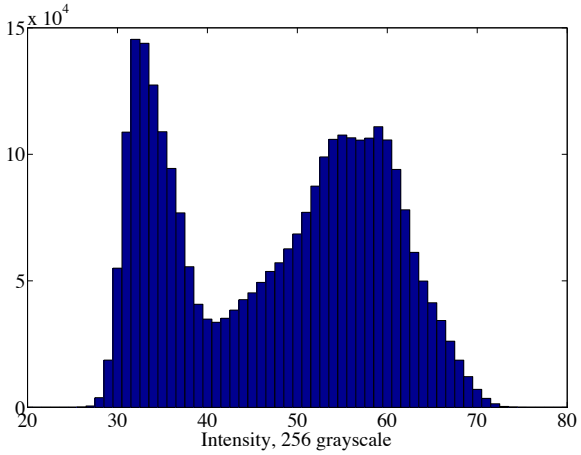
sequences. Figure 6.6a is a histogram of fly pixel intensity values at over 200 frames from 3 different camera views. This characteristic bimodal shape is due to the opaque nature of the body cuticle, which consistently appears darker than the other body parts (i.e., wings and legs). We utilize this appearance consistency to further segment fly pixels into body and appendage groups. At each frame and for each camera, we fit a 1D Gaussian mixture model with 2 members to the segmented fly pixels using the EM algorithm. We then determine the threshold value located at the local minima between the modes of the two Gaussians densities to segment the body and appendages (See Figure 6.6d). For our purposes, the segmentation results in two binary images I_{body} and I_{full} (see Figure 6.6e). This second degree of segmentation is utilized in our model fitting such that body locations in the geometric model are only matched to pixels categorized as body.

6.6 Model Registration

To register the 3D fly model with image measurements, we minimize the distance between a set of corresponding 3D points from the model and 3D lines from the image. We assume a set of M calibrated pin-hole cameras

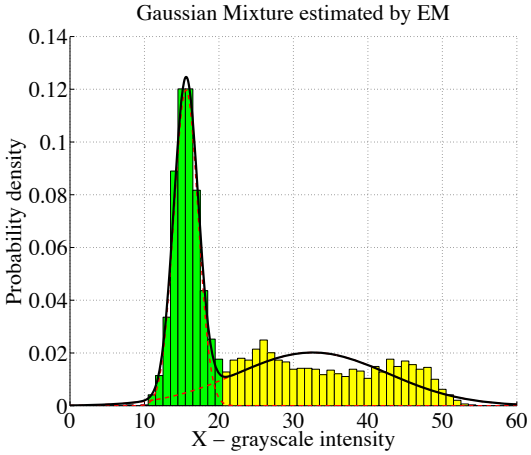
$$\lambda_j^i \begin{bmatrix} u_j^i \\ v_j^i \\ 1 \end{bmatrix} = K_i \begin{bmatrix} R_i & C_i \end{bmatrix} \begin{bmatrix} X_j' \\ 1 \end{bmatrix} \qquad i = 1 \dots M \tag{6.24}$$

with known *intrinsic* parameters K_i and *extrinsic* parameters (R_i, C_i) (see Appendix A). X_j' denotes the jth 3D point in our fly model with respect to the fixed frame, and (u_j^i, v_j^i) are the pixel coordinates of this point in camera i. In order to create correspondences between the model and image silhouette features in a given camera view, the model is first projected using (6.24) to produce the set of 2D points corresponding to 3D points on the model surface. A binary image, $I_{H(p)}$, is produced and the boundaries locations, $\mathbf{x}_j^i$ are extracted. Next, a closed B-spline curve is fit to the discrete boundary $\{\mathbf{x}_j^i\}$ to calculate the normal vector $\mathbf{n}_j^i$ at each boundary location. For each point $\mathbf{x}_j^i$, a search



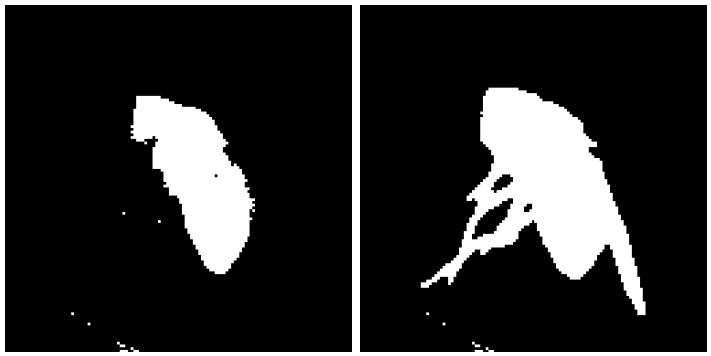
(a) Histogram of fly pixels calculated from background subtraction in over 200 frames across 3 different camera views. The characteristic bimodal shape is due to the opaque nature of the fly's body cuticle (lower intensity peak) versus the more translucent appendages (higher intensity peak).

(b) Typical image of *Drosophila* take off behavior during a startle response



(c) Histogram of fly pixels from (d) with estimated 2 member Gaussian mixture model. Local minima between the modes of the two Gaussians is chosen as the threshold to classify body and appendage pixels

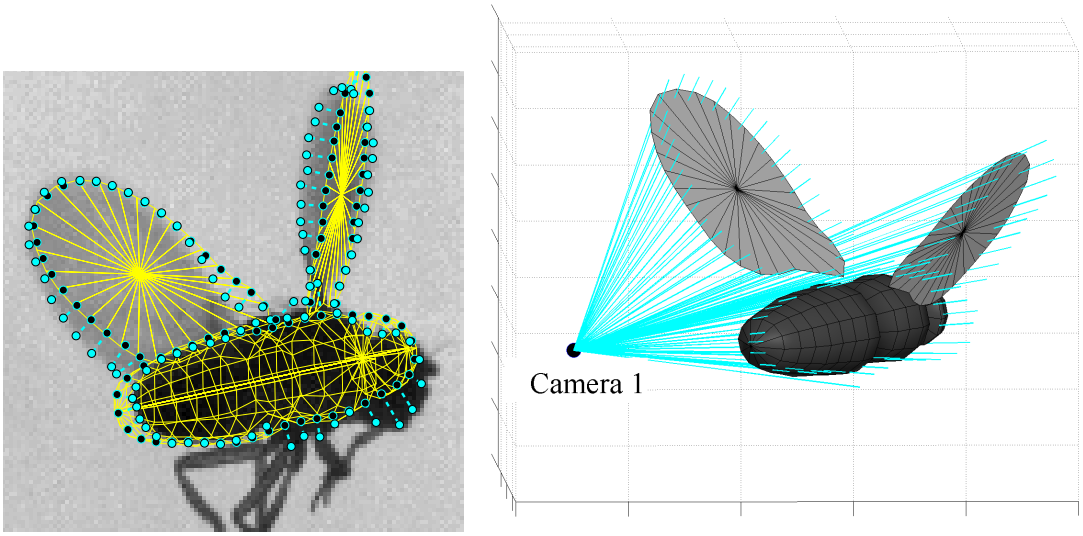
(d) Image segmented into body (green) and appendage (yellow) pixels



(e) Binary images used for silhouette feature detection (see section 6.6); I_{body} (left), I_{full} (right)

Figure 6.6: Segmentation procedure for *Drosophila* video

in the data binary image (I_{body} or I_{full}) is performed along the normal $\mathbf{n}_j^i$ to locate edges. The image intensity values are convolved with a derivative kernel and a corresponding edge position $\mathbf{e}_j^i$ is assigned to the closest location above a preset threshold (see Figure 6.7a for an illustration). Since the 3D coordinates of the projected points $\mathbf{x}_j^i$ are known, one obtains a set of correspondences between edge locations $\mathbf{e}_j^i$ and 3D model locations X_j'. These correspondences are recomputed at every iteration of the Kalman filter update, similar to the widely used iterated closest point (ICP) algorithms (see [89] for a review and fast implementation).



(a) (b)

Figure 6.7: (a) Correspondence between projected model points $\mathbf{x}_j^i$ and detected edge locations $\mathbf{e}_j^i$ shown in black and cyan, respectively. (b) The projection rays corresponding to edge points $\mathbf{e}_j^i$ are reconstructed and the distance between the 3D model and corresponding projection ray is minimized. Projection rays are only shown for 1 of 3 camera views for illustration purposes.

Next, the projection rays corresponding to the 2D edge locations are reconstructed so that the Euclidean distance between the model points and corresponding rays can be minimized. To achieve this, the projection rays are represented in *Plüker* coordinates to permit an easy calculation of the distance between a point and a line. Let $L_j^i = (n_j^i, m_j^i)$ denote the Plüker coordinates of the projection ray connecting edge point $\mathbf{e}_j^i$ with camera center C_i, where n_j^i is the unit vector that points along the line and $m_j^i = x \times n_j^i$ is the moment for a given point x along the line. Given the

camera calibration, these coordinates are calculated as

$$n_j^i = R_i^T K_i^{-1} \begin{bmatrix} \mathbf{e}_j^i \\ 1 \end{bmatrix} \tag{6.25}$$

$$n_j^i = \frac{n_j^i}{\|n_j^i\|} \tag{6.26}$$

$$m_j^i = C_i \times n_j^i. \tag{6.27}$$

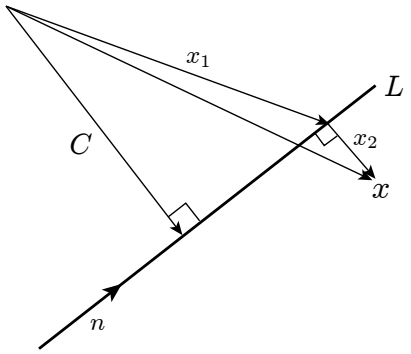


Figure 6.8: Calculating the distance, $\|x_2\| = \|x \times n - m\|$, from a point to a line represented in Plüker coordinates

A point x is incident with the line L if $x \times n - m = 0$. To calculate the distance between a point $x \notin L$ and line $L = (n, m)$ with $m = C \times n$, let $x = x_1 + x_2$ such that $x_2 \perp n$ (see Figure 6.6 for an illustration). Then

$$\|x \times n - m\| = \|x_1 \times n + x_2 \times n - m\| \tag{6.28}$$

$$= \|x_2 \times n\| \tag{6.29}$$

$$\|x_2\|, \tag{6.30}$$

thus $\|x \times n - m\|$ is an error vector that measures the distance between x and L. Hence, the state

is updated by collecting all of the correspondences across all camera views and minimizing the error

$$\min \sum_{i,j} \|(\mathbf{M}(p)X_j)_{3\times 1} \times n_j^i - m_j^i\|^2. \tag{6.31}$$

This approach for 3D pose estimation is also utilized by [86]. By concatenating the error vectors, we can rewrite this error function in the form standard to the Kalman filter, $\|z - h(p)\|^2$ where $h(p) = (\mathbf{M}(p)X)_{3\times 1} \times n$ and $z = m$. One advantage of minimizing point-to-line distances in 3D is computational savings because evaluating $h(p)$ only involves transforming 3D model points. If the error function minimized a point-to-point distance in the image plane, then $h(p)$ involves the projection of the 3D model and the calculation of its occluding contour for each function evaluation.

6.7 *Drosophila* Constraints

The *Drosophila* tracking algorithm must incorporate two constraints. The first insures that the quaternions maintain unit length, and the second constrains the body roll angle to remain symmetric between the wings. This constraint is motivated by the muscle anatomy in the fly's thorax, which actuates each wing simultaneously through deformations in the exoskeleton (Figure 6.9). The quaternion constraint has the form $\mathbf{w}_1(p_k) = Q_k/\|Q_k\| = 1$. Estimating the roll angle of the fly's body is difficult because its cylindrical shape exhibits strong rotational symmetry. In order to compensate for this fact and provide accurate estimation of the body's orientation, the roll angle of the body is assumed to be symmetric with respect to the the wings. This technique is illustrated in Figure 6.10. The approach is to rotate the body about its x axis so that the z axis bisects the angle formed by the wing vectors in the body's transverse plane. The following section provides equations for the left wing only; an analogous calculation is carried out for the right wing. Let $\mathrm{R}_{Q^b} = \begin{bmatrix} \mathrm{X}^b & \mathrm{Y}^b & \mathrm{Z}^b \end{bmatrix}$ and $\mathrm{R}_{Q^b}\mathrm{R}_{Q^{lw}} = \begin{bmatrix} \mathrm{X}^{lw} & \mathrm{Y}^{lw} & \mathrm{Z}^{lw} \end{bmatrix}$ denote the coordinate axes of the body and left wing relative to the fixed frame at the current time step (the subscript k is omitted for brevity). I define $V_L = \mathrm{Y}^{lw}$ and $V_R = -\mathrm{Y}^{rw}$ as the *wing vectors* that point from the wing tip towards the wing joint.

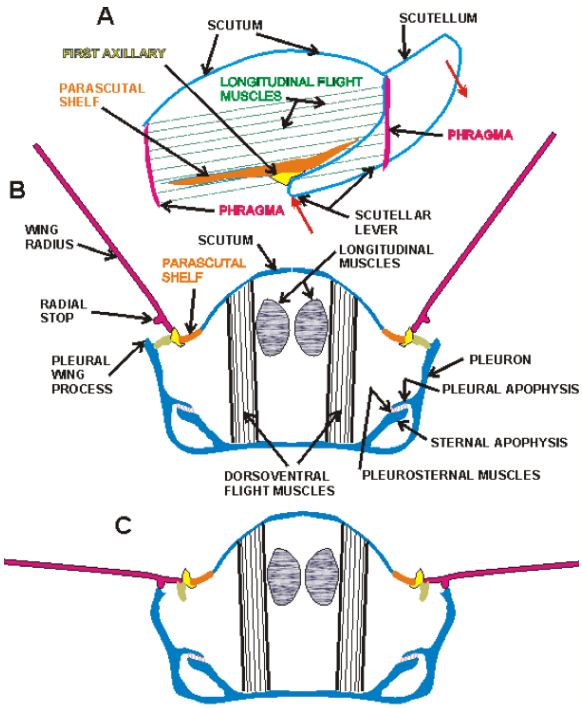


Figure 6.9: Mechanics of the wing hinge based on static analysis of dead tissue. The main motion is thought to operate as follows: longitudinal flight muscles run between the anterior and posterior ends of the thorax (A). When these muscles contract at the start of the down-stroke, the inward movement of the posterior thoras wall causes the roof (scutum) to bow upward, while the lateral thorax walls above the wings (parascutal shelves) move outward (B). Throughout the downstroke, a rigid projection within the lateral wall (the pleural wing process) is thought to act as a lower wing fulcrum. At the end of the downstroke. the scutum lifts up, stretching, and thereby activating, the dorsoventral flight muscles which power the upstroke. Also during the latter part of the down-stroke, the lateral thorax walls are deformed, which is thought to store energy in the elastic exoskeleton (C). This strain energy is released as elastic recoil at the start of the upstroke. (Image courtesy Wai Pang Chan, University of Washington, Dept. of Biology)

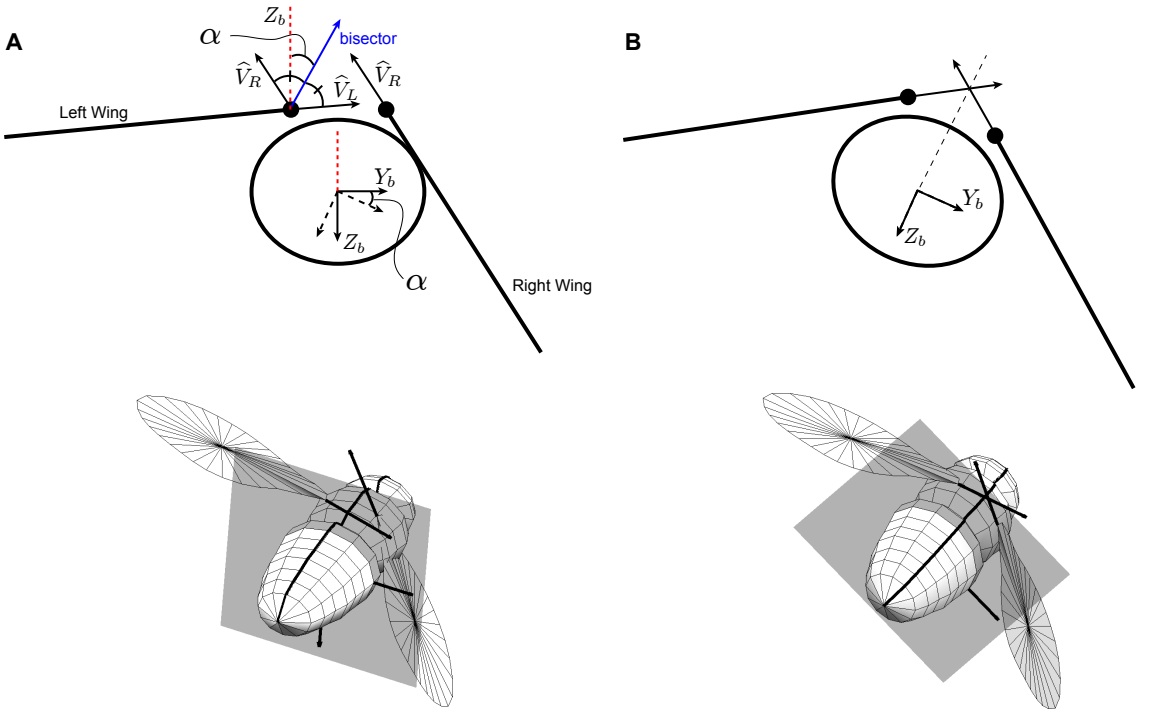


Figure 6.10: Because the roll angle of the body is unobservable from silhouette data in the images, a symmetry constraint within the transverse plane of the body must be incorporated. (A) Unconstrained estimate of the fly's pose; (top) projection of the model vectors into the transverse plane, (bottom) 3D pose with transverse plane illustrated in gray. This body configuration is highly unlikely given the biomechanics of *Drosophila*. (B) Constrained estimate after rotating body by angle α and updating joint angle vectors

The vectors Y^b and Z^b define an orthonormal basis in the transverse planar subspace of the fly's body. This is the planar subspace were the symmetry constraint is imposed. Let

$$\widehat{V}_L = \langle V_L, \mathrm{Z}^b \rangle \mathrm{Z}^b + \langle V_L, \mathrm{Y}^b \rangle \mathrm{Y}^b \tag{6.32}$$

$$= V_L^x \, \mathbf{i} + V_L^y \, \mathbf{j} \tag{6.33}$$

denote the projection of the *left wing vector* into the transverse plane. Next, $\widehat{V}_R$ is mirrored about the body's z axis and the angle between them is calculated as

$$2\alpha = \cos^{-1}\left(\left\langle \left(\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix}\right) \widehat{V}_R, \widehat{V}_L \right\rangle \right). \tag{6.34}$$

Since α is always positive, we change signs if $|V_L^x| > |V_R^x|$, which denotes a counter-clockwise rotation (Figure 6.10 is a clockwise rotation, $\alpha > 0$). The constrained body transformation is calculated by applying the coordinate transformation that encodes the roll update to the unconstrained transformation

$$\begin{bmatrix} \mathrm{R}_{Q^{b\dagger}} & T^\dagger \\ \mathbf{0}^T & 1 \end{bmatrix} = e^{\widehat{\xi}\alpha} \begin{bmatrix} \mathrm{R}_{Q^{b*}} & T^* \\ \mathbf{0}^T & 1 \end{bmatrix} \qquad \text{where } \xi = \begin{pmatrix} -\mathrm{X}^b \times T \\ \mathrm{X}^b \end{pmatrix} \tag{6.35}$$

Since our model is a kinematic chain, this roll transformation also rotated the wings to an incorrect position. Let $X_i^* \in \mathbb{R}^3$ denote the i^{th} wing point in our model at the unconstrained estimate. The constrained value of the wing joint angles is calculated as

$$\mathbf{Q}^\dagger = \underset{\mathbf{Q}}{\operatorname{argmin}} \sum_i \left(X_i^* - \mathbf{M}(\mathbf{Q}) X_i \right)^2 \tag{6.36}$$

where $\mathbf{M}$ is defined in section 6.3 (i.e., I minimize the distance between the wings points at the unconstrained state and the constrained state, holding the body transformation fixed and modifying the wing joint angles). This calculation that imposes the roll angle symmetry constraint, denoted $\mathbf{w}_2(p_k)$, is applied after the quaternion projection such that the complete projection function is

given by $p_k^\dagger = \mathbf{w}(p_k) = \mathbf{w}_2\big(\mathbf{w}_1(p_k)\big)$. This nonlinear constraint is incorporated into the Kalman filter framework using the Sigma Point transform (See Section 2.4 for details).

6.8 Experiments

I tested the algorithm on video sequences of 3 day old female *Drosophila* filmed at 6,000 fps with a shutter speed of 50 μs. The video was recorded from 3 high-speed cameras (Photron Ultima APX, San Diego, CA) that captured orthogonal views at a resolution of 512 x 512. Flies entered the recording volume by crawling up a glass pipette, and the cameras were focused on the pipette tip to maximize the resolution at the start of take off. Once airborne, after ascending a few body lengths, the flies typically became out of focus in one or more cameras. The sequences are divided in two groups; *voluntary* take off (flies permitted to remain on pipette undisturbed until they flew away) and *escape* take off (flies are startled with a looming obstacle on a collision course towards it). For each video sequence, I manually initialized the geometric model to the first frame according to Section 6.3. The database of training samples used for prediction according to Section 6.4 consists of 380 samples from a voluntary take off and 111 samples from an escape take off. Initially the training samples are captured manually, then they are replaced with the estimated values of our tracking algorithm after successfully tracking the sequence. These results demonstrate the strength of the algorithm in tracking complex flight maneuvers of *Drosophila*, even in the presence of misleading visual information.

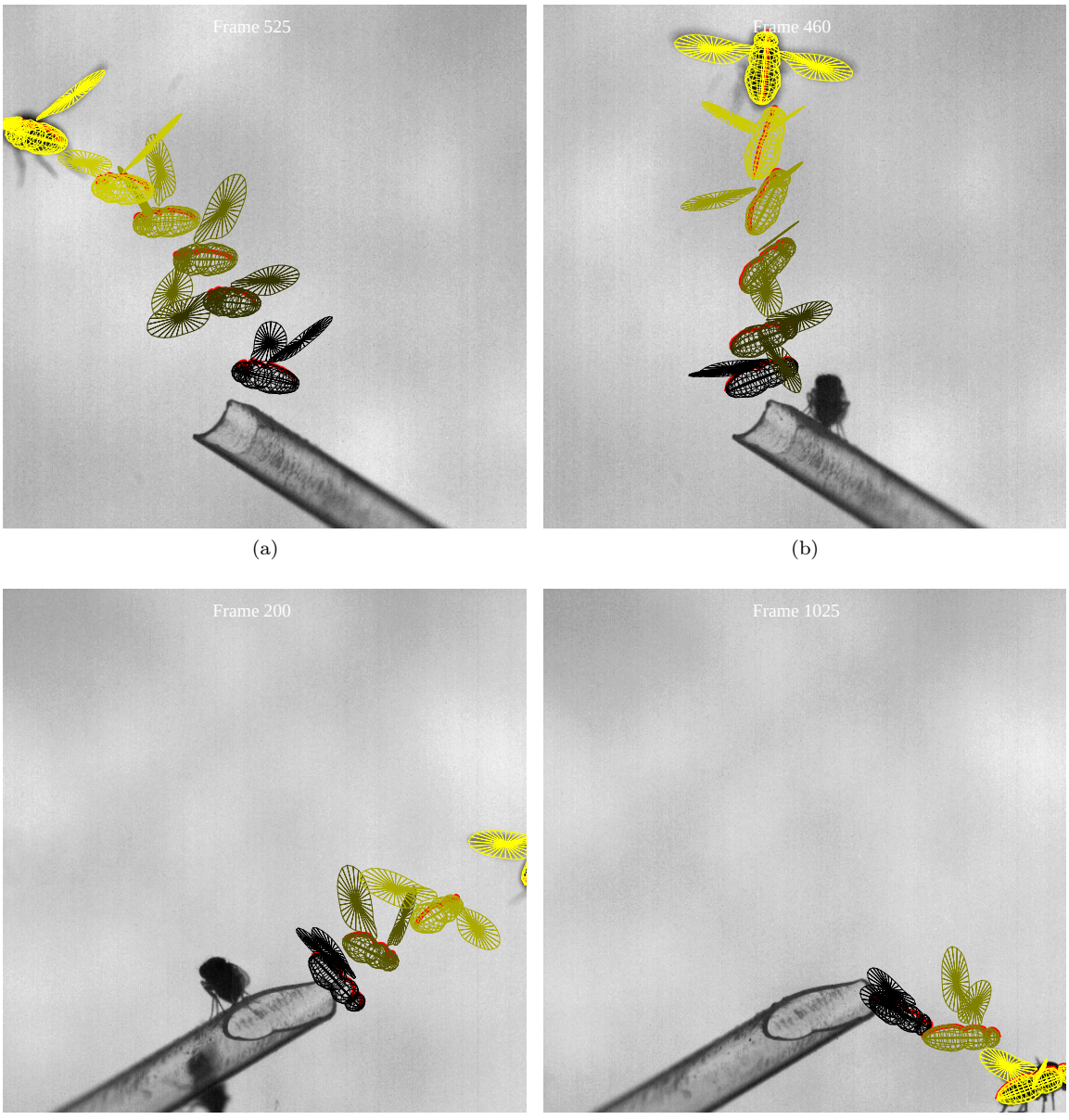
6.8.1 Voluntary Take Off

The video initially tracked and used as the "voluntary" part of our training database as shown in Figure 6.11. The estimated location of the model is plotted at particular time intervals to show the gross trajectory of the fly from one of the three camera views. This figure illustrates a steady and controlled takeoff where the fly turns sharply to the right after becoming airborne. However, there is large variability in the trajectories followed by *Drosophila* during voluntary take off. In Figures 6.12a and 6.12b, the fly undergoes large roll angle rotations at the initial stages of take off before uprighting

Figure 6.11: Tracking results of *Drosophila* using the proposed algorithm. Estimated location of model is plotted every 50 frames (i.e., 8.3 ms) from 1 of 3 camera views. The estimated states from this sequence are used in the training set for subsequent videos. Red line indicates dorsal edge.

itself to regain a level orientation. The initial thrust provided by the legs may not be applied directly to its center of mass, thus inducing some initial roll motion. The constraints described in Section 6.7 provide sufficient information to accurately track the correct roll displacement of the fly during these unsteady take offs. The direction of takeoff is also highly variable, where sometimes, the fly directs itself downward instead of upwards, perhaps to explore a region of the environment it finds interesting (Figure 6.12d).

To determine the accuracy of the proposed method, I compared body pose estimates in six videos with those reported in [16] that were captured manually using the customized software described in Section 6.3. For the manually tracked data, a reduced body model was fit to the images typically every 5 frames and a spline was used to smoothly interpolate the location of the body model at intermediate frames. This approach served to partially decrease the labor-intensive nature of manually digitizing points and to remove some of the variance attributed to human fitting by providing temporal smoothness. Figure 6.13 demonstrates that our automated method is able to achieve estimates that are comparable with human visual inspection. In reality, there are no "ground truth" estimates for the *Drosophila* data sets presented here. Whereas human motion tracking systems are typically



(a) (b)

(c) (d)

Figure 6.12: Tracking results of various other voluntary sequences. Estimated location of model is plotted at various intervals to illustrate the gross trajectory of the fly from 1 of 3 camera views.

able to measure their performance against a marker-based system, the tiny size of *Drosophila* only permits visual measurements subject to human interpretation.

Thus, the errors reported are differences between the automated tracking algorithm and manual human tracking. In some cases, it may be possible for the algorithm to achieve greater accuracy than human. The algorithm is typically able to estimate the location of body center of mass within 5% of the body length, an absolute distance that is on the order of 0.1 mm. Body orientation is also estimated well. As expected, the roll angle exhibits the largest deviations and variance due to the greater uncertainty associated with rotations about a highly symmetric axis. Figure 6.13B is the time trace of the body orientation and translation for the video sequence with the largest error in roll angle. Within this video sequence, I visually inspected two locations (C & D) where the roll angle exhibited large errors after the fly had begun take off. Based on subsequent visual inspection, it appears that the human estimates were more accurate in C, while the automated estimates provide slightly improved accuracy in D. Both display the reduced body frame model used for manual fitting. The long axis indicates the head and tail locations, and the raised "T" junction indicates the approximate wing joint locations and provides the visual cue used to determine roll angle. Overall, these results indicate that the algorithm will be useful in practical application because it achieves estimates comparable to human interpretation, while significantly decreasing the labor involved in measuring such important kinematic data.

6.8.2 Escape Take Off

I also tested *Drosophila* video with more complex motion exhibited by the organism during an escape response. Here, the fly primarily uses its legs to provide the initial thrust needed to escape the looming threat, while the wings are initially kept tucked towards the rear. This behavior provides the extremely fast initial accelerations needed to escape, but also generally results in more unstable motion where the fly tumbles and flips around (Figure 6.14a). In addition, these fast maneuvers can cause significant wing deformations that are not captured by our current rigid model (see Figure 6.15a). However, given the multiple camera views and scaled motion priors, the algorithm is able to

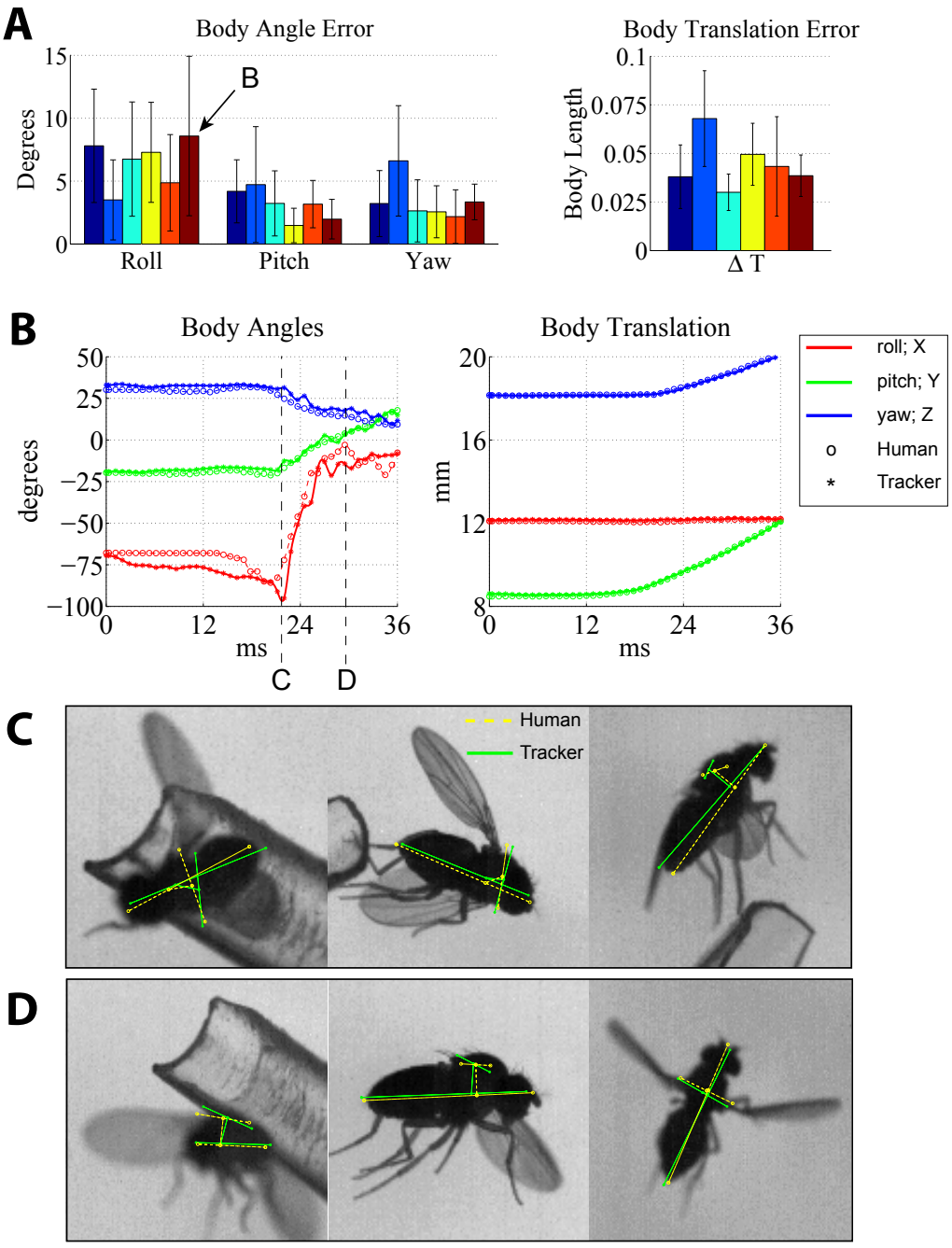


Figure 6.13: Performance metrics for the proposed algorithm. Only body pose is compared because human-tracked wing motion is unavailable. (A) RMS deviations between the human estimates and our tracker for body orientation and translation. Each bar represents a separate video sequence, and error bars indicates one standard deviation. Strong performance is achieved compared with manual human estimates, however, the roll angle demonstrates the greatest deviation and variance, as expected due to the symmetric nature of the fly's body. (B) The video sequence with largest roll error and variance is explored. The time steps indicated by (C) and (D) show the largest deviations in roll angle once the fly has taken off. From visual inspection, the human estimate in (C) appears more accurate than the algorithm's estimate, while in (D), the algorithm appears to provide a better estimate and more accurate roll angle.

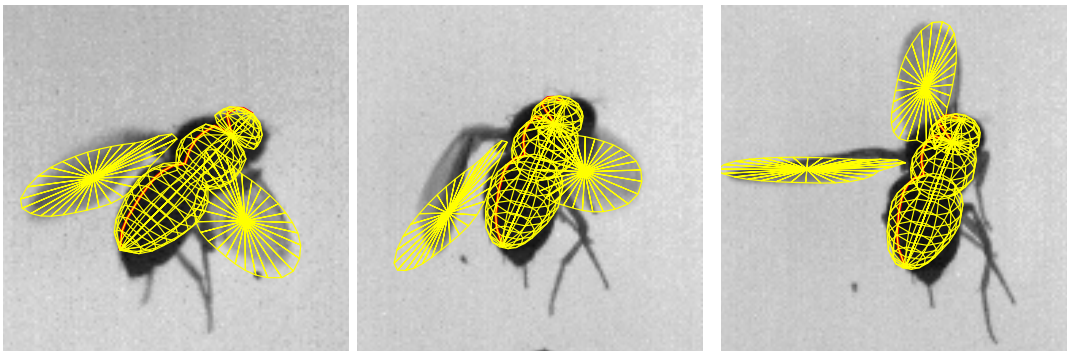
continue tracking and provide good estimation once the wing assumes a less deformed configuration (Figure 6.15b). The primary failure mode of the current algorithm is shown in Figure 6.8.2, where the right wing of the fly is flipped. The frame is taken from an upstroke of the wing path, so the right wing should be undergoing supination like the left wing. Instead, the leading edge is facing downwards as if during a downstroke. This failure only seems to occur during the escape maneuvers when complicated body motions self occlude one of the wings in one or more camera views. Despite the strong motion prior, this causes the state posterior distribution to become multi modal. In addition, the incorrect orientation of the right wing could be caused by the inaccuracies in the body orientation estimate, which is primarily due to inaccuracies in the body shape (length, width, deforming centerline axis, etc.).

(a) (b)

Figure 6.14: Tracking results of escape response sequences. Estimated location of model is plotted at various intervals to illustrate the gross trajectory of the fly from 1 of 3 camera views. The fly exhibits more fast and complicated motion to avoid collision with a looming target.

6.9 Discussion

A practical model-based visual tracking algorithm that estimates the 3D motion of *Drosophila* from multiple camera angles was presented. The algorithm uses a local optimizer and relied on prior



(a) (b)

Figure 6.15: (a) During escape maneuvers, the fly's wing can undergo large deformations that are not captured by our current rigid body model (right). In other camera views, this deformation is not apparent (left). (b) Despite this large error, the algorithm does not lose track and is able to continue successful estimation.

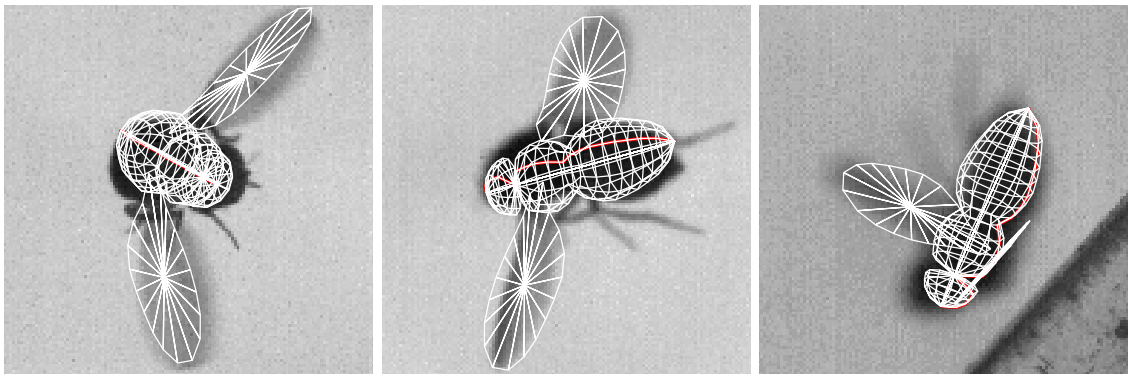


Figure 6.16: The primary failure mode of the tracking algorithm. (Left) The fly is facing towards the camera during an upstroke of the wing motion. The left wing (top) is in the proper configuration, but the right wing (bottom) is flipped in the wrong orientation (pronation instead of supination).

knowledge about the motion to keep the estimates close to the true solution. By simply matching motion patterns in a training set, the approach provided accurate prediction in video sequences containing multiple types of behaviors (voluntary vs. escape take off). In addition, a statistically sound method of incorporating state constraints within the Kalman filter framework was applied. The use of unit quaternions to model the complex wing motions, and the unobservability of the body's roll angle forced the incorporation of state constraints. Reparamterizing the model to eliminate the need for constraints is not possible in this intended application. Promising results were achieved that have comparable accuracy to manual-based human digitization. This approach enables biologists to analyze much larger datasets than possible at present. As a result, the tracking scheme will accelerate insight into the flight behavior and sensory-motor control system of many species of flies.

Chapter 7

Conclusions

Without question, automated phenotyping and behavior analysis will continue to be an important area of research in the years to come. It represents the limiting step in many fields of biological research because of the vast amounts of data that must be analyzed. This thesis has advanced the state of the art in automated behavior analysis by developing visual tracking techniques for nematodes, zebrafish, and fruit flies. These visual tracking algorithms function at a high spatial resolution and measure detailed kinematic variables needed to support current biological studies. In Chapter 4, I developed the first system to automatically track planar fish motion from a top view. This algorithm is currently being utilized to determine the effects of vertebrae stiffness on swimming performance during ontogeny. Chapter 5 presents a method for estimating the motion of worms during complex mating behavior, while Chapter 6 demonstrates the 3D tracking of fruit flies during complex flight maneuvers. Although these results demonstrate promising achievements, there are two key areas where future work is needed: (1) increased sophistication and robustness of the tracking algorithms and (2) application of the algorithms to important biological questions so as to validate and refine them as methods.

Improvements can be made in various components of the tracking algorithm to greatly improve its utility for use in behavioral genetics. Some of these improvements are specific to a particular organism. For instance, the current geometric fly model can be refined to more accurately represent the organism's shape. While the current model assumes rigid body motion of a fixed body with rotating wings, the actual fly exhibits wing bending, non-stationary wing joints, body bending,

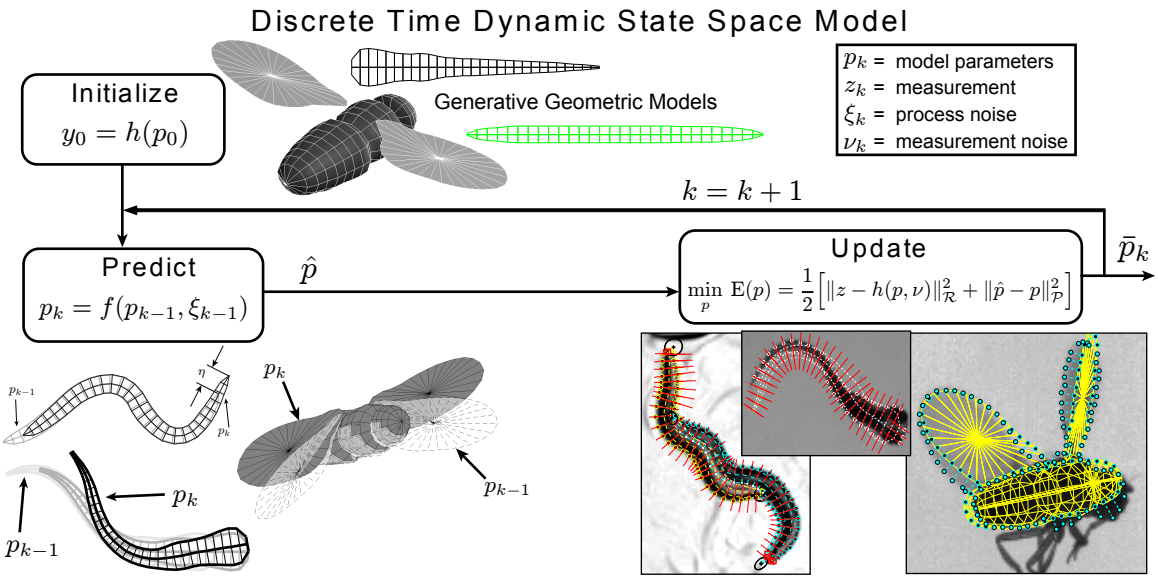


Figure 7.1: Summary of the results of this thesis. I develop a general framework for model-based animal tracking using a discrete time dynamic state space model (DSSM). A geometric model is designed and parameterized by a finite set of parameters, p. After initializing the model to the current image sequence, the pose of the model in the current image frame is predicted based on its location in the previous frames. This predicted pose is updated using measurements from the image (e.g., edge locations). This process is repeated until the entire image sequence is tracked, yielding time varying kinematic parameters of the organism's movement. This approach gives the tracking algorithm a modular design where different geometric, motion, and measurement models can be "plugged in" whenever the experimental parameters are changed (e.g., organism, lighting, etc.).

and head motion during various behaviors. Extending the model to accurately represent the body and wing deformations present in the actual fly will require additional mathematical modeling, and techniques to estimate these "deformation" parameters in addition to the rigid body motion. For the worms and fish, a quantitative analysis should be performed to determine the minimal number of bending modes necessary to capture the motion. Utilizing the most compact set of parameters to model the motion will increase the robustness and efficiency of the algorithms.

Other improvements are general and apply to all visual motion tracking algorithms. For instance, I primarily utilize the organism's silhouette produced in high-contrast images to update its location. However, the silhouette alone may not provide sufficient information to determine the correct posture when strong occlusions are present. Instead, a more advanced observation model/image likelihood measure that includes image texture and/or optical flow (i.e., image velocity field) will be important to make tracking robust across different experimental conditions. In addition, recent advances in

graphics processing unit (GPU) computing could potentially increase the high throughput nature of the algorithms. All of these methods will continue to be developed with the goal of not only applying them to the three organisms discussed here, but also to other organisms whose time-varying visual features can provide useful information. This will continue to support the development of useful and practical technology for automated behavior capture and screening across different species.

In addition, other nonlinear estimation techniques should be explored. For all three organisms, I utilized a Sigma Point Kalman filter (SPKF) to estimate the state variables of the animal model. However, the SPKF assumes that the state variables follow a normal distribution. As explained in Section 2.2, this is analogous to solving for the local optimal solution and choosing the single "best" hypothesis at each time frame. The normal distribution assumption of is often violated in various tracking scenarios, so many researchers have resorted to sequential Monte Carlo (SMC) techniques [34] to estimate the state variables. These techniques directly model the state probability distribution as a weighted collection of discrete samples (typically a few thousand). Because of this general model, the state probability distribution can become multi-modal, which has the effect of propagating multiple hypotheses at each time frame. Although exciting results were achieved with the SPKF, when SMC methods will provide improved performance remains an open question. The two primary design parameters in an animal motion tracking system are accuracy and speed. Thus, in-depth analysis is needed to determine when the increased robustness of SMC will provide greater accuracy than the SPKF, given that it is more computationally expensive.

Finally, the modular design of the algorithm must be maintained so that it can be widely applicable across different organisms and experimental setups (Figure 7). The system must function as a toolbox, where the researcher can choose which tools are appropriate to a given experiment. This is the real advantage in my approach that will permit rapid advancement in scientific knowledge with little re-engineering whenever the experimental parameters are changed (e.g., organism, lighting, etc.). This will provide a better "impedance match" between the molecular tools and behavioral assays currently available. With the current collaborations established with researchers studying nematodes, zebrafish, and fruit flies, I anticipate a very broad impact to biological science given the

wide use of these genetic model organisms.

Bibliography

[1] Geneticist seeks engineer: must like flies and worms (editorial). *Nature Methods*, 4(6):463–463, 2007.

[2] Y. Abdel-Aziz and H. Karara. Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry. In *Proceedings of the Symposium on Close-Range Photogrammetry*, pages 1–18, Falls Church, VA, 1971. American Society of Photogrammetry.

[3] D. L. Altshuler, W. B. Dickson, J. T. Vance, S. P. Roberts, and M. H. Dickinson. Short-amplitude high-frequency wing strokes determine the aerodynamics of honeybee flight. *Proceedings of the National Academy of Sciences*, 102(50):18213–18218, 2005.

[4] P. Andrews, H. Wang, D. Valente, J. Serkhane, P. P. Mitra, S. Saar, O. Tchernichovski, and I. Golani. Multimedia signal processing for behavioral quantification in neuroscience. In *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia*, pages 1007–1016, New York, NY, USA, 2006. ACM Press.

[5] P. Bang, P. Yelick, J. Malicki, and W. Sewell. High-throughput behavioral screening method for detecting auditory response defects in zebrafish. *J. Neurosci. Meth.*, 118(2):177–187, August 2002.

[6] M. Banghart, K. Borges, E. Isacoff, D. Trauner, and R. H. Kramer. Light-activated ion channels for remote control of neuronal firing. *Nat Neurosci*, 7(12):1381–1386, 2004.

[7] B. M. Bell and F. W. Cathey. The iterated kalman filter update as a gauss-newton method. *IEEE Trans. on Automatic Control*, 38(2):294–297, 1993.

[8] A. Blake and M. Isard. *Active Contours*. Springer, 1998.

[9] R. Blaser and R. Gerlai. Behavioral phenotyping in zebrafish: Comparison of three behavioral quantification methods. *Behavior Research Methods*, 38:456–469, 2006.

[10] M. A. Bogue and S. C. Grubb. The mouse phenome project. *Genetica*, 122(1):71–74, Sept. 2004.

[11] K. Branson and S. Belongie. Tracking multiple mouse contours (without too many samples). In *CVPR*, pages 1039–1046, 2005.

[12] C. Bregler, J. Malik, and K. Pullen. Twist based acquisition and tracking of animal and human kinematics. *IJCV*, 56(3):179–194, Feb. 2004.

[13] T. Brox and J. Weickert. Level set segmentation with multiple regions. *IEEE Transactions on Image Processing*, 15(10):3213–3218, October 2006.

[14] S. Budick and D. O'Malley. Locomotor repertoire of the larval zebrafish: swimming, turning and prey capture. *J Exp Biol*, 203(17):2565–2579, 2000.

[15] H. A. Burgess and M. Granato. Modulation of locomotor activity in larval zebrafish during light adaptation. *J Exp Biol*, 210:2526–2539, 2007.

[16] G. Card and M. Dickinson. Performance trade-offs in the flight initiation of drosophila. *J Exp Biol*, 211(3):341–353, Feb. 2008.

[17] J. Carranza, C. Theobalt, M. A. Magnor, and H.-P. Seidel. Free-viewpoint video of human actors. *ACM Trans. Graph.*, 22(3):569–577, 2003.

[18] T. Chan and L. Vese. Active contours without edges. *IEEE Trans. Image Processing*, 10(2):266–277, 2001.

[19] D. Cremers. Dynamical statistical shape priors for level set based tracking. *PAMI*, 28(8):1262–1273, August 2006.

[20] D. Cremers, S. J. Osher, and S. Soatto. Kernel density estimation and intrinsic alignment for shape priors in level set segmentation. *Int. J. of Computer Vision*, 69(3):335–351, September 2006.

[21] D. Cremers, M. Rousson, and R. Deriche. A review of statistical approaches to level set segmentation: integrating color, texture, motion and shape. *International Journal of Computer Vision*, 72(2):195–215, April 2007.

[22] D. Cremers and S. Soatto. A pseudo-distance for shape priors in level set segmentation. In *IEEE Int. Workshop on Variational, Geometric and Level Set Methods*, 2003.

[23] C. Cronin, J. Mendel, S. Muhktar, Y. Kim, R. Stirbl, J. Bruck, and P. Sternberg. An automated system for measuring parameters of nematode sinusoidal movement. *BMC Genetics*, 6(1):5, Feb. 2005.

[24] S. Dambreville, Y. Rathi, and A. Tannenbaum. Shape-based approach to robust image segmentation using kernel pca. In *CVPR*, pages 977–984, 2006.

[25] S. Dambreville, Y. Rathi, and A. Tannenbaum. Tracking deformable objects with unscented kalman filtering and geometric active contours. In *ACC*, 2006.

[26] A. G. Davies, J. T. Pierce-Shimomura, H. Kim, M. K. VanHoven, T. R. Thiele, A. Bonci, C. I. Bargmann, and S. L. McIntire. A central role of the bk potassium channel in behavioral responses to ethanol in c. elegans. *Cell*, 115(6):655–666, Dec. 2003.

[27] M. de Bono and C. Bargmann. Natural variation in a neuropeptide y receptor homolog modifies social behavior and food response in c. elegans. *Cell*, 94:679–689, 1998.

[28] J. De Geeter, H. Van Brussel, J. De Schutter, and M. Decreton. A smoothly constrained kalman filter. *PAMI*, 19(10):1171–1177, 1997.

[29] Q. Delamarre and O. Faugeras. 3d articulated models and multiview tracking with physical forces. *Computer Vision and Image Understanding*, 81(3):328–357, Mar. 2001.

[30] J. Deutscher and I. Reid. Articulated body motion capture by stochastic search. *IJCV*, 61:185–205, 2005.

[31] M. Dickinson, F. Lehmann, and K. Gotz. The active control of wing rotation by drosophila. *J Exp Biol*, 182(1):173–189, Sept. 1993.

[32] W. Dickson, A. Straw, C. Poelma, and M. Dickinson. An integrative model of insect flight control. In *Proceedings of the 44th AIAA Aerospace Sciences Meeting and Exhibit*, 2006.

[33] P. Domenici and R. Blake. The kinematics and performance of fish fast-start swimming. *J Exp Biol*, 200(8):1165–1178, Apr. 1997.

[34] A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte-Carlo Methods in Practice*. Springer-Verlag, 2001.

[35] Z. Feng, W. Li, A. Ward, B. J. Piggott, E. R. Larkspur, P. W. Sternberg, and X. S. Xu. A c. elegans model of nicotine-dependent behavior: Regulation by trp-family channels. *Cell*, 127:621–633, 2006.

[36] E. Fontaine, A. Barr, and J. W. Burdick. Model-based tracking of multiple worms and fish. In *ICCV Workshop on Dynamical Vision*, 2007.

[37] E. Fontaine, J. W. Burdick, and A. Barr. Automated tracking of multiple c. elegans. In *IEEE EMBS*, pages 3716–3719, 2006.

[38] N. Freimer and C. Sabatti. The human phenome project. *Nat Genet*, 34(1):15–21, May 2003.

[39] S. Fry, R. Sayaman, and M. Dickinson. The aerodynamics of free-flight maneuvers in drosophila. *Science*, 300(5618):495–498, 2003.

[40] S. N. Fry, R. Sayaman, and M. H. Dickinson. The aerodynamics of hovering flight in drosophila. *J Exp Biol*, 208(12):2303–2318, 2005.

[41] L. Fuiman and P. Webb. Ontogeny of routine swimming activity and performance in zebra danios (teleostei, cyprinidae). *Animal Behaviour*, 36:250–261, 1988.

[42] A. Gelb. *Applied Optimal Estimation*. The MIT Press, 1974.

[43] W. Geng, P. Cosman, C. Berry, Z. Feng, and W. Schafer. Automatic tracking, feature extraction and classification of c. elegans phenotypes. *IEEE Trans. Biomed. Eng.*, 51(10):1811–1820, Oct. 2004.

[44] R. Gerlai. Phenomics: fiction or the future? *Trends in Neurosciences*, 25(10):506–509, Oct. 2002.

[45] K. G. GOTZ. Course-control, metabolism and wing interference during ultralong tethered flight in drosophila melanogaster. *J Exp Biol*, 128(1):35–46, Mar. 1987.

[46] C. Graetzel, S. N. Fry, and B. J. Nelson. A 6000 hz computer vision system for real-time wing beat analysis of drosophila. In *Proc. IEEE / RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 278–283, 2006.

[47] G. Hamarneh and T. McInerney. Controlled shape deformations via medial profiles. In *Vision Interface*, pages 252–258, 2001.

[48] D. G. HARPER and R. W. BLAKE. Short communication: A critical analysis of the use of high-speed film to determine maximum accelerations of fish. *J Exp Biol*, 142(1):465–471, Mar. 1989.

[49] H. Hatze. High-precision three-dimensional photogrammetric calibration and object space reconstruction using a modified dlt-approach. *Journal of Biomechanics*, 21(7):533–538, 1988.

[50] K.-M. Huang, P. Cosman, and W. Schafer. Automated tracking of multiple c. elegans with articulated models. In *IEEE ISBI*, pages 1240–1243, 2007.

[51] M. Isard and J. MacCormick. Bramble: A bayesian multiple-blob tracker. In *ICCV*, pages 34–41, 2001.

[52] K. Ito and K. Xiong. Gaussian filters for nonlinear filtering problems. *IEEE Trans. Automat. Contr.*, 45(5):910–927, may 2000.

[53] J. Jackson, A. Yezzi, and S. Soatto. Tracking deformable moving objects under severe occlusions. In *IEEE CDC*, pages 2990–2995, 2004.

[54] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

[55] S. Julier and J. LaViola. On kalman filtering with nonlinear equality constraints. *IEEE Transactions on Signal Processing*, 55:2774–2784, 2007.

[56] S. Julier, J. Uhlmann, and H. Durrant-Whyte. A new method for the nonlinear transformation of means and covariances in filters and estimators. *Automatic Control, IEEE Transactions on*, 45(3):477–482, Mar 2000.

[57] I. Kakadiaris and D. Metaxas. Model-based estimation of 3d human motion. *PAMI*, 22:1453–1459, 2000.

[58] T. Kaletta and M. O. Hengartner. Finding function in novel targets: C. elegans as a model organism. *Nature Reviews Drug Discovery*, 5:387–399, 2006.

[59] R. Kehl, M. Bray, and L. Van Gool. Full body tracking from multiple views using stochastic sampling. In *CVPR*, pages 129–136, 2005.

[60] Z. Khan, T. Balch, and F. Dellaert. Mcmc-based particle filtering for tracking a variable number of interacting targets. *PAMI*, 27(11):1805–1819, Nov. 2005.

[61] Z. Khan, T. Balch, and F. Dellaert. Mcmc data association and sparse factorization updating for real time multitarget tracking with merged and multiple measurements. *PAMI*, 28:1960–1972, 2006.

[62] Z. Khan, T. Balch, and F. Dellart. A rao-blackwellized particle filter for eigentracking. In *CVPR*, pages 980–986, 2004.

[63] V. Kyrki and D. Kragic. Tracking unobservable rotations by cue integration. In D. Kragic, editor, *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2744–2750, 2006.

[64] M. Leventon, W. Grimson, and O. Faugeras. Statistical shape influence in geodesic active contours. In *CVPR*, pages 316–323, 2000.

[65] S. Lima and G. Miesenbock. Remote control of behavior through genetically targeted photostimulation of neurons. *Cell*, 121:141–152, 2005.

[66] R. Lints and D. Hall. Handbook of c. elegans male anatomy. *WormAtlas*, page http://www.wormatlas.org/handbook/contents.htm, 2005.

[67] C. Loer and C. Kenyon. Serotonin-deficient mutants and male mating behavior in the nematode caenorhabditis elegans. *J. Neurosci.*, 13(12):5407–5417, 1993.

[68] J. MacCormick and A. Blake. A probabilistic exclusion principle for tracking multiple objects. In *ICCV*, pages 572–578, 1999.

[69] M. McElligott and D. O'Malley. Prey tracking by larval zebrafish: Axial kinematics and visual control. *Brain, Behavior, and Evolution*, 66:177–196, 2005.

[70] M. J. McHenry. Mechanisms of helical swimming: asymmetries in the morphology, movement and mechanics of larvae of the ascidian distaplia occidentalis. *J Exp Biol*, 204(17):2959–2973, 2001.

[71] I. Mikic, M. Trivedi, E. Hunter, and P. Cosman. Human body model acquisition and tracking using voxel data. *IJCV*, 53:199–223, 2003.

[72] T. B. Moeslund and E. Granum. A survey of computer vision-based human motion capture. *Comput. Vis. Image Underst.*, 81(3):231–268, 2001.

[73] T. B. Moeslund, A. Hilton, and V. Kruger. A survey of advances in vision-based human motion capture and analysis. *Comput. Vis. Image Underst.*, 104(2):90–126, 2006.

[74] R. Murray, Z. Li, and S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[75] U. K. Müller and J. L. van Leeuwen. Swimming of larval zebrafish: ontogeny of body waves and implications for locomotory development. *J Exp Biol*, 207(5):853–868, 2004.

[76] M. Niethammer and A. Tannenbaum. Dynamic geodesic snakes for visual tracking. In *CVPR*, pages 660–667, 2004.

[77] M. Nørgaard, N. Poulsen, and O. Ravn. New developments in state estimation for nonlinear systems. *Automatica*, 36:1627–1638, 2000.

[78] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, 2003.

[79] N. Paragios and R. Deriche. Geodesic active contours and level sets for the detection and tracking of moving objects. *PAMI*, 22(3):266–280, March 2000.

[80] F. Porikli, O. Tuzel, and P. Meer. Covariance tracking using model update based on lie algebra. In *CVPR*, pages 728–735, 2006.

[81] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 1992.

[82] Y. Rathi, N. Vaswani, A. Tannenbaum, and A. Yezzi. Particle filtering for geometric active contours with application to tracking moving and deforming objects. In *CVPR*, pages 2–9, 2005.

[83] Y. Rathi, N. Vaswani, A. Tannenbaum, and A. Yezzi. Tracking deforming objects using particle filtering for geometric active contours. *PAMI*, 29(8):1470–1475, 2007.

[84] D. F. Rogers. *An Introduction to NURBS: with historical perspective*. Academic Press, 2001.

[85] B. Rosenhahn, T. Brox, and H.-P. Seidel. Scaled motion dynamics for markerless motion capture. In *CVPR*, 2007.

[86] B. Rosenhahn, T. Brox, and J. Weickert. Three-dimensional shape knowledge for joint image segmentation and pose tracking. *IJCV*, 73(3):243–262, July 2007.

[87] N. Roussel, C. Morton, F. Finger, and B. Roysam. A computational model for c. elegans locomotory behavior: Application to multiworm tracking. *IEEE Trans. Biomed. Eng.*, 54(10):1786–1797, 2007.

[88] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, Dec. 2000.

[89] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *International Conference on 3D Digital Imaging and Modeling (3DIM)*, 2001.

[90] J. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.

[91] K. Shoemake. Animating rotation with quaternion curves. *SIGGRAPH Comput. Graph.*, 19(3):245–254, 1985.

[92] G. Sibley, G. Sukhatme, and L. Matthies. The iterated sigma point kalman filter with applications to long range stereo. In *Robotics Science and Systems*, 2006.

[93] H. Sidenbladh, M. J. Black, and L. Sigal. Implicit probabilistic models of human motion for synthesis and tracking. In *ECCV*, 2002.

[94] D. Simon and T. L. Chia. Kalman filtering with state equality constraints. *IEEE Trans Aerosp. Electron.*, 38(1):128–136, 2002.

[95] C. Sminchisescu and A. Jepson. Generative modeling for continuous non-linearly embedded visual inference. In *Proc. International Conference on Machine Learning*, 2004.

[96] C. Sminchisescu, C. Sminchisescu, and B. Triggs. Kinematic jump processes for monocular 3d human tracking. In B. Triggs, editor, *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–69–I–76 vol.1, 2003.

[97] C. Sminchisescu and B. Triggs. Estimating articulated human motion with covariance scaled sampling. *The International Journal of Robotics Research*, 22(6):371–391, June 2003.

[98] J. M. Snyder. *Generative Modeling for Computer Graphics and CAD*. Academic Press, 1992.

[99] J. M. Snyder and J. T. Kajiya. Generative modeling: A symbolic system for geometric modeling. In *SIGGRAPH*, pages 369–378, 1992.

[100] B. Stenger, P. Mendonca, and R. Cipolla. Model-based 3d tracking of an articulated hand. In *CVPR*, pages 310–315, 2001.

[101] T. Svoboda, D. Martinec, and T. Pajdla. A convenient multi-camera self-calibration for virtual environments. *PRESENCE: Teleoperators and Virtual Environments*, 14(4):407–422, August 2005.

[102] C. Theobalt, J. Carranza, M. Magnor, J. Lang, and H. Seidel. Enhancing silhouette-based human motion capture with 3d motion fields. In *Proc. IEEE Pacific Graphics 2003*, pages 185–193, 2003.

[103] D. H. Thorsen, J. J. Cassidy, and M. E. Hale. Swimming of larval zebrafish: fin-axis coordination and implications for function and neural control. *J Exp Biol*, 207(24):4175–4183, 2004.

[104] A. Tsai, A. Yezzi, W. Wells, C. Tempany, D. Tucker, A. Fan, W. Grimson, and A. Willsky. Model-based curve evolution technique for image segmentation. In *CVPR*, pages 463–468, 2001.

[105] A. Tsai, A. Yezzi, W. Wells, C. Tempany, D. Tucker, A. Fan, W. Grimson, and A. Willsky. A shape-based approach to the segmentation of medical imagery using level sets. *IEEE Trans. Med. Imag.*, 22(2):137–154, Feb. 2003.

[106] D. Tweed and A. Calway. Tracking many objects using subordinated condensation. In *BMVC*, pages 283–292, 2002.

[107] E. D. Tytell and G. V. Lauder. The c-start escape response of polypterus senegalus: bilateral muscle activity and variation during stage 1 and 2. *J Exp Biol*, 205(17):2591–2603, 2002.

[108] R. Urtasun, R. Urtasun, D. Fleet, and P. Fua. 3d people tracking with gaussian process dynamical models. In D. Fleet, editor, *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 238–245, 2006.

[109] D. Valente, D. Valente, H. Wang, P. Andrews, P. P. Mitra, S. Saar, O. Tchernichovski, I. Golani, and Y. Benjamini. Characterizing animal behavior through audio and video signal processing. *IEEE Multimedia*, 14(4):32–41, 2007.

[110] R. Van der Merwe and E. Wan. The square-root unscented kalman filter for state and parameter-estimation. In E. Wan, editor, *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '01)*, volume 6, pages 3461–3464 vol.6, 2001.

[111] R. van der Merwe and E. Wan. Sigma-point kalman filters for probabilistic inference in dynamic state-space models. In *Proc. of the Workshop on Advances in Machine Learning*, Montreal, Canada, June 2003.

[112] R. van der Merwe and E. Wan. Rebel:recursive bayesian estimation library. http://choosh.bme.ogi.edu/rebel/, 2004. Matlab®Toolkit.

[113] L. A. Vese and T. F. Chan. A multiphase level set framework for image segmentation using the mumford and shah model. *Int. J. of Computer Vision*, 50:271–293, 2002.

[114] J. A. Walker. Estimating velocities and accelerations of animal locomotion: a simulation experiment comparing numerical differentiation algorithms. *J Exp Biol*, 201(7):981–995, Apr. 1998.

[115] D. Weihs. The mechanism of rapid starting of slender fish. *Biorheology*, 10:343–350, 1973.

[116] C. Yang and E. Blasch. Kalman filtering with nonlinear state constraints. In E. Blasch, editor, *Proc. 9th International Conference on Information Fusion*, pages 1–8, 2006.

[117] C. Yang, R. Duraiswami, and L. Davis. Fast multiple object tracking via a hierarchical particle filter. In *ICCV*, pages 212–219, 2005.

[118] A. Yezzi, A. Tsai, and A. Willsky. A statistical approach to snakes for bimodal and trimodal imagery. In *ICCV*, pages 898–903, 1999.

[119] J. M. Zanker. The wing beat of drosophila melanogaster. i. kinematics. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 327(1238):1–18, feb 1990.

[120] T. Zhang and D. Freedman. Tracking objects using density matching and shape priors. In *ICCV*, pages 1056–1062, 2003.

Appendix A

Camera Calibration

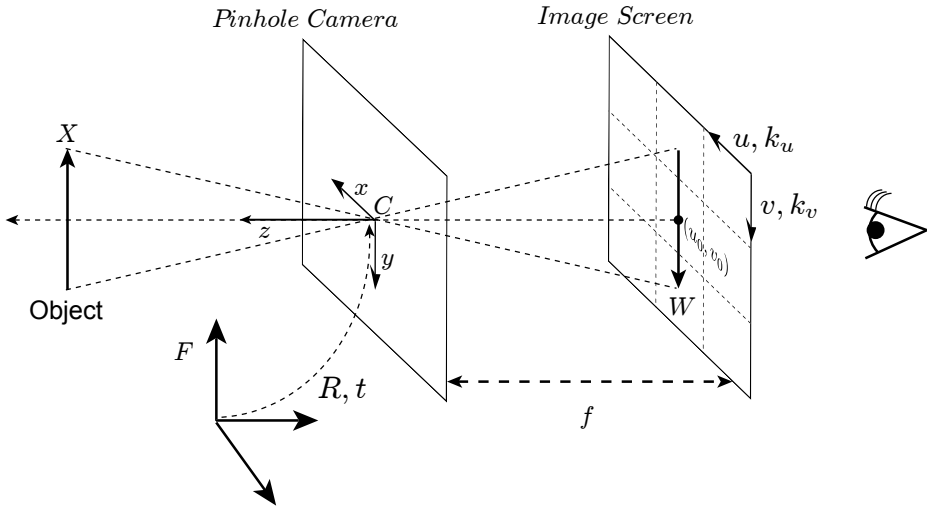


Figure A.1: Illustration of pin-hole camera model used in calibration

The most common way to model perspective projections is using the pinhole camera model, illustrated in Figure A. Here, F is a fixed reference frame, C is the optical center of the camera and origin of the camera reference frame, f is the focal length of the camera, k_u, k_v are the scaling parameters in the (u, v) directions in the image plane, and (u_0, v_0) is the location of the principal point in the image plane. We can think of recording images in a camera as projecting their 3D coordinates into 2D locations in the image plane. A simple way to model this transformation is by the Direct Linear Transform (DLT) method [2], which relies upon the colinearity of the points $X = (x, y, z)$, $C = (x_0, y_0, z_0)$, and $W = (u, v)$. Let $\mathbf{A} = \begin{bmatrix} x - x_0 & y - y_0 & z - z_0 \end{bmatrix}^T$ be the vector that points from the camera center to the object. Also, note that in the camera coordinate frame, the vector

$\mathbf{B}$ that points from the camera center to the projected point W is $\mathbf{B} = \begin{bmatrix} u-u_0 & v-v_0 & -f \end{bmatrix}^T$. Thus we see that from the colinearity condition, these vectors are scaled versions of each other, $\mathbf{B} = c\mathbf{A}, \quad c \neq 0$. However, to directly relate the coordinates, we must describe them in a common reference frame. Thus, we end up with the relation

$$\begin{bmatrix} u-u_0 \\ v-v_0 \\ -f \end{bmatrix} = c \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x-x_0 \\ y-y_0 \\ z-z_0 \end{bmatrix} \tag{A.1}$$

$$\mathbf{A}^C = R^{CF}\mathbf{A}^F \tag{A.2}$$

where R^{CF} transforms the fixed reference frame into the camera reference frame and $\mathbf{A}^C, \mathbf{A}^F$ are the vector $\mathbf{A}$ in the corresponding reference frames. (A.1) is equivalent to

$$u - u_0 = -k_u f \frac{r_{11}(x-x_0) + r_{12}(y-y_0) + r_{13}(z-z_0)}{r_{31}(x-x_0) + r_{32}(y-y_0) + r_{33}(z-z_0)} \tag{A.3}$$

$$u - u_0 = -k_v f \frac{r_{21}(x-x_0) + r_{22}(y-y_0) + r_{23}(z-z_0)}{r_{31}(x-x_0) + r_{32}(y-y_0) + r_{33}(z-z_0)} \tag{A.4}$$

after substituting for c, and k_u, k_v are scaling parameters that convert the real-world length units into pixels. These equations constitute the DLT calculation and are re-written as

$$u = \frac{L_1 x + L_2 y + L_3 z + L_4}{L_9 x + L_{10} y + L_{11} z + 1}, \qquad v = \frac{L_5 x + L_6 y + L_7 z + L_8}{L_9 x + L_{10} y + L_{11} z + 1} \tag{A.5}$$

where

(A.6)

$$[f_u, f_v] \equiv [k_u f, k_v f] \tag{A.7}$$

$$D \equiv -(r_{31}x_0 + r_{32}y_0 + r_{33}z_0) \tag{A.8}$$

$$L_1 = \frac{u_0 r_{31} - f_u r_{11}}{D} \tag{A.9}$$

$$L_2 = \frac{u_0 r_{32} - f_u r_{12}}{D} \tag{A.10}$$

$$L_3 = \frac{u_0 r_{33} - f_u r_{13}}{D} \tag{A.11}$$

$$L_4 = \frac{x_0(f_u r_{11} - u_0 r_{31}) + y_0(f_u r_{12} - u_0 r_{32}) + z_0(f_u r_{13} - u_0 r_{33})}{D} \tag{A.12}$$

$$L_5 = \frac{v_0 r_{31} - f_v r_{21}}{D} \tag{A.13}$$

$$L_6 = \frac{v_0 r_{32} - f_v r_{22}}{D} \tag{A.14}$$

$$L_7 = \frac{v_0 r_{33} - f_v r_{23}}{D} \tag{A.15}$$

$$L_8 = \frac{x_0(f_v r_{21} - v_0 r_{31}) + y_0(f_v r_{22} - v_0 r_{32}) + z_0(f_v r_{23} - v_0 r_{33})}{D} \tag{A.16}$$

$$L_9 = \frac{r_{31}}{D} \tag{A.17}$$

$$L_{10} = \frac{r_{32}}{D} \tag{A.18}$$

$$L_{11} = \frac{r_{33}}{D} \tag{A.19}$$

Given a set of N points with their known 3D and 2D locations ($N \geq 6$), the 11 DLT parameters can be calculated in a least squares sense by first rearranging (A.5) into linear system:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} x & y & z & 1 & 0 & 0 & 0 & 0 & -ux & -uy & -uz \\ 0 & 0 & 0 & 0 & x & y & z & 1 & -vx & -vy & -vz \end{bmatrix} \begin{bmatrix} L_1 \\ \vdots \\ L_{11} \end{bmatrix} \tag{A.20}$$

$$w = aL \tag{A.21}$$

and then concatenating the set of N points into an overdetermined system:

$$\begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} a_1 \\ \vdots \\ a_N \end{bmatrix} L \tag{A.22}$$

$$W = AL \tag{A.23}$$

where

$$L = (A^T A)^{-1} A^T W. \tag{A.24}$$

One problem that arises with the method above is that the 11 DLT parameters actually consist of only 10 independent unknowns ($x_0, y_0, z_0, u_0, v_0, f_u, f_v$, and 3 angles for the rotation matrix R^{CF}). Thus, one of the parameters is redundant and is coupled to the others in a nonlinear fashion. To address this problem, I use a modified approach introduced by Hatze [49] that refines the least squares estimate from (A.24) to take this coupling into account. I choose to define L_1 in terms of the other 10 parameters, although other choices are possible (see [49] for details):

$$L_1 = \frac{-(L_{11}L_2 - L_{10}L_3)(L_{11}L_6 - L_{10}L_7) + (L_{10}L_2 + L_{11}L_3)L_5L_9 - (L_2L_6 + L_3L_7)L_9^2}{(L_{10}^2 + L_{11}^2)L_5 - (L_{10}L_6 + L_{11}L_7)L_9} \tag{A.25}$$

$$L_1 = g(L_{2:11}). \tag{A.26}$$

The modified DLT parameters are then calculated by minimizing the reprojection error of (A.23) by:

$$\widehat{L}_{2:11} = \underset{L_{2:11}}{\operatorname{argmin}} \|AL - W\| \qquad \text{where, } L = \begin{bmatrix} g(L_{2:11}) & \cdots & L_{11} \end{bmatrix}^T. \tag{A.27}$$

These modified DLT parameters ensure that the projection transformation retains orthogonal coordinate frames.

A more common representation for calibrated pin-hole cameras in the robotics and computer vision community is

$$\begin{bmatrix} U \\ V \\ S \end{bmatrix} = K \begin{bmatrix} R & t \end{bmatrix} \begin{bmatrix} X \\ \\ 1 \end{bmatrix} \tag{A.28}$$

$$= P \begin{bmatrix} X \\ \\ 1 \end{bmatrix} \qquad P \in \mathbb{R}^{3\times 4} \tag{A.29}$$

with *intrinsic* parameters K and *extrinsic* parameters (R, t). $X = (x, y, z)$ is given with respect to a fixed reference frame, and the homogeneous pixel coordinates of this point in the camera are given by $u = \frac{U}{S}$ and $v = \frac{V}{S}$). Here,

$$K = \begin{bmatrix} -fk_u & 0 & u_0 \\ \\ 0 & -fk_v & v_0 \\ \\ 0 & 0 & 1 \end{bmatrix}, \tag{A.30}$$

(R, t) is a rigid body transformation from the fixed reference frame to the camera reference frame, and $C = -R^T t$ is the location of the camera center in the fixed frame. In the case of an orthographic camera model, the projection matrix has the simple form

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \\ 0 & 1 & 0 & 0 \\ \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{A.31}$$

The DLT parameters calculated can be converted into the standard intrinsic/extrinsic form by

$$C = \begin{bmatrix} L_1 & L_2 & L_3 \\ \\ L_5 & L_6 & L_7 \\ \\ L_9 & L_{10} & L_{11} \end{bmatrix}^{-1} \begin{bmatrix} -L_4 \\ \\ -L_8 \\ \\ -1 \end{bmatrix} \tag{A.32}$$

$$u_0 = \frac{L_1 L_9 + L_2 L_{10} + L_3 L_{11}}{L_9^2 + L_{10}^2 + L_{11}^2} \tag{A.33}$$

$$v_0 = \frac{L_5 L_9 + L_6 L_{10} + L_7 L_{11}}{L_9^2 + L_{10}^2 + L_{11}^2} \tag{A.34}$$

which provides the camera center and principal point location. Next, the scaling parameters are

calculated and used to solve for the rotation matrix:

$$f_u^2 = \frac{(u_0 L_9 - L_1)^2 + (u_0 L_{10} - L_2)^2 + (u_0 L_{11} - L_3)^2}{L_9^2 + L_{10}^2 + L_{11}^2} \tag{A.35}$$

$$f_v^2 = \frac{(v_0 L_9 - L_5)^2 + (v_0 L_{10} - L_6)^2 + (v_0 L_{11} - L_7)^2}{L_9^2 + L_{10}^2 + L_{11}^2} \tag{A.36}$$

$$R = \frac{1}{\sqrt{L_9^2 + L_{10}^2 + L_{11}^2}} \begin{bmatrix} \frac{u_0 L_9 - L_1}{fu} & \frac{u_0 L_{10} - L_2}{fu} & \frac{u_0 L_{11} - L_3}{fu} \\ \frac{v_0 L_9 - L_5}{fv} & \frac{v_0 L_{10} - L_6}{fv} & \frac{v_0 L_{11} - L_7}{fv} \\ L_9 & L_{10} & L_{11} \end{bmatrix} \tag{A.37}$$

The determinant of R must be checked to make sure it is 1, corresponding to a right-handed rotation. If the $\det(R) = -1$, then $R = -R$ will convert it from a left-handed to right-handed rotation.

Although the DLT formulation is uncommon in most modern techniques, it does provide a simple and practical technique for laboratory calibration. It was used by Card in the experimental study of *Drosophila* take off [16], so it was relevant to my fly tracking algorithm. Other notable calibration techniques are due to Svoboda [101] and Bouguet (`http://www.vision.caltech.edu/bouguetj/calib_doc/`), which include a lens distortion model not included in our technique.

Appendix B

B-Spline Curves

Let $P(u)$ be the position vector along a curve as a function of the parameter u. The corresponding B-spline curve is given by

$$P(u) = \sum_{i=1}^{n+1} B_i \Phi_i^k(u) \qquad 2 \leq k \leq n+1 \tag{B.1}$$

where B_i are the position vectors of the $n+1$ control polygon vertices, and Φ_i^k are the B-spline basis functions. The basis functions are defined by the Cox-de Boor recursion formula

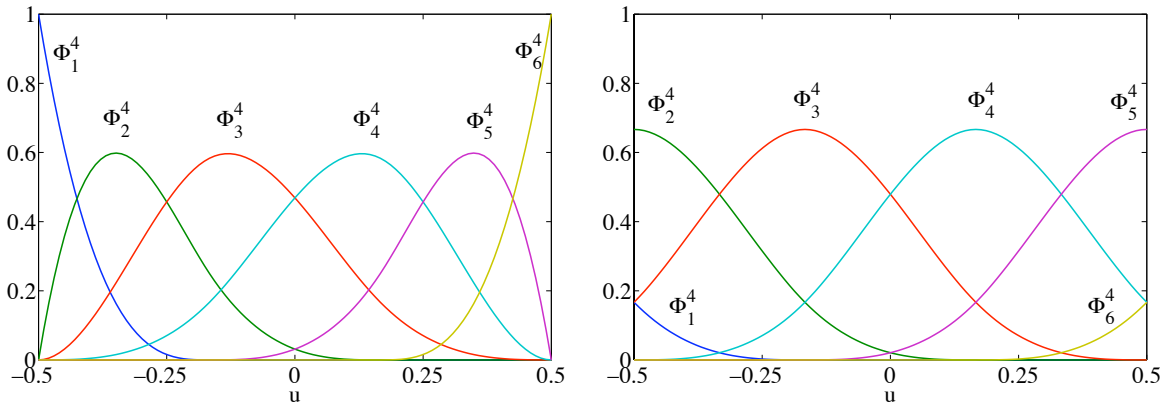
$$\Phi_i^1(u) = \begin{cases} 1 & \text{if } x_i \leq u < x_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{B.2}$$

and

$$\Phi_i^k(u) = \frac{(u - x_i)\Phi_i^{k-1}(u)}{x_{i+k-1} - x_i} + \frac{(x_{i+k} - u)\Phi_{i+1}^{k-1}(u)}{x_{i+k} - x_{i+1}}. \tag{B.3}$$

The values x_i are elements of a knot vector satisfying the relation $x_i \leq x_{i+1}$, and the basis functions have the following properties

$$\sum_{i=1}^{n+1} \Phi_i^k(u) \equiv 1 \qquad \Phi_i^k(u) \geq 0 \quad \forall u. \tag{B.4}$$



(a) (b)

Figure B.1: (a) Open uniform B-spline basis functions with $k = 4$, $n + 1 = 6$, and knot vector $[X] = \left[-\frac{1}{2} \quad -\frac{1}{2} \quad -\frac{1}{2} \quad -\frac{1}{2} \quad -\frac{1}{6} \quad \frac{1}{6} \quad \frac{1}{2} \quad \frac{1}{2} \quad \frac{1}{2} \quad \frac{1}{2}\right]$. The repeated values at the beginning and end of the knot vector cause the first and last basis functions to become more dominant. (b) Periodic uniform B-spline basis functions with $k = 4$, $n + 1 = 6$, and knot vector $[X] = \left[-\frac{9}{6} \quad -\frac{7}{6} \quad -\frac{5}{6} \quad -\frac{3}{6} \quad -\frac{1}{6} \quad \frac{1}{6} \quad \frac{3}{6} \quad \frac{5}{6} \quad \frac{7}{6} \quad \frac{9}{6}\right]$. In this case, each basis function is just a translation of the other.

Thus, the curve $P(u)$ is a polynomial spline function of order k and degree $k - 1$ on each interval $x_i \leq u < x_{i+1}$. In addition, $P(u)$ and its derivatives of order $1, 2, \ldots, k - 2$ are all continuous over the entire curve. For example, a fourth order B-spline is a piecewise cubic curve, and its first and second derivatives are continuous.

I utilize B-spline curves extensively in the geometric generative models used to model the shape of an organism. For instance, a B-spline is used to model the body profile of the zebrafish, nematode, and fruit fly. Several characteristics of the B-spline formulation make them an ideal choice for most modeling applications. First, the B-spline basis is nonglobal. That is, each control vertex B_i is associated with a unique basis function Φ_i^k, so it only affects the shape of the curve over the range of parameter values where $\Phi_i^k \neq 0$. In addition, the order of the basis functions, and hence the degree of the resulting curve, can be modified without changing the number of control vertices. For the body profiles, I typically use an open B-spline basis (Figure B.1a) because the first and last points on the curve are coincident with the first and last control point vertices. For a body profile, these first and last locations will typically be zero or another small value in order to create a closed surface (see Figure B). However, for other spline curves where it is not important to specify the end values, I use the periodic basis (Figure B.1b).

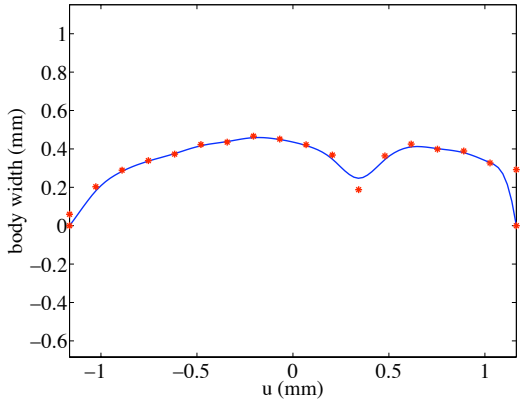


Figure B.2: Illustration of fly thorax profile constructed with B-splines, $k = 4$ and $n + 1 = 20$. The control points, B_i are shown as red dots. Here an open B-spline basis (Figure B.1a) is used because the first and last values of the function are equal to zero. This function is revolved around a centerline body axis to create a closed 3D body surface.

When implemented on a computer, the continuous B-spline functions are evaluated on a discrete grid. In this case, (B.1) can be rewritten in a convenient matrix form

$$\begin{bmatrix} P_1(u_1) \\ P_2(u_2) \\ \vdots \\ P_j(u_j) \end{bmatrix} = \begin{bmatrix} \Phi_1^k(u_1) & \Phi_2^k(u_1) & \cdots & \Phi_{n+1}^k(u_1) \\ \Phi_1^k(u_2) & \ddots & & \Phi_{n+1}^k(u_2) \\ \vdots & & \ddots & \vdots \\ \Phi_1^k(u_j) & \cdots & \cdots & \Phi_{n+1}^k(u_j) \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_{n+1} \end{bmatrix} \tag{B.5}$$

$$\mathbf{P} = \Lambda(u)\mathbf{B} \tag{B.6}$$

where u is sampled j times in a specified interval. Here, $\mathbf{P} \in \mathbb{R}^{j \times N}$, $\Lambda(u) \in \mathbb{R}^{j \times n+1}$, and $\mathbf{B} \in \mathbb{R}^{n+1 \times N}$, so (B.5) can be used to represent an N dimensional curve depending on the dimensionality of the control points $\mathbf{B}$. For my applications to generative modeling, typically $N = 1$, although I use 3D B-spline curves in the body model of fruit flies (Section 3.4). This same matrix representation can also be used to "fit" a B-spline function to a set of known data points. Given a set data points $\mathbf{D} = \begin{bmatrix} D_1 \\ \vdots \\ D_m \end{bmatrix} \in \mathbb{R}^{m \times N}$, an $m \times p$ matrix of B-spline bases, $\Lambda_{m \times p}(u)$, is calculated with parameter u_j, $j = 1, \ldots, m$. The domain of u, (i.e., $u \in [u_{min}\ u_{max}]$) and the number of control points p are specified beforehand. Then, the control points that define this B-spline are estimated by the

Moore-Penrose pseudo inverse

$$\mathbf{B} = \left(\Lambda^T_{m\times p}(u)\Lambda_{m\times p}(u)\right)^{-1}\Lambda^T_{m\times p}(u)\,\mathbf{D}. \tag{B.7}$$

This overview is based on the book by Rogers [84], so I refer the reader to this reference for more details.