Markerless deformation capture of hoverfly wings using multiple calibrated cameras

Stephen Grant Gaffney

Submitted for the degree of Doctor of Philosophy

Heriot-Watt University

School of Engineering and Physical Sciences

September 2012

This thesis introduces an algorithm for the automated deformation capture of hover-fly wings from multiple camera image sequences. The algorithm is capable of extracting dense surface measurements, without the aid of fiducial markers, over an arbitrary number of wingbeats of hovering flight and requires limited manual initialisation. A novel motion prediction method, called the 'normalised stroke model', makes use of the similarity of adjacent wing strokes to predict wing keypoint locations, which are then iteratively refined in a stereo image registration procedure. Outlier removal, wing fitting and further refinement using independently reconstructed boundary points complete the algorithm. It was tested on two hovering data sets, as well as a challenging flight manoeuvre. By comparing the 3-d positions of keypoints extracted from these surfaces with those resulting from manual identification, the accuracy of the algorithm is shown to approach that of a fully manual approach. In particular, half of the algorithm-extracted keypoints were within 0.17mm of manually identified keypoints, approximately equal to the error of the manual identification process. This algorithm is unique among purely image based flapping flight studies in the level of automation it achieves, and its generality would make it applicable to wing tracking of other insects.

Acknowledgements

I wish to thank my supervisor, Andrew Moore, for invaluable guidance throughout this project. He helped me develop ideas, kept me on track with his reliable instincts for which areas to pursue, and was always patient and encouraging.

Thanks to Yvonne Huddart for performing the camera calibration, a time-consuming task requiring careful supervision, and for advising me on how to work with calibration data. This gave me a good headstart with data analysis.

Thanks to Paulo Favaro for a discussion which provided insight into promising areas of the literature. As this project touches on so many research areas it was a great help to be pointed in the right direction.

I am very thankful for the support of Katherine Atkins. On numerous occasions she was there to provide feedback on my fledgling ideas. Her advice and encouragement kept me going during the writing process.

And a very special thanks to my parents for supporting me throughout my studies. Their assistance made it possible for me to complete the project.

ACADEMIC REGISTRY
Research Thesis Submission

Name:	Stephen G. Gaffney		
School/PGI:	School of Engineering and Physical Sciences (Mechanical Engineering)		
Version: *(i.e. First, Resubmission, Final)*	Final	Degree Sought (Award **and** Subject area)	PhD

Declaration

In accordance with the appropriate regulations I hereby submit my thesis and I declare that:

1) the thesis embodies the results of my own work and has been composed by myself
2) where appropriate, I have made acknowledgement of the work of others and have made reference to work carried out in collaboration with other persons
3) the thesis is the correct version of the thesis for submission and is the same version as any electronic versions submitted*.
4) my thesis for the award referred to, deposited in the Heriot-Watt University Library, should be made available for loan or photocopying and be available via the Institutional Repository, subject to such conditions as the Librarian may require
5) I understand that as a student of the University I am required to abide by the Regulations of the University and to conform to its discipline.

* *Please note that it is the responsibility of the candidate to ensure that the correct version of the thesis is submitted.*

Signature of Candidate:	Stephen Gaffney	Date:	14 March 2013

Submission

Submitted By *(name in capitals):*	
Signature of Individual Submitting:	
Date Submitted:	

For Completion in the Student Service Centre (SSC)

Received in the SSC by *(name in capitals):*			
Method of Submission *(Handed in to SSC; posted through internal/external mail):*			
*E-thesis Submitted (**mandatory for final theses**)*			
Signature:		Date:	

Please note this form should bound into the submitted thesis.

Updated February 2008, November 2008, February 2009, January 2011

Contents

Chapter 1

Introduction

1.1 Overview

1.1.1 Thesis goal

This thesis is an offshoot of a larger project that aims to develop aeroelastic models of insect flight for use in the development of Micro Air Vehicles. Such models would relate inertial, elastic and aerodynamic forces acting on an insect's wings during free flight to kinematic data (deformation, velocity and acceleration measurements) from the wing surface. The automated capture of deforming wing shape is the subject of this thesis.

This thesis presents an algorithm capable of automatically tracking the precise three-dimensional shape of flapping wings from multiple synchronised high-speed videos, requiring limited manual initialisation. Eristalis hoverflies were selected for study due to the distinctive patterns of venation on their wings and their hovering ability. The scope was initially limited to hovering flight, but the algorithm has been tested on a more complicated manoeuvre. The algorithm is able to handle the significant self-occlusions of the wings as they camber and twist during flight and is robust to sudden illumination changes. The success of the algorithm represents a contribution to not only the study of flapping flight, but also to the field of three-dimensional nonrigid surface tracking.

1.1.2 Data Acquisition

The data was collected by Iain D. Wallace and Simon M. Walker in Oxford in 2005, prior to the start of this project. Hoverflies were wild-caught and placed in a clear-sheet polyester cylinder, in which they were allowed to fly freely. The hovering flight was recorded by four high-speed Photron Ultima APX cameras, taking synchronised images 4000 times per second at a resolution of 1024×512 pixels. This allows for approximately 30 frames per wingbeat. An ambient light was positioned directly above the cylinder, to encourage the fly to hover below it, but laser illumination was required to light the images. Light from a 20W pulsed laser (Oxford Lasers HSI1000) was directed through four optical fibres then expanded by lenses to produce 50mm wide beams. These were shone directly at the four cameras from the opposite side of the cylinder, as indicated in Fig. 1.1. The laser specifications were chosen so as to not disturb the fly: the infrared 805 nm wavelength cannot be seen by the species, and the 20 μs pulses were not sufficient to cause the fly to overheat. The laser and cameras were synchronised using a timing pulse from one of the cameras. When the fly crossed a light gate (consisting of crossed beams from red laser pointers, directed at photodiode detectors), camera images, stored on a continuous circular buffer, were saved. Additional documentation on the apparatus can be found in [1].

The recording method described above was an improvement on earlier iterations, which involved tethering the hoverfly and marking its wings. For tethering, the fly's thorax was glued to a thin metal rod that held it in a fixed position without getting in the way of its wings. A fine-tipped black marker pen was used to make dots over the wing membranes to serve as features to track between images. Tethering has been shown to cause a "strong distortion of the stroke pattern that results in a reduction of translational forces and a prominent nose-down pitch moment" [2]. Marking, like tethering, would also have created an impediment to natural flight due to the non-negligible mass of the marker dots. The final approach allowed for completely free flight and images that were bright, well-focused, and free of motion blur.

After each hoverfly recording, additional images were taken of a calibration target as it was waved through the measurement volume, capturing it at a wide range of positions and orientations. The target consisted of a metal plate with a grid of regularly spaced circular dots of known size and spacing. This enabled the calculation of camera positions, orientations and internal parameters such as focal length and lens distortion, as described in Appendix A.

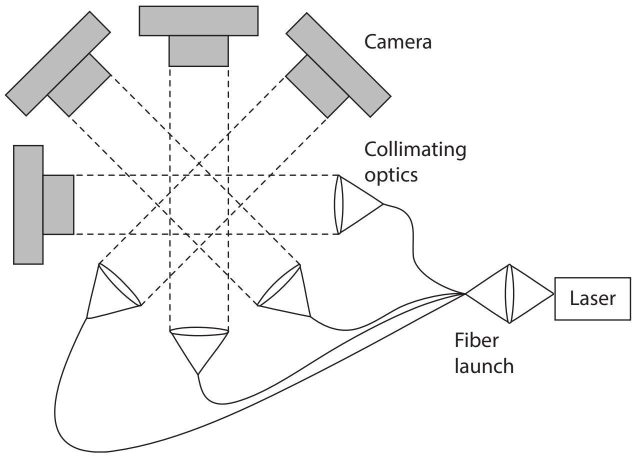


Figure 1.1: Schematic of apparatus setup, from [1].

While the design of the image capture apparatus provided solutions to the problems of fast wing motion and unpredictable body position, the recordings differ from those typically seen in tracking literature. The reliance on backlighting, for example, results in low texture wing projections that are very different from the highly textured objects commonly tracked. The use of high-speed cameras also put a limit on image resolution, further reducing texture. The number of cameras is also very small. Shape measurement and tracking algorithms often assume dozens of viewpoints as a way of handling self-occlusion, but this approach would be prohibitively expensive for high-speed cameras. Together, these properties pose challenges that make the problem unique.

1.1.3 The images

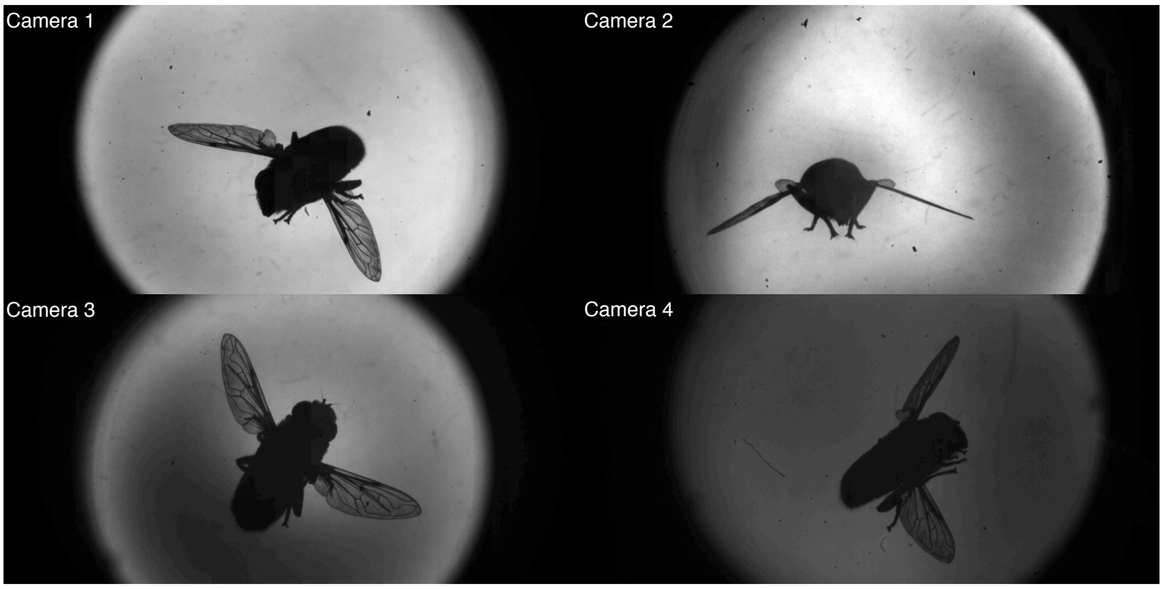


Figure 1.2: Example images from four cameras.

The laser backlighting has the effect of creating a silhouette of the fly: it appears in outline against a circle of bright light. The near-transparent membrane of each wing lets through light, revealing the veins as black bands. The intersection points of these dark veins provide the features that will be tracked between images. The edge of the wing boundary, which also stands out against the bright background, will be tracked as well. The wings, which measure 12mm in length, project to roughly 200×40 pixels. Figure 1.2 shows example images of the hoverfly at one instant during a successful recording.

Stepping through the images, several image characteristics become apparent which increase the challenge of wing tracking, illustrated in Fig. 1.3. There's a high degree of self-occlusion as the wings twist and fold, occlusion from the fly legs and abdomen, and occlusion from the silhouettes of background objects (most likely dust) that occasionally obscure wing membranes. As the wing turns to be more edge-on to the camera, the veins are increasingly obscured until the wing appears as a narrow strip of dark pixels. Parts of the wing may therefore not be visible in any views for some frames. Image sets were chosen that fit with Ellington's definition of hovering flight, with advance ratio limited to 0.1 [3], but this inter-frame body motion is noticeable. In some frames the wing partially stretches out beyond the circular region of illumination, into the dark exterior. Even in the bright region, variation in illumination (due to intensity differences in the laser pulses) results in some images being considerably darker, with low contrast. Finally, the thin hairs on the fly abdomen create a soft boundary between the fly and the background, posing a challenge to fly segmentation. Any wing tracking algorithm that can run on these images must cope with each of these issues.

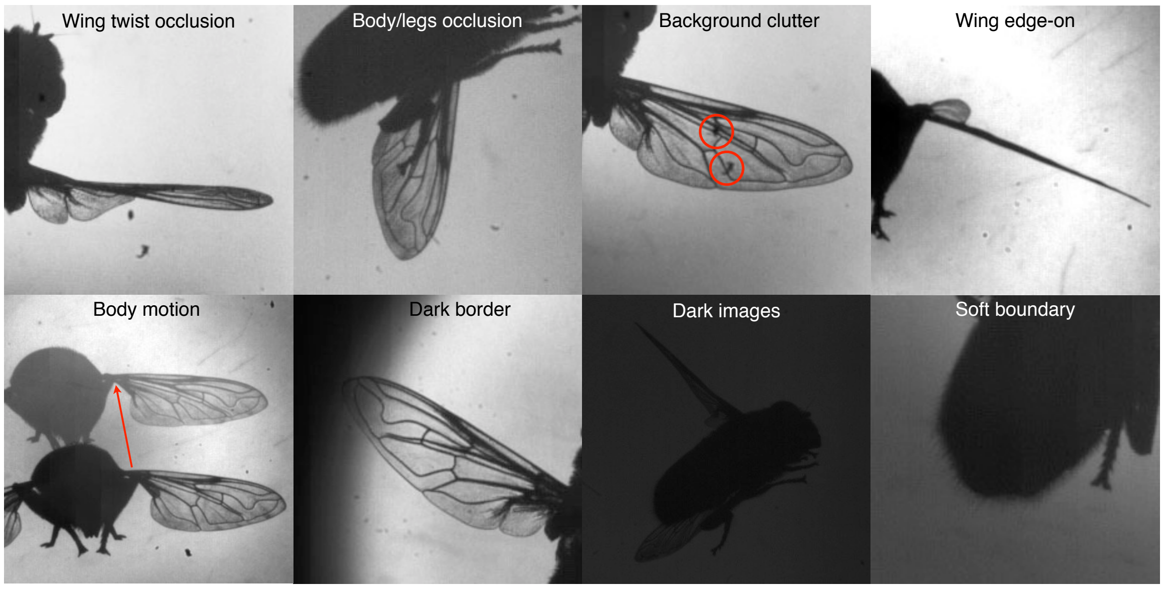


Figure 1.3: Examples of image characteristics that pose challenges to tracking.

1.1.4 Wing structure and flight properties

In order to measure wing shape from images, it is necessary to precisely identify the wing boundary and a set of interior points. Fringe projection methods for establishing interior points have been used for studying dragonfly flight [4] but were opted against here due to the transparency of the hoverfly wing membranes. An obvious alternative is to make use of the well-defined dark venation on the wings, common to many winged insects. These veins, or ‘spars’, can be divided into two groups: longitudinal veins, which radiate from the wing base, and often branching distally, and cross-veins, which connect them [5]. Veins vary in thickness and rigidity, and serve to support the wing, limiting bending and folding, in addition to their transport functions and wing tear limitation. An insect will twist the leading edges of its wings to create wing curvature in all directions nonparallel to longitudinal veins [6]. Cross-veins allow camber by being more flexible due to their thinness, lack of corrugated structure or tracheae, and weakened regions [7]. In photographs, veins can vary in opacity due to differences in composition: longitudinal veins lacking in tracheae, sometimes called secondary pseudoveins or vena spuria, can appear lighter. Such veins are found in hoverfly wings, which have a very distinctive venation pattern, as shown in Fig. 1.4.

The merging of the disciplines of wing morphology with insect flight in the early 1970s marked the beginning of experimental investigations into functional wing design [5]. These

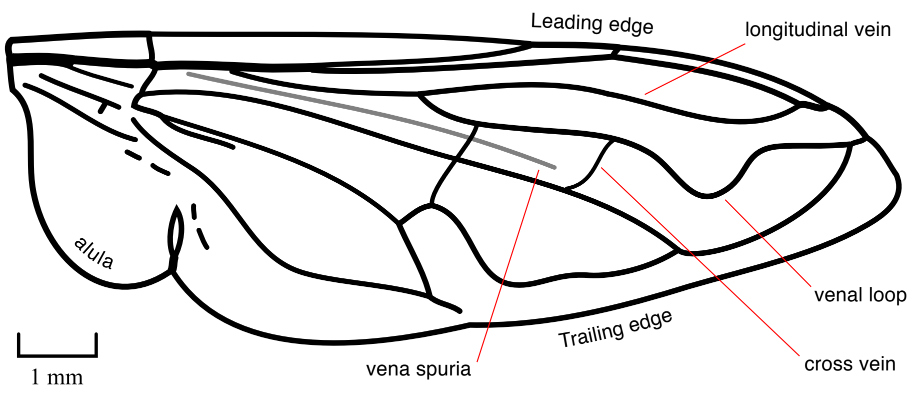


Figure 1.4: Spar structure of Eristalsis hoverfly, adapted from [6].

investigations have yielded insights into how insects' flapping wings generate the forces required to lift their body-weight (two to three times more lift than has been shown in conventional fixed wing flight studies [8]). Early tethered flight studies gave way to the high-speed photography of free-flight, pioneered by Weis-Fogh and Norberg, which in turn led to the discovery of the 'clap and fling' flight mechanism [9]. Through airflow visualisations, a wide array of complex symbiotic flight mechanisms have also been revealed [10], including wake capture, active and inactive upstrokes, and leading edge vortices. A reliable method to automatically track wing deformation from high-speed video would constitute another important step in this timeline of progress.

1.2 A review of three-dimensional tracking techniques

The field of three-dimensional tracking lacks a generalised framework. There is no general solution that will work for all objects under all viewing conditions. Instead, algorithms attempting such tracking are usually constructed heuristically, and make simplifying assumptions concerning object shape, object motion and image characteristics. The simplest approach treats objects as a collection of points, such as the fiducial markers commonly used in human motion capture systems. Alternatively, objects may be simple geometrical shapes, curves, planes or complex surfaces. Nonrigid surfaces, such as the hoverfly wings of this project, present an enormous challenge, particularly in the absence of a deformation model.

As with two-dimensional tracking, algorithms vary in their computational complexity, likelihood of error accumulation, generality of trackable objects, and robustness to occlusion, deformation and changes in illumination. They also exhibit a trade-off between accuracy and long-term stability. Some algorithms require a pre-defined model to represent 3-d structure or motion, while others use training images to construct a model. Many require landmarks to be automatically pulled out (or manually identified) and matched as 'point correspondences' to build, test or fit the model, and will vary in their robustness to false correspondences. They may examine texture information only, or discard texture information in favour of extracted shape information, or seek to fit combined shape and appearance models. These techniques will be touched upon in the next few sections, examining their strengths and weaknesses for the problem at hand.

Whatever the particular nature of the data, 3-d tracking can be broken down into a number of challenging subproblems. One subproblem is 2-d segmentation - the identification of pixels in an image occupied by the object of interest. This goes hand in hand with the subproblem of registration - finding a map between points on the object and these occupied pixels. A third subproblem is 2-d tracking - the repeated segmentation and registration of object pixels in an image sequence, giving the 2-d motion (and deformation) of the object. A fourth subproblem is the matching of object features between different views. In the following sections, approaches to these subproblems are reviewed, and divided based on whether they deal with one or multiple cameras, and one or multiple time instants, as illustrated in Fig. 1.5. A number of algorithms that combine solutions to these subproblems to attempt multi-camera 3-d tracking are also reviewed.

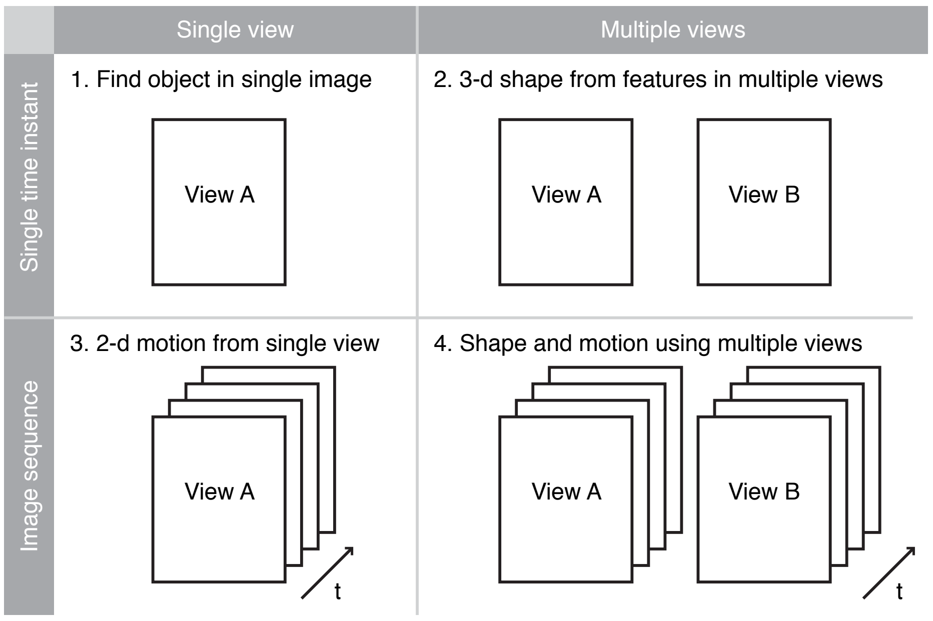

Single view
Multiple views
Single time instant
Image sequence
1. Find object in single image
2. 3-d shape from features in multiple views
3. 2-d motion from single view
4. Shape and motion using multiple views
View A
View B
t

Figure 1.5: Literature review structure illustration.

1.2.1 Single view, single time instant: object localisation

Before any object can be tracked between images, its location must be identified in an initial image. In the case of hoverfly wing tracking, we wish to know which pixels in the image are occupied by the wing (*segmentation*) and how to relate the pixel locations to wing locations (*registration*). The human visual system solves these problems very quickly: when presented with an image of a hoverfly, we effortlessly identify its wings, their orientation and landmarks on their distinctive pattern of venation (Fig. 1.4). The challenges of programming computers to solve such problems are addressed by the field of computer vision.

This section touches upon techniques from the computer vision literature as well as complementary techniques from image processing literature. Techniques discussed include those aimed at the localisation of general objects as well as those that can be tailored to localising wing-like objects.

Local affine-invariant features for object recognition

An important concept in object recognition is the 'local feature'. Local features are points on the surface of an object that can be described by texture information in a small region around them. By thinking of an object as a collection of distinctive local features, we can hope to recognise the object even when much of it is obscured. In the last decade, a number of techniques have emerged that can identify such features and characterise them in a way that allows them to be recognised despite (limited) changes in viewpoint, orientation, scale, brightness, perspective and image compression. These advances have motivated a trend in object recognition, observed by Ferrari et al [11], towards matching from example images of the object rather than 3-d object models.

By mid-2004, at least six distinct types of 'affine invariant' (or, more correctly, 'co-variant') feature detectors had been developed[1] [12]. The Harris detector, which identifies corners, can be adapted to give affine and scale invariance. Scale invariance can be achieved by calculating a 'characteristic scale' for a local feature. One way of doing this is to use the Laplacian-of-Gaussian function: a given interest point is localised at several different scales, and the maximum Laplacian-of-Gaussian response gives the characteristic scale [13]. Affine invariance for a feature point can be obtained by adapting its neighbourhood to a shape determined by the second moment matrix. This is referred to as the Harris-Laplace detector, and was developed by Mikolajczyk and Schmid in 2002 [14]. In this way, we have image patches that automatically deform with changing viewpoint, in such a way that they keep covering identical parts of a scene.

After interest points/regions have been established, a 'descriptor' is calculated for each one. This is a vector that characterises the local photometric properties of the region. There are a number of different descriptors available. Interest points can be matched by comparing their descriptors, accomplished using methods such as correlation, Euclidean distance or Mahalanobis distance. David Lowe's *SIFT* descriptor (Scale Invariant Feature Transform) is a 128-dimension vector built from thresholded image gradients[2], which has been demonstrated to be superior to the majority of descriptors [15].

A small set of feature points, once matched to a model image, can be used to find other features. Ferrari, Tuytelaars and Van Gool demonstrated this with a method they call 'image

[1]A number of different detector binaries can be obtained from the University of Oxford: http://www.robots.ox.ac.uk/∼vgg/research/affine/detectors.html#binaries

[2]Code for generating the SIFT vector and obtaining point correlations is provided by Lowe at the University of British Columbia's website: http://www.cs.ubc.ca/∼lowe/keypoints/

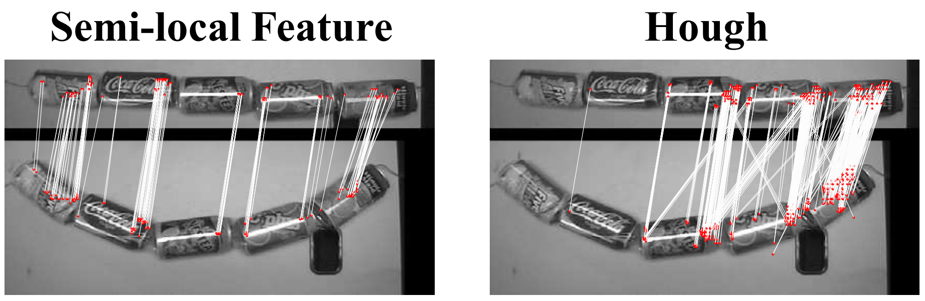


Figure 1.6: Successful shape-context matching for ‘snake of cans’, compared to Hough clustering method [17].

exploration’ [11]. A model image is divided into small regions and features are grouped into these regions. Anchoring on an initial small set of feature points, additional matches are sought in neighbouring regions, with a search method that tolerates non-rigid deformations and weeds out bad matches. Eventually, much of the object is covered in matches and the occupied image regions give an approximate object boundary.

The local matching process of invariant features can be combined with additional ‘semi-local’ constraints, such as geometric or statistical relations between points to improve matching. Tuytelaars and van Gool devised two semi-local constraints, one geometric and one photometric, which can eliminate false matches [16]. They combined these constraints with a system that exploits two types of features: one purely intensity-based and one which exploits edges around corners. A related approach is ‘shape context’ [17], which essentially captures the distribution of the relative positions of all surrounding features with respect to the current feature [18]. It does this by drawing a line from it to all other features and using a histogramming method to estimate the distribution of length and orientation for these lines. The feature is now supported by a larger region without causing much damage to its robustness, whereas enlarging the supporting pixel region around a landmark would normally reduce invariance to changes. Its effectiveness can be seen in Fig. 1.6.

Local feature matching can also be used for finding poorly textured objects with holes and tubular components. By generalising the SIFT descriptor to edges, so that features can still be identified even if on an object’s border, Mikolajczyk et al [15] created an algorithm that can recognise objects such as bicycles and badminton racquets in cluttered scenes. From only a single training image, it can create a model which can be recognised in new images through iteratively tightening geometric restrictions. Their method only works for roughly planar objects, however, and could not readily be applied to highly deforming surfaces. A related method [19] uses edges to learn the shape of an object class from multiple training images. Edge pixels are identified and grouped into segments, and adjacent pairs of these segments form the features whose geometric properties are used for matching between images - no additional texture information is used. This allows characteristic edge segments to be identified for the class, and from these, a statistical shape analysis technique

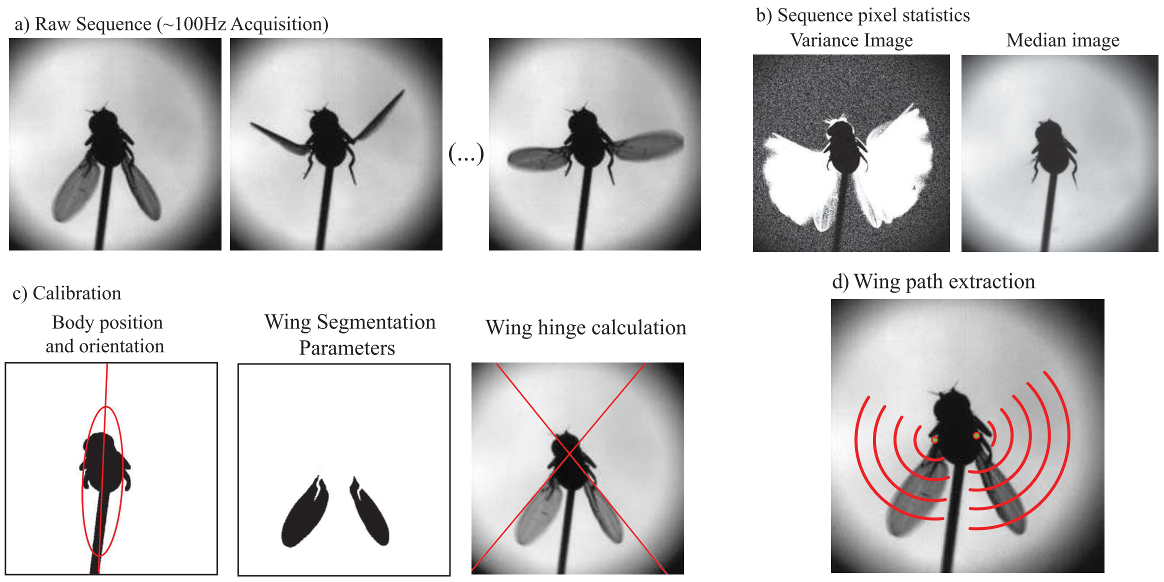

a) Raw Sequence (~100Hz Acquisition)
(...)
b) Sequence pixel statistics
Variance Image
Median image
c) Calibration
Body position and orientation
Wing Segmentation Parameters
Wing hinge calculation
d) Wing path extraction

Figure 1.7: Wing segmentation and path extraction, from [21].

[20, discussed in Section 1.2.3] can build a model of shape variations, capable of being recognised in new images.

Object extraction through image processing

Another way of approaching the problem of image segmentation for extracting an object is through image processing methods tailored to the data. For example, digital images can be enhanced to exaggerate the contrast of an object against its background. This is often the first step towards identifying its location within the image. Suitable thresholding can then create a binary image where 1s lie on the object with 0s elsewhere.

This strategy has been adopted for a real-time motion analysis of Drosophila wings [21]. A sequence of images from a single camera were analysed to segment the wings, deduce body orientation, wing hinge positions, and determine the 2-d paths swept by the wings of a tethered fly. The stages are illustrated in Fig. 1.7. First, a background image was constructed by finding the mean value at each pixel over an image sequence. Subtraction of this image (*background subtraction*) removed the effect of lighting variations between images. Morphological processing allowed the body to be selected as a large blob region after thresholding, and this was combined with a thresholded variance image to create a wing isolation mask. After applying the masks, an additional thresholding was performed (with optimal threshold calculated using the Max-Lloyd quantisation algorithm), leaving a binary image with only wing pixels 'on'. A morphological opening filled small holes caused by semi-transparent sections of wing. Wing edges were found with the Canny filter, and a Hough transform pulled out the strongest line from these edges (which 80% of the time gave the leading edge). The median of the line intersections was taken as the 'hinge position' of the wing. Finally, leading and trailing edge pixels were isolated by intersecting

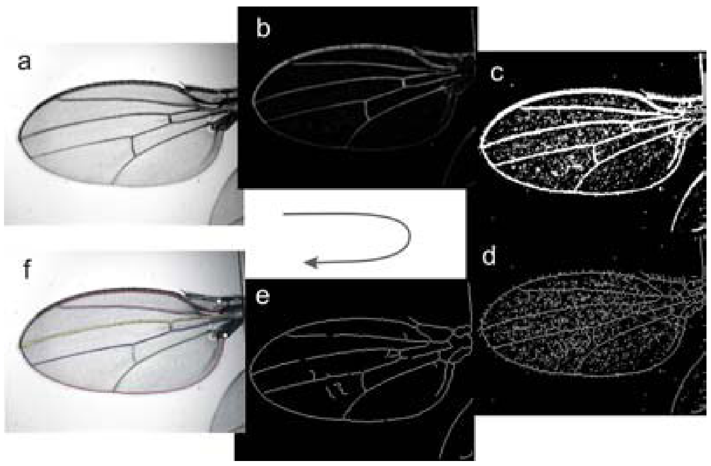


Figure 1.8: Spar extraction steps, reprinted from [22].

the wing paths (which follow concentric circles around the hinges) with the binary wing image.

A range of additional image processing techniques were used in another Drosophila study [22], producing a system of automatic image analysis for the measurement of veins and edges on the wings. Anaesthetised flies had their wings manoeuvered onto a slide to be photographed. ‘Feature extraction’ was then performed on the images, illustrated in see Fig 1.8. The image was reversed (making the background dark and the spars light); noise reduction was performed with a median filter and an ‘opening’ operation, minimising background features; the image was thresholded and then holes were filled (a standard morphological operation); and skeletonisation reduced the features to one pixel in width. The parameters of the various stages were chosen based on trial and error. This left the image in its final state, from which intersection points of the spars could be pulled out. The measurement stage involved two manually identified landmarks defining the orientation of the wing, then spar intersections were used to fit an *a priori* B-spline model, with nine curves representing the wing spars.

Localisation of vein-like structures is also studied in the context of digital retinal image analysis. Here the goal is to take an image of a retina, identify the small, circular optic disc, segment and trace the blood vessels that radiate from it, and analyse their structure for the purpose of detecting diseases involving vasculature changes. Automated vessel extraction is an active research area. The methods developed can generally be placed in one of two categories: the ‘pixel processing approach’, and ‘vessel tracking’. The former usually involves a morphological or adaptive filtering technique, followed by postprocessing such as thinning or reliability improvement through artificial intelligence methods [23]. The latter takes an initial vessel point and traces along it using local image properties. It is fast and scales well to varying image resolution.

An effective example of the pixel processing approach was described is Hoover’s ‘thresh-

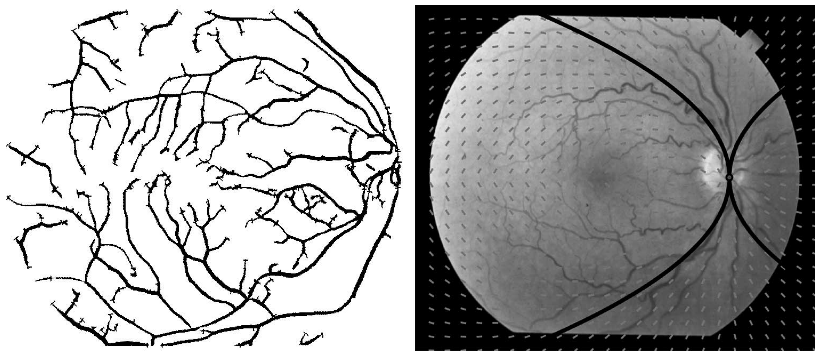


Figure 1.9: Vessels from threshold probing [24], and some model-fitting results [25].

old probing' technique [24]. This was employed in a 2004 study which relied on accurate vessel segmentation for fitting a vessel model and deducing location of the optic disc [25]. This thresholding technique combines local vessel properties with network structure properties. A special convolution filter is applied to the image, called the 'matched filter response' (*MFR*). This filter facilitates the search for regions of specific optical and spatial properties: in particular, the Gaussian-shaped curve of the greyscale cross-section of a blood vessel [26]. Twelve 16x16 pixel kernels comprise the filter to account for possible orientations of the vessels. An area of the MFR image is probed at iteratively lower thresholds. At each iteration, a test of region-based attributes is performed to determine whether to proceed to the next iteration and, finally, whether the pixels represent a vessel. This starts with a region grown around a starting pixel to contain all connected pixels above a certain threshold. Weak-MFR pieces are included if they border previously vessel-classified pieces, allowing gaps to be bridged. This segmentation technique has been used to fit a geometrical parametric model to retina images [25] (shown in Fig. 1.9)[3].

A good example of the vessel tracking of approach comes from Cree et al [27]. It builds on the same assumption that the cross-section of blood vessels can be described by a Gaussian curve. If a point on a vessel has coordinates (x,y), and the orientation of the vessel with respect to the x-axis is θ, then a new (u,v) coordinate system can be defined where u points in the same direction of the vessel and v is orthogonal:

$$\begin{aligned} u &= x\sin\theta - y\cos\theta \\ v &= x\cos\theta + y\sin\theta \end{aligned} \tag{1.1}$$

To account for tortuous vessels, a quadratic term in v is added to u to allow the vessel to

[3]A Windows executable implementation of threshold probing can be found at http://www.parl.clemson.edu/stare/probing/.

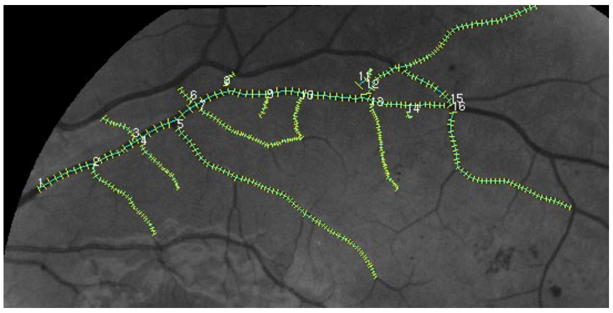


Figure 1.10: Vessels from model-based tracking [27].

flex:

$$\begin{aligned} u &= x\sin\theta - y\cos\theta + \eta v^2 \\ v &= x\cos\theta - y\sin\theta \end{aligned} \tag{1.2}$$

Here, η is a scalar describing curvature. For this coordinate system, intensity along the vessel cross-section is given by

$$I(v) = A - Be^{-(v-v_0)^2/2\sigma^2} \tag{1.3}$$

where A is background intensity, B is vessel contrast with respect to the background, σ is vessel width, and v_0 is a shift of the vessel centre in the v direction. The algorithm starts at a point on a vessel where the width and orientation are known, then a small step along the vessel is made. In a small region (whose size is based on σ) around this new point, a non-linear least squares fit of the model is made to the greyscale data. This gives a new width and orientation. Width is regulated by averaging the width found at the current step with the previous one. The process repeats, allowing 'walking' along the vessel. Once this vessel has been fully tracked, a search is performed for branches by stepping along the vessel, looking at thresholded regions of interest on either side. If components and a watershed line extend from one region of interest (ROI) to the other, then a branch is present and tracking may begin on that branch (See Fig. 1.10). This technique has proved reliable even in cases of high noise and poor contrast. One way the tracker has been initialised involves finding the outline of the optic disc (using morphological operations and the canny filter), then examining pixels on the circumference of a circle centred on the disc [23]. Dips in intensity to well below the median curve indicate likely vessels.

1.2.2 Multiple views, single time instant: static 3-d shape

The problem of deducing three-dimensional structure of an object from a single image is severely under-constrained: without the aid of prior models there are infinitely many shapes that can give rise to a particular two-dimensional projection. Having two or more cameras allows the 3-d location to be determined for any feature whose image coordinates are found in multiple views: a process called ‘intersection’. This requires the knowledge of certain camera parameters, which can be determined through standard photogrammetric calibration procedures (most commonly with the aid of a target object with precisely sized and spaced grid markings). These parameters describe the position and orientation of the camera; the focal length and distortion of the lens (radial and tangential); and sensor properties such as pixel aspect, skew and principal point. Combined, they provide the geometry of image formation (mapping world coordinates onto image coordinates), and for every point in an image they determine the ray of points in 3-d space that project onto it (called the ‘visual ray’).

In the ideal scenario of perfectly calibrated cameras with perfect feature localisation, the back-projected rays from a matched feature point will intersect each other. This intersection point is the feature’s 3-d location. It is important to note that imperfect calibration and measurement noise introduces uncertainty in the resulting 3-d localisation. This uncertainty causes the rays to be skew, and varies with the angle between the rays, as illustrated in Fig. 1.11 [4]. As the rays do not actually intersect, it is necessary to estimate a 3-d location by minimising a suitable cost function [29, Chap. 12]. This can be done by finding the 3-d point that minimises the reprojection error (i.e. minimises the separation between the input image points and the projections of the estimated 3-d point). The maximum likelihood estimate minimises the sum of the squares of these distances and can be found as a root of a degree-6 polynomial [29, p. 319]

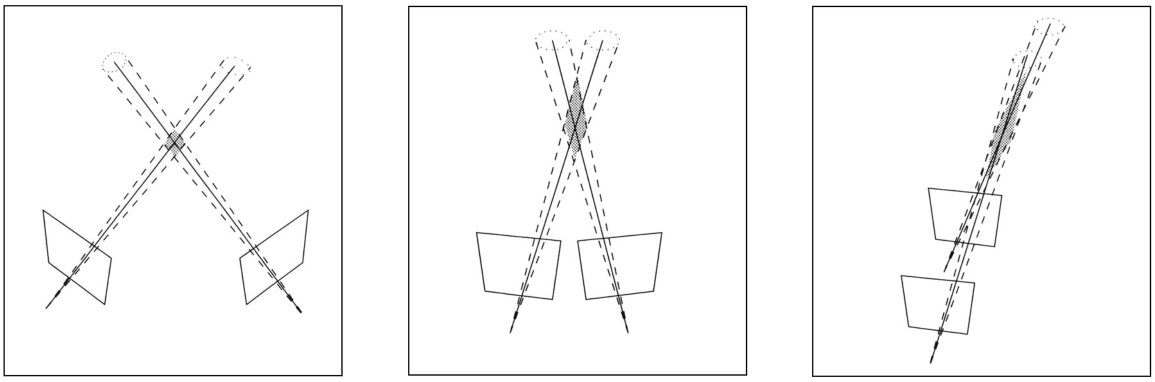


Figure 1.11: Reconstruction uncertainty, from [29, p. 321]

The points along a visual ray if projected into the image of a second camera, form a two-dimensional line called the ‘epipolar line’. This point-line relationship between cameras, called ‘epipolar geometry’, provides a valuable constraint to aid the search for point correspondences: an object feature identified in one view will (modulo measurement noise) be found on the epipolar line in another view (see Fig. 1.12).

[4]It is possible to incorporate this uncertainty, assuming Gaussian noise in 2-d feature locations, in a Bayesian framework for 3-d tracking. This has been used for individual point tracking [28], but it may be computationally intractable for nonrigid objects.

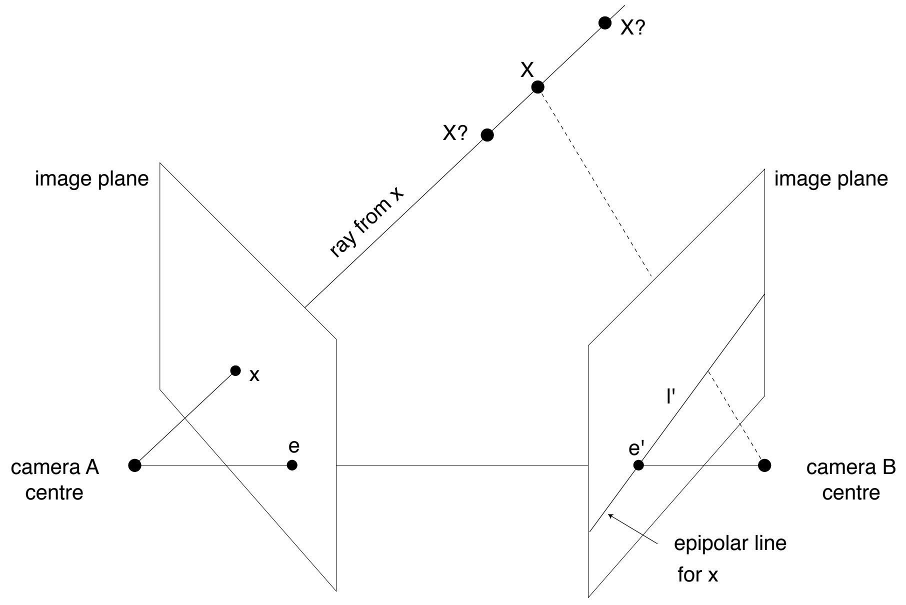


Figure 1.12: An image point x and corresponding epipolar line l' in a second view. e and e' are the projections of cameras B and A, respectively, called *epipoles*. The scene point X that projects to x must project somewhere along the epipolar line l'. Adapted from [29, p. 240]

The mapping from an image point in one view to a line in another can be encapsulated in an array called the 'fundamental matrix' F [29]. The epipolar line l' from an image point x is given by:

$$l' = \mathrm{F}x \tag{1.4}$$

and any two corresponding points x and x' will satisfy:

$$x'^{\top}\mathrm{F}x = 0. \tag{1.5}$$

The fundamental matrix is fully determined by the camera parameters described above.

Rays from image features will intersect if and only if they lie on each other's epipolar lines. These lines are coplanar, as illustrated in Figure 1.13(a), and so every point in this 'epipolar plane' will project back onto these epipolar lines. Similarly, any scene point can be identified with an epipolar plane. As shown in Figure 1.13(b), the set of all epipolar planes is seen to rotate around the baseline (the line between the camera centres), and the set of all epipolar lines pass through the epipoles (the projections of the camera centres).

The common scenario of short baseline setups (i.e. where the camera centres are much closer to each other than to the objects in the scene) with side by side left and right cameras (i.e. optical axes are coplanar) is called binocular stereo. In this scenario, the epipolar lines have another useful property: the *epipolar ordering constraint*. Consider a set of features visible to both cameras on the surface of an opaque object. If they all lie on an epipolar line in one view, then they will have the same left to right order on the corresponding

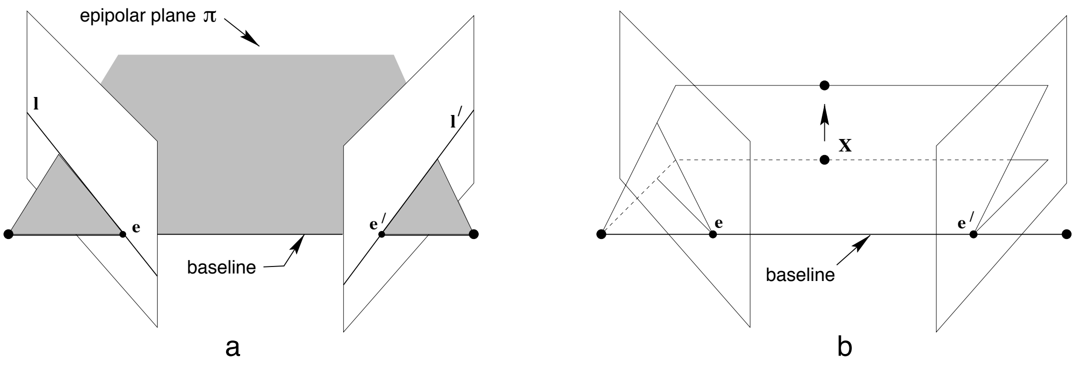


Figure 1.13: The correspondence of epipolar lines. (a) Corresponding epipolar lines are coplanar, lying in the *epipolar plane*. (b) As all epipolar lines go through the epipoles, the epipolar plane rotates around the line joining these epipoles. Adapted from [29, p. 240]

epipolar line in the other view. This can be very useful aid in constraining the search for correspondences. A pre-processing step called 'image rectification' can be used to warp the images in such a way that epipolar lines become parallel 'scanlines', as illustrated in Figure 1.14. The required 'warp' is a projective transformation, also known as a 'homography'.

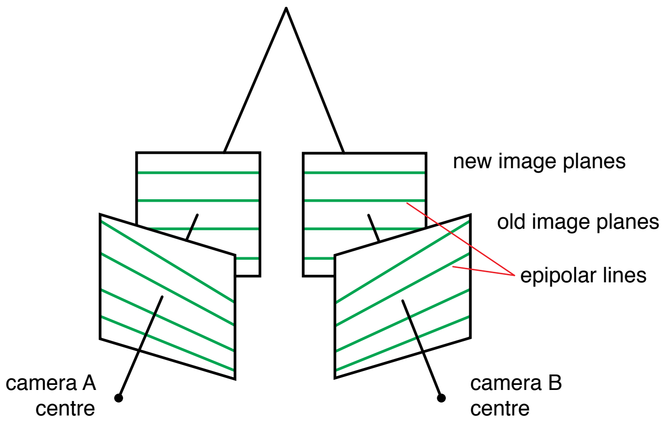


Figure 1.14: Image rectification: images are deformed onto new planes such that epipolar lines are horizontal. Adapted from [30]

The epipolar ordering constraint has been used in stereo reconstruction algorithms by Baker and Binford [31] and Ohta and Kanade [32]. Both of these algorithms use dynamic programming to match edge features found along scanlines. In the Baker and Binford algorithm, initial passes of the Viterbi dynamic programming algorithm match 'half' edges from the left view to the right and vice versa for each scanline. A final pass links pixels between the edges for a denser depth map. Ohta and Kanade perform an inter-scanline search after their intra-scanline search. These algorithms also use the assumption of 'figural continuity': the constraint that the continuity of 3-d contours from surface edges or markings should result in the continuous disparity of their projections in the stereo images [30]. (Here 'disparity' is the separation in pixels of matching image features if the stereo images were overlaid.) This assumes that the 3-d surfaces are smooth and so result in continuous or piecewise continuous projections. This smoothness also results in a more general 'disparity continuity' that should hold for the whole surface.

The reconstruction of 3-d curves can be aided by epipolar geometry in another way. Schmid and Zisserman [33] found that the osculating plane of a 3-d curve, illustrated in Figure 1.15, defines a homography between the curve projections. The curvature and tangent lines at corresponding image points on projected curves can also determine the osculating plane of the 3-d curve and the corresponding homography. Further, given a point correspondence in two views, the tangent lines and curvatures can be transferred to a third view.

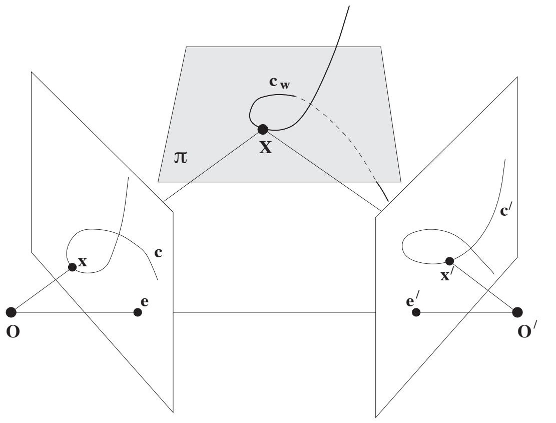


Figure 1.15: Image locations and (non-zero) curvature define osculating plane (π) of 3-d space curve. From [33]

Schmid and Zisserman outline an algorithm that can use this homography-generation for the matching of curves between two or three views. First, curves in the images are extracted. This involves pulling out the edge pixels ('edgels') using a Canny edge detector, linking them together up to a 1-pixel gap, then fitting cubic b-splines to them (minimising a cost function with separation and smoothness terms) to obtain smooth, parameterised curves. For each curve in the first image, possible matching curves in other views are identified using two curve constraints from epipolar geometry: the *epipolar beam* and *epipolar tangency*. The former constrains curve matches in a second view to intersect or lie within the 'beam' of epipolar lines corresponding to the curve points in the first view. The latter, illustrated in Figure 1.16, holds that corresponding curves must be tangent to corresponding epipolar lines.

In the primary matching stage, a putative correspondence between curve points (based on epipolar intersection) can be evaluated using a photometric similarity measure on neighbourhoods around the points. The selection of appropriate neighbourhoods uses the assumption that 3-d curves are part of a surface, and this surface can be locally approximated by using the osculating plane. A small pixel neighbourhood of a curve point in one view should therefore be mapped onto its corresponding neighbourhood in another view by the homography induced by the osculating plane. Normalised cross correlation on these neigh-

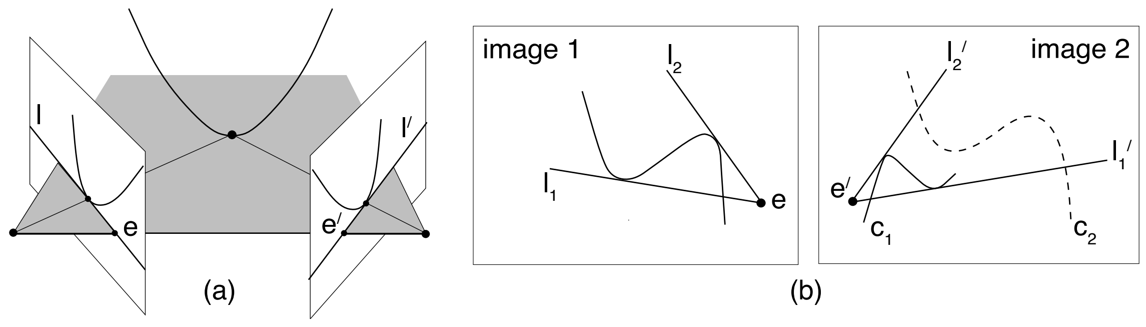


Figure 1.16: Curve tangency epipolar constraint. (a) Where a 3-d curve is tangent to an epipolar plane, the projections will be tangent to the epipolar lines (l and l'). (b) The curve in *image 1* is tangent to two epipolar lines (l_1 and l_2), so its match in *image 2* must also be tangent to the corresponding two epipolar lines ($l_1{}'$ and $l_2{}'$). Adapted from [33]

bourhoods can then be used as a score for the match. The algorithm discards matches with a score below a threshold (0.6) and points are gathered into subchains using the ordering constraint and figural continuity. The three longest subchains (over 10 pixels) are kept. A third view can be brought in for further matching discrimination. Putatively matched curve points from two views can generate points, and therefore curves, in a third view, allowing additional similarity scoring. Finally a ‘winner takes all’ scheme chooses matched curve segments with the highest similarity scores. An example of successfully matched curves is shown in Figure 1.17.

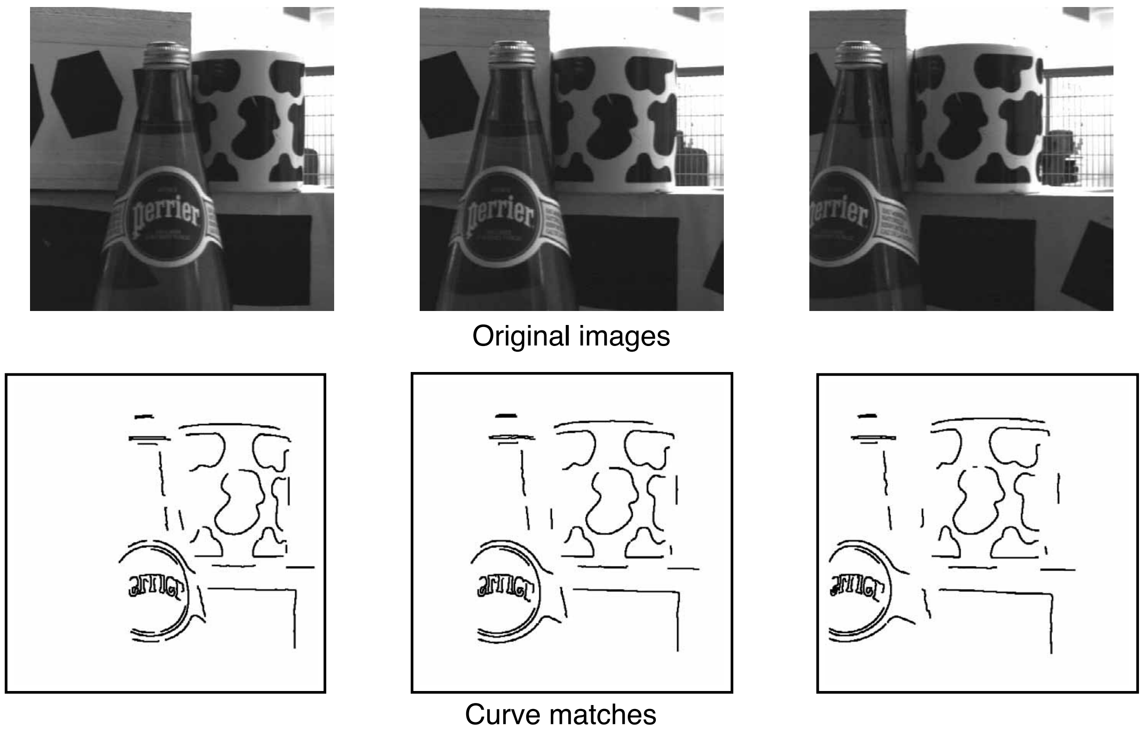


Figure 1.17: Example of successfully matched 3-d curves, projected back into original views. From [33]

Multi-view stereo

The use of multiple static calibrated cameras to build a dense three-dimensional model of a static object is known in the literature as 'multi-view stereo'. A 2006 review of multi-view stereo algorithms [34] rated 58 algorithms for accuracy and completeness in reconstructing two toy objects. The algorithms were provided with 640×480 images of the objects (which had been laser-scanned for a 'ground truth') from up to 363 viewpoints, and complete results are available online [35]. The review identified six properties that vary between the algorithms: scene representation, photoconsistency measure, visibility model, shape prior, reconstruction algorithm, and initialisation requirements.

For scene representation, some divide the scene into voxels (a 3-d grid whose elements are either occupied or empty) [36], others use level-sets (functions for the distance to the closest surface) [37, 38] or polygon meshes [39] or depth maps.

The photoconsistency measure describes how pixel values are compared to evaluate the 3-d model estimates and are typically divided between scene space measures and image space measures. The former projects elements of the 3-d estimate into each image to examine the discrepancy between small windows around the projections, and integrates these errors over the surface [37]. The latter uses the 3-d model to warp pixel windows from one view to another: error is integrated over the set of images [38].

The visibility model is the method of identifying views in which an object feature will be visible rather than occluded. Geometric methods determine visibility for every point on the surface estimate; quasi-geometric methods use geometric shortcuts such as selecting the closest camera; and outlier-based methods that assume outlier-rejection strategies will weed out bad views.

Of the shape priors that are often used to assist reconstruction, common choices are minimal surfaces (e.g. shrinking triangle meshes), maximal surfaces (finding the largest photo-consistent reconstruction), image-based local smoothness (where neighbouring pixels are assumed to have similar depth values), and surface-based priors [39].

Reconstruction algorithms can be divided into four classes:

1. Those that compute a cost function on the 3-d scene volume to extract a surface. Examples include voxel colouring, graph cuts, and max-flow algorithms.
2. Those that iteratively evolve a surface which minimises a cost function. This can be applied to voxel scenes (with 'space carving'-type algorithms that progressively remove inconsistent voxels), level sets (minimising PDEs through a volume) or mesh-based surfaces (where equations describe forces on the mesh).
3. Those that compute a set of depth maps, subject to consistency constraints, that may be merged.
4. Those that extract feature points and fit a surface to their 3-d locations.

Finally, algorithms vary in the form of initialisation that they use. Typically this is information about the extent of the object to be reconstructed. Some require a rough bounding volume, others need the images to be segmented, and some that use depth-map representations require a maximum and minimum allowed depth to be specified.

The reconstruction algorithm by Furukawa and Ponce [40] was consistently one of the top performers in terms of accuracy and completeness. It works by building up a dense set of small, oriented, textured rectangular patches that cover the object's 3-d surface, then uses them to create a 3-d triangulated mesh. The process does not require initialisation of the types mentioned above, but does assume that the surface is Lambertian. It starts with the extraction of affine-invariant local features (discussed in Section 1.2.1) - in particular using Harris corners and Difference-of-Gaussian features. Each image feature is paired with every other feature within 2 pixels of its epipolar lines. The rays of each feature pair are intersected to get a 3-d location for putative patches, which are pared down and refined through visibility and photometric constraints. The algorithm then iterates through expansion and filtering steps: an image exploration procedure (of the kind discussed in Section 1.2.1) is used to search for additional patches, then outliers are rejected. An algorithm called 'poisson surface reconstruction' then converts the resulting dense set of oriented points into a triangle mesh, upon which further photometric and regularisation constraints can be enforced. A visualisation of the steps is shown in Fig. 1.18.

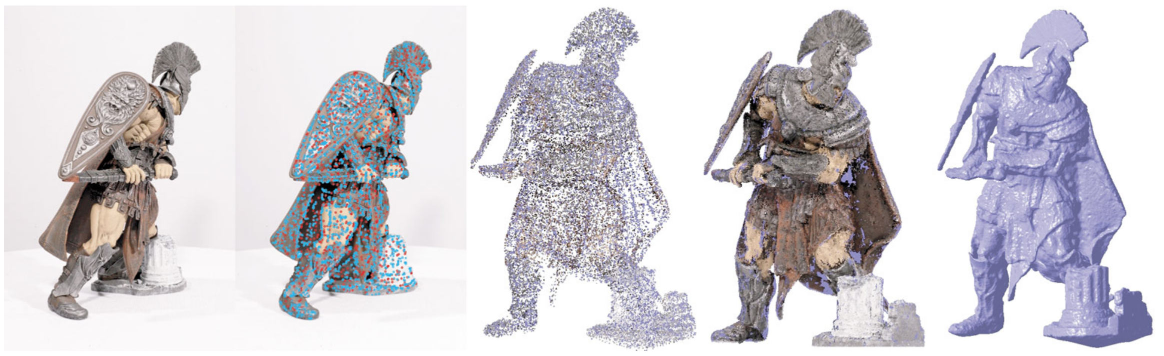


Figure 1.18: From left to right: an example image; extracted features; initial image patches; patches after expansion and filtering; 3-d mesh model. From [40].

The state of the art multi-view stereo algorithms produce 3-d models with sub-millimetre accuracy and very few outliers. The fifteen algorithms in the Seitz review that produced the most accurate model from a minimally textured dinosaur toy (7cm $\times$ 7cm $\times$ 9cm) were accurate to within 0.5mm for 90% of [35] their points. The Furukawa and Ponce algorithm achieved an accuracy of within 0.28mm for 90% of its points.

Silhouettes and visual hulls

A good 3-d reconstruction does not necessarily require texture. Silhouettes (i.e. binary images of segmented objects) from multiple calibrated views can be used to obtain an approximate 3-d shape. This shape comes from identifying the volume that is consistent with each silhouette. As concave regions of the object do not appear in any silhouette, the true 3-d volume is enclosed by the shape, and the quality of its approximation depends on the number of views, camera positions and complexity of the object [41]. Laurentini [42] coined the term ‘visual hull’ for the maximal volume consistent with the silhouettes from a given set of viewpoints. The back-projection of an individual silhouette outline is called its ‘visual cone’, and it is the intersection of these cones, considered as solids, that produces the visual hull [43], illustrated in Fig. 1.19.

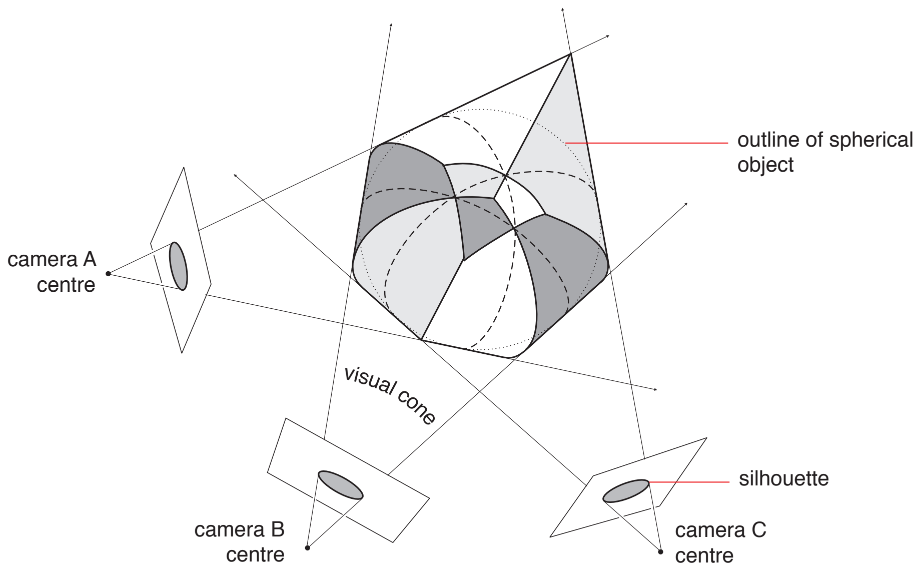


Figure 1.19: Visual cones from three images of a spherical object intersecting to produce a visual hull. Adapted from [43].

Calculation of the visual hull is an important step in a number of multiple-view stereo techniques [34]. Numerous algorithms have been developed to calculate it. These approaches typically involve attempting to directly implement cone intersections using some polygonal or volumetric approximation [43]. A polygon approximation for silhouettes was used in an early attempt [44] that created polyhedral visual hulls. Later techniques commonly split the scene volume into voxels and identify which ones are consistent with the silhouettes, performing this step in either 3-d or 2-d (projecting the voxels back into the images). Szeliski’s ‘voxel carving’ algorithm [45], for example, projects voxels into every image to test whether they are consistent with each silhouette. It does this using a coarse-to-fine ‘octree’ formulation: an initial large voxel region is tested for overlap with silhouettes, and split into 8 smaller regions if an overlap exists. A voxel representation of an object can be turned into 3-d polygonal mesh using an algorithm such as *Marching Cubes* [46]. Drawbacks of a voxel-based approach include the presence of aliasing artifacts in the resulting models from the choice of resolution, and biasing effects from the choice of grid orientation [47].

Another class of visual hull algorithm is the image-based approach. This involves the use of epipolar geometry rather than the calculation of 3-d intersections, creating high precision object models from points that lie on the surface of the hull. Some of these methods produce images of the visual hull from an arbitrary viewpoint. Matusik et al's *image-based visual hull* algorithm [48] accomplishes this by sampling along visual rays from the virtual camera image, and can render new viewpoints in real time. Later algorithms have managed to extend this approach to produce view-independent polyhedral models. An algorithm by Boyer and Franco [49] calculates points on the visual hull using epipolar line intersections. For each boundary point of a polygonal silhouette, epipolar lines are identified in other views, and their silhouette intersections are extracted. As the epipolar line is the projection of a visual ray, the intersection points demarcate depth intervals of the ray, as illustrated in Fig. 1.20. The start and end points of depth intervals consistent with all silhouettes necessarily lie on the visual hull. The algorithm then converts the resulting set of hull surface points to a Delaunay triangulated mesh.

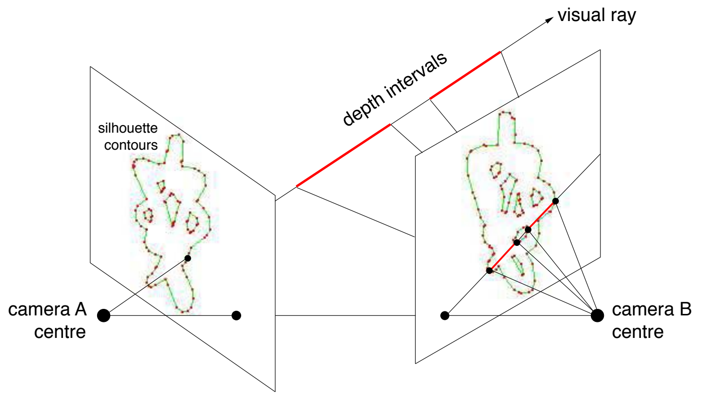


Figure 1.20: Depth intervals from silhouette contour epipolar lines. Adapted from [49].

State of the art visual hull methods [43, 47] attempt to find exact polyhedron models made of 'cone strips' (smooth sections of the visual cones, shown in white, light grey and dark grey in Fig. 1.19). The algorithm by Lazebnik et al. [43] assumes smooth silhouettes, and constructs a generalised polyhedron with smooth edges and faces corresponding to cone surfaces and their intersections. Silhouette contours are densely discretised and curvature approximated. This aids the search for corresponding contour points that are tangent to epipolar lines (so-called 'frontier points', where two strips cross each other) and 'intersection points', where rays from points on three contours meet. These points make up the complete set of visual hull vertices. From here the algorithm identifies the set of all edges (cone intersection curves), fills in the faces with a triangular mesh, and iteratively refines the model as additional views are considered.

The *EPVH* algorithm from Franco and Boyer [47] produces a virtually indistinguishable polyhedron but is more computationally efficient. They compute ray depth segments, as described for their earlier algorithm [49], to find edges that lie in the cone strip interiors, then fill in the cone intersection curves and intersection points that connect them. The final model is produced by considering all views at once, as opposed to the iterative refinement of the other algorithm. An example of the visual hull output from is shown in Figure 1.21.

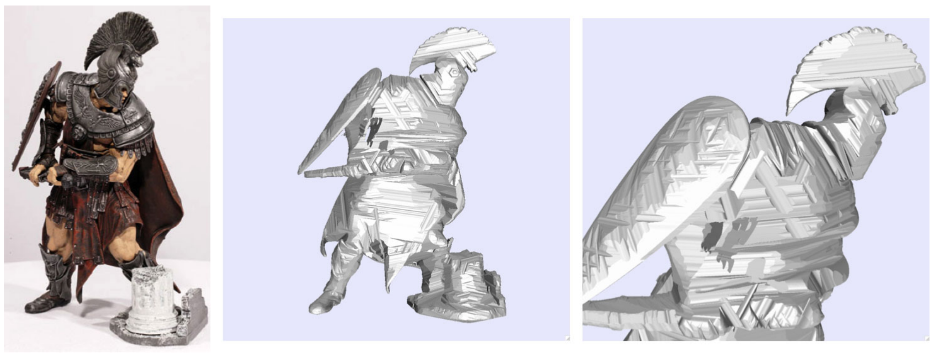


Figure 1.21: Visual hull reconstruction from [47]. Left: example input view, middle: rendered polyhedron, right: hull detail.

1.2.3 Single view, image sequence: 2-d motion

The previous two sections have examined methods of localising an object's 2-d projection and using multiple 2-d projections to build a static 3-d shape. The next step is to bring in the 'time' dimension, and so this section examines methods that allow deforming shapes to be tracked through an image sequence.

Active Contours, Deformable Templates and Spatio-temporal filtering

The field of 'visual tracking' is concerned with the repeated localisation of an object in an image sequence, and primarily focuses on following a shape and its boundary between images rather than seeking dense correspondences inside the shape. A review of visual tracking methods is provided by Blake [50], which covers the most basic forms of tracking ('simple appearance models' such as template-based cross-correlation searches over an image) and moves onto the range of techniques called 'active contours'.

Active contours provide methods of 'top-down' shape-based matching involving the specification of a parameterised curve $r(s), 0 \leq s \leq 1$ which is designed to be attracted to particular features in an image $I(r)$. This curve is often a simple shape, such as an oval outline for human head tracking, an ellipse for a mouth or a tear-drop shape for a leaf. One of the most popular varieties of active contour is the theory of 'snakes'. The theory behind snakes uses the metaphor of an image as an intensity 'landscape' with a potential energy field, $F(r)$, over which the active contour, or 'snake' can slide. The energy field might be defined by a filter on the image. If the filter gives a high output where image contrast is high, for example, the snake will be attracted to object edges. This is controlled by equilibrium equations set up for $r(s)$, so that $F(r(s))$ is maximised over $0 \leq s \leq 1$. The 'external' potential energy (which causes F to be maximised) is counterbalanced by an 'internal' energy which tends to preserve curve smoothness:

$$\underbrace{\left(\frac{\partial(\omega_1 r)}{\partial s} - \frac{\partial^2(\omega_2 r)}{\partial s^2}\right)}_{\text{internal forces}} + \underbrace{\nabla F}_{\text{external force}} = 0 \tag{1.6}$$

Here, w_1 and w_2 are positive coefficients which determine the restoring forces of the snake's elasticity and stiffness, respectively, and may vary along the snake. To allow a kink, say, at $s = s_0$, w_2 would dip to zero at this point [51].

For shape tracking, the behaviour over a sequence of images $I(r,t)$ must be defined. To deal with the changing intensity patterns, a model is required which encapsulates any prior knowledge about motion and deformations which could occur. This is used to give the snake a mass, and add 'inertia' and 'viscosity' to the equilibrium equation:

$$\underbrace{\rho r_{tt}}_{\text{inertial force}} = \underbrace{-(\gamma r_t - \frac{\partial(\omega_1 r)}{\partial s} - \frac{\partial^2(\omega_2 r)}{\partial s^2})}_{\text{internal forces}} + \underbrace{\nabla F}_{\text{external force}} = 0 \tag{1.7}$$

Careful specification of the parameters allows control over allowed motion.

The form of dynamical modelling in equation 1.7 can be combined with another powerful type of structure, a deformable template, to create a 'dynamic contour'. The soft shape constraints of the snake model are replaced with explicitly enforced constraints on a deformable template, which is a parametric shape-model $r(s;X)$ with relatively few degrees of freedom (Fig. 1.22). Matching proceeds in a similar fashion to snakes, where a search is conducted over parameter vector X to minimise an energy equation.

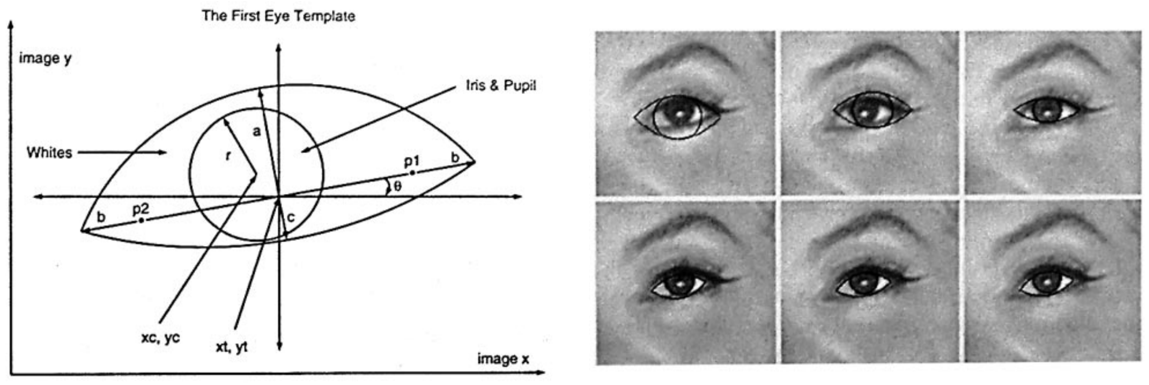


Figure 1.22: Parameters of an eye template, and fitting through 'gradient descent' [51].

The implicit constraints on shape and motion from equation 1.7 become more explicit by using image measurements to build a model for likely motions. Predictions can be made for the motion from one frame to the next, then measured image features can refine the predicted positions. The Kalman filter is a very popular tool for applying this probabilistic mechanism. It explicitly represents uncertainty in the observation of each point as noise from a Gaussian distribution [52]. An automatic balance is found between observation and prediction as measurement uncertainty is traded off against uncertainty in the predictions.

The more powerful temporal filtering method of 'particle filters' exceed the capabilities of the Kalman filter, shedding restrictions on dynamics and proving more robust to clutter. Active contour particle filter methods can perform well even in difficult tracking scenarios, successfully tracking fast-moving outlines of deforming objects. Figure 1.23 shows a leaf whose teardrop outline was successfully followed in spite of the heavy camouflage of a cluttered leaf background.

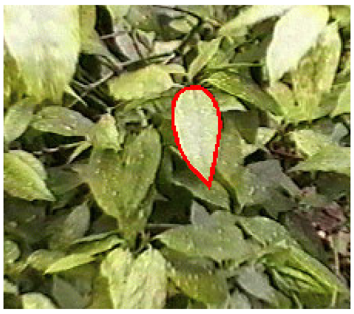


Figure 1.23: Active contour following leaf outline [51].

The active contour methods are highly effective motion trackers, but suffer from limited accuracy in deducing object boundaries. (The boundaries established by particle filter are even less precise than those obtained through Kalman filtering.) For this reason, a recent paper [53] combined active contours with the morphological 'watershed' segmentation algorithm. The watershed algorithm, which normally provides an over-segmentation of an image, can provide boundary information used to improve the accuracy of the active contour segmentation. This involves the constraining of the contours to stick at the border of homogeneous regions.

Active Shape and Texture Models

Active Shape Models are statistical models of objects which can capture non-rigid motions. A training set of labelled examples is used to build a 'Statistical Shape Model' which constrains the shape, and an iterative fitting process is used, analogous to that for active contours [20]. The training set is built from points manually marked on different objects of the same class, in order that the shape constraints are consistent with intra-class variations. Principle Components Analysis (*PCA*) is used to construct an eigen-space of shape. Fitting involves searching along profiles about the current marker point and updating the shape model parameters accordingly. Figure 1.24 shows an example of ASM convergence. This used a fast multi-resolution search strategy, allowing convergence within one second.

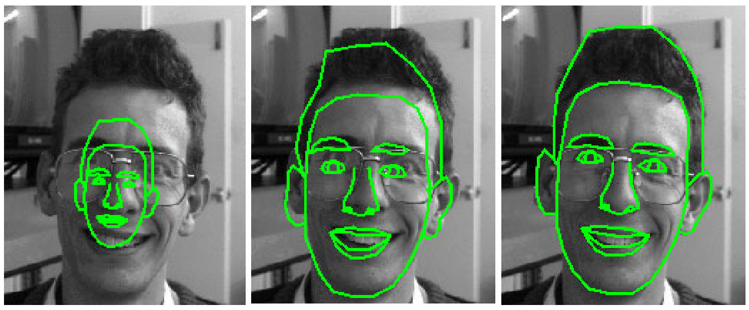


Figure 1.24: Iterative fitting of a face model to ASM pioneer, Tim Cootes. From [54]

Active appearance models are similar. They attempt to match position of marker points as well as a representation of the object's texture to an image. This has been found to be better at matching to object textures than the ASM, but the ASM is faster and achieves a higher degree of accuracy in feature point localisation [55]. Active shape models and active appearance models both depend on a good initial estimate of target location [56]. Section 1.2.1 dealt with methods which could be used to this end.

Robust point matching

Other non-rigid shape matching strategies exist, which attack the problem from a different angle. These deal with matching points between two large point sets which may contain many outliers, and formulate the matching as an optimisation problem to preserve local structures. The 'thin plate spline robust point matching' method (*TPS-RPM*) of Chui and Rangarajan [18] is one such method. Thin plate splines (TPS) provide a closed-form non-rigid spatial mapping between coordinate systems based on the correspondence of a number of known points (control points) in those systems, and can be used in the context of tracking to represent the deformation of surfaces [18]. Standard thin plate splines give an 'interpolating' warp of minimal curvature by minimising an energy function. This function can also be modified to include an additional smoothing parameter, producing a 'smoothing thin plate spline'. In the TPS-RPM algorithm, a TPS parameterisation is used as a 'soft assignment' for correspondence. Correspondences are weighted by an entropy term, then used to refine the transformation [57]. The algorithm distinguishes itself from previous work by iteratively solving for both correspondence and transformation. It can cope with large deformations and a large number of outliers, as illustrated in in Figure 1.25, and results compare favourably with the closely related 'Iterated Closest Point' (ICP) algorithm.

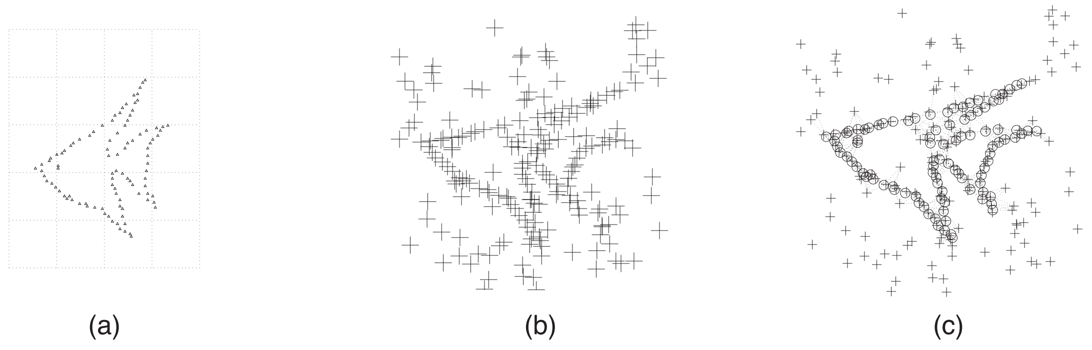


Figure 1.25: TPS-RPM demonstration: (1) 'template' reference shape points (2) noisy point set to be matched (3) matching results: template points circled. From [18].

The TPS-RPM algorithm was used in the edge-based object recognition algorithm of Ferrari et al [19], discussed in Section 1.2.1. In that algorithm, the point sets came from a sampling of the pixels on the object boundaries, and TPS-RPM was used to find transformations between objects of the same class. The algorithm also has uses in medical imaging, aiding the registration of medical images based on identified control points.

Optical Flow

Another important concept in object tracking is *optical flow*. This is the 'apparent motion' of intensity patterns from one image to another [58], providing a 'flow field' of the 2-d displacements of every pixel, and therefore a 2-d registration of the images. This can be used as an estimation of the 'motion field': the 2-d projection of the 3-d motion of objects in the scene. A review of optical flow techniques by Baker et al. [58] observed that most approaches formulate optical flow as an optimisation of a weighted-sum global energy function:

$$E_{\text{Global}} = E_{\text{Data}} + \lambda E_{\text{Prior}} \tag{1.8}$$

The data term evaluates consistency of the flow with the images, and the prior term provides a bias for certain types of movement. Most algorithms assume 'brightness constancy' (that moving pixels retain their intensity and colour), and attempt to find a flow which minimises changes in intensity. Alternatively, image features may be used, such as intensity gradients or SIFT features [59]. The prior term is typically a smoothness prior, such as a simple first-order prior, minimising the sum of the squares of the flow gradients. Alternatively an affine prior could be used, biasing towards affine transformations, or a 'rigidity' prior that biases for movements along epipolar lines.

Optical flow algorithms may be continuous, such as 'steepest descent' algorithms [60], or discontinuous, such as the discrete Markov Random fields approach (*Dynamic MRF*) of Glocker et al [61]. The Dynamic MRF algorithm is a discrete optimisation algorithm, in that it discretises the continuous space of flow solutions, allowing a more efficient search. The algorithm uses an iterative refinement technique, starting with sparse sampling of the possible motions on low resolution versions of the images, followed by refinement steps based on the uncertainty in the estimations.

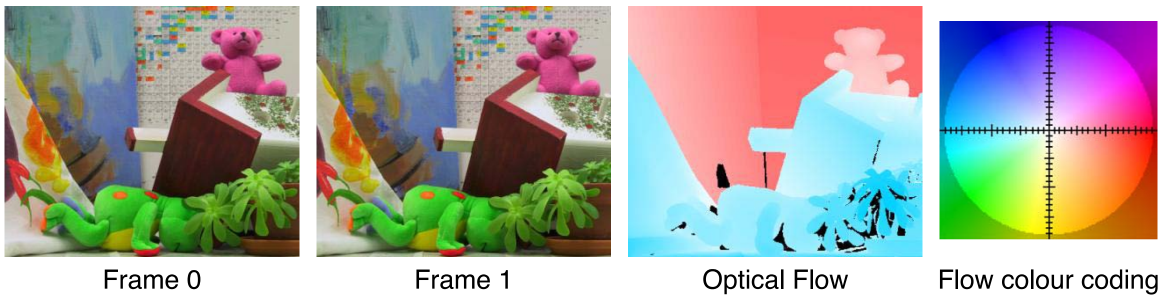


Figure 1.26: Optical flow example. Two frames from a scene, and the corresponding optical flow field (with pixel displacements indicated by a colour map). From [60].

Image-based non-rigid surface registration

Closely related to the problem of optical flow is the area which deals specifically with the 2-d flow field corresponding to a particular non-rigid surface: image-based non-rigid surface registration. Algorithms in this area define a template, composed of a reference image and a defined ‘region of interest’ containing the surface. The tracking process is the identification of a map between the 2-d template and the target image: an image warp which should overlay the template on the target image. The search for the best warp is an optimisation problem of the same form as optical flow: an energy/cost function to be minimised includes a data term (enforcing similarity of pixel neighbourhoods) and a priors term. Suitable priors typically include surface smoothness (neighbouring pixels move together), smooth motion (with suitably high frame rate, surface points should move smoothly), and may include properties such as inextensibility and zero gaussian curvature [62]. These algorithms also attempt to cope with occlusions from external objects and self-occlusions. Figure 1.27 demonstrates attempts at surface registration in the case of these occlusions, where the warp is visualised with a regular grid.

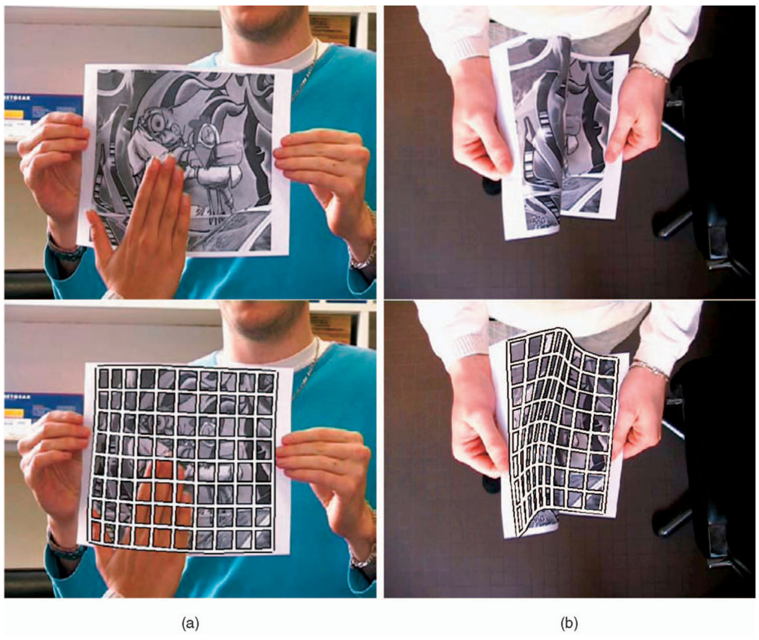


Figure 1.27: Occlusion handling in surface registration techniques without self-occlusion reasoning. (a) Successful handling of external occlusion. (b) Unsuccessful handling of self-occlusion. From [63].

These registration algorithms can be divided into two categories: direct, pixel-based approaches (e.g. [63–66]) that minimise the intensity discrepancy between template and target, and feature-based approaches (e.g. [67–69]) that use feature identification and matching, minimising the distance between matched features [63]. (There are also some algorithms which attempt a ‘fusion’ of both of these approaches, e.g. [70]). Direct, pixel-based approaches use all of the image data so can be more accurate and don’t require a matching step, but are limited to short baseline scenarios. Feature-based approaches permit wide-baseline scenarios at the cost of discarding image data. Both have sensitivity to changes in surface appearance: during warp estimation for pixel-based approaches, and during feature matching for feature-based approaches [62].

Self-occlusions are best handled by direct approaches, as these implicitly use dense correspondences around self-occluded regions. In the direct approach of Gay-Bellile et al [63], self-occlusion reasoning is incorporated using the concept of a ‘shrinker’. The shrinker is a term in the cost equation (in addition to the data term and smoothing term) which penalises variations in a spatial warp derivative [66]. It has the effect of forcing the warp not to fold, but instead to shrink on the occlusion boundary, so that a neighbourhood of occluded template pixels can map to the same location. The successful application of this algorithm is demonstrated on frames of a video sequence of folding paper in Figure 1.28. A similar algorithm which performs mesh shrinking was developed by Hilsmann and Eisert [66].

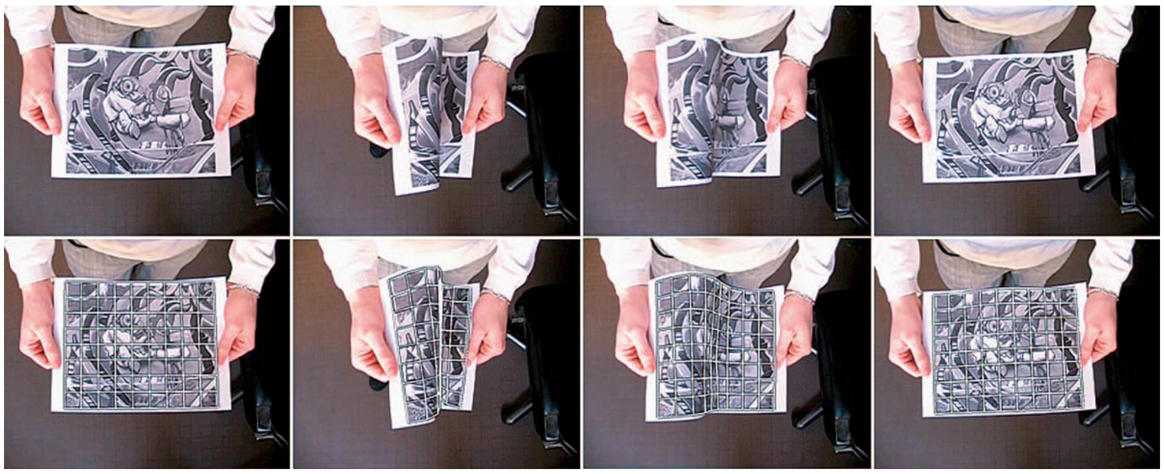


Figure 1.28: Demonstration of successful self-occlusion handling with ‘shrinker’ method. From [63].

3-d reconstruction from an image sequence

An alternative to seeking purely image-based registrations is to associate the template image with a parameterised 3-d mesh and to incorporate assumptions about allowed mesh deformations. This allows for simultaneous image registration and 3-d surface reconstruction. One such algorithm, by Salzmann et al [71], disallows large changes in angles between the triangular facets of the mesh. The algorithm assumes that 3-d shape is known in the first frame (effectively the reference template), as well as 3-d to 2-d template correspondences. Random point samples are selected from each facet, then tracked through the image sequence in 2-d using normalised cross correlation. Another cross correlation between reference and target images refines the 2-d locations, resulting in a noisy 3-d to 2-d correspondence. The reconstruction of the 3-d mesh is solved as a second order cone programming problem. Figure 1.29 shows reconstruction results from a video of folding paper.

Some algorithms attempt to learn possible mesh shapes (‘modes of deformation’) from an image sequence by extending the Active Appearance Models approach. Salzmann et al [72] create a deformation model for an inextensible mesh by assuming that a mesh can be parameterised using a small subset of the angles of its facets. Example reference mesh

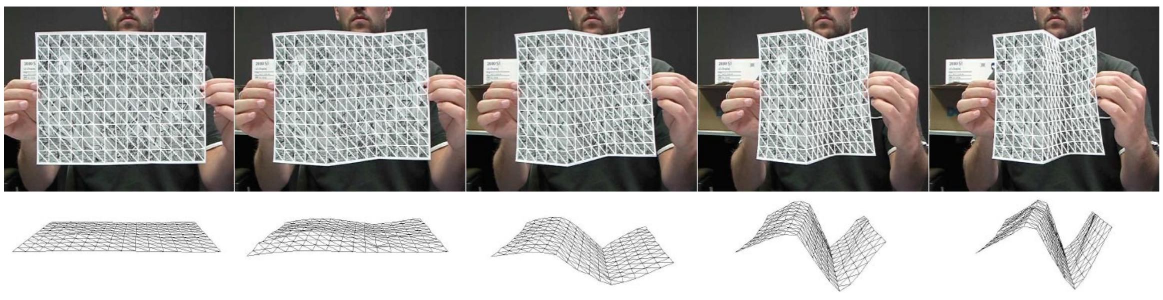


Figure 1.29: Mesh reconstruction results for frames from folding paper sequence. From [71].

shapes are shown in Figure 1.30. They create a representative set of possible mesh shapes by randomly sampling angles from uniform distribution, then use PCA to reduce dimensionality. An algorithm by Gay-Bellile et al [73] uses a very similar approach, learning possible mesh shapes with PCA and finding a piecewise-smooth image deformation field which minimise intensity differences over canny edge pixels.

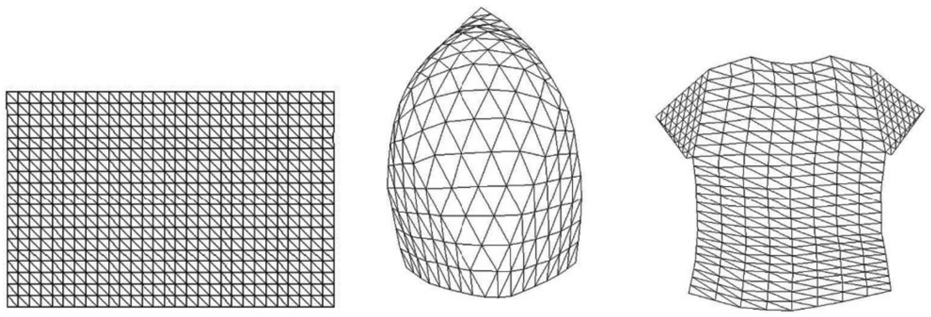


Figure 1.30: 3-d mesh templates used for a sheet of paper, a spinnaker, and a t-shirt. From [72].

Varol et al [74] implemented an local feature-based algorithm which does not require a reference view or tracking points over many images (as in [72]). It finds correspondences between pairs of images by matching SIFT features, and uses these to estimate homographies between local planar patches. Assuming a calibrated camera, these patches can then be reconstructed in 3-d up to a scale ambiguity. Identifying the overlaps between the patches allows them to be connected in 3-d space. This gives a point cloud to which a triangular mesh is added. Patch centres are converted to a point cloud, and then a triangular mesh is fitted using an inextensibility constraint. This stage can be performed for multiple frames simultaneously, enforcing consistency across frames.

The local feature-based algorithm by Perriollat et al [75] also relies on a calibrated camera and 3-d point clouds generated prior to a triangulated mesh. A set of local feature points are first matched between an input image and a template image (of known 3-d shape). Visual rays are projected out from these points, and then pairwise reference separations can be used to give an upper bound on the depth of each of these points. The upper bounds are refined iteratively by considering each point's influence on the bounds of all other points. The resulting point cloud can be used as control points for the fitting of a mesh, and this fitting can include geometric and temporal smoothness priors.

1.2.4 Multiple views, image sequence: 3-d shape and motion

The number of algorithms that successfully take on the challenge of measuring 3-d shape and motion of unmarked generic deforming objects (i.e. ‘markerless motion capture’) from multiple views is currently very small [76–78]. Such algorithms integrate solutions to each of the subproblems of the preceding sections, obtaining estimates of both shape and motion at each frame of an image sequence. Courchay et al. [77] observed that the majority of motion capture algorithms possess significant shortcomings in at least one of three areas: (a) density and accuracy of shape measurements, (b) density and accuracy of motion measurements, and (c) coupling of shape and motion estimation.

The algorithm from Aganj et al. [79], for example, is faulted for the limited shape accuracy that comes from its shape-from-silhouttes approach. The approach by Varanasi et al. [80], on the other hand, can produce accurate 3-d shapes, but obtains only a sparse set of temporal correspondences. Neither approach couples the estimation of shape and motion, and so fail to exploit their redundancy, which could otherwise provide spatio-temporal consistency constraints. Three algorithms distinguish themselves with their dense, accurate and coupled measurements of shape and motion: Furukawa and Ponce [78], Courchay et al. [77], and Neumann and Aloimonos [81].

Furukawa and Ponce [78] represents shape using a triangle mesh with fixed connectivity, and frames motion capture as the tracking of the mesh vertices. The algorithm requires an accurate 3-d mesh in the first frame to proceed - this comes from the multi-view stereo algorithm of [40], described in Section 1.2.2. Each consecutive frame is examined in turn, and for each frame the vertex positions are updated using an iterative framework containing three main procedures: local rigid motion estimation, global nonrigid motion estimation, and outlier filtering. The local motion estimation is performed for each vertex. The procedure first optimises motion parameters (translation and rotation) normal to the surface, then the full set of motion parameters, using a conjugate gradient optimisation. This optimisation maximises a photoconsistency measure (normalised cross-correlation between texture samples around vertices in views that pass a ‘visibility’ test, and samples from a ‘reference’ frame) and a smoothness measure (the difference between initial and final displacement estimates). After each vertex has been adjusted, the global estimation step performs an additional optimisation over the whole mesh. It minimises the sum of a ‘data attachment’ term (the displacement of each vertex from its predicted position), a surface smoothness term (using a discrete laplacian operator) and a ‘local rigidity’ term (the difference between mean edge lengths in the current frame and a reference frame). Thresholds on the residuals of these terms are used to identify erroneously tracked vertices. The filtering step removes these, and the global optimisation is performed again. These steps are repeated three times to give the final mesh for each frame. While many other motion capture algorithms are limited to simple motions with only small displacements between each frame, this algorithm can handle large complex motions. It also avoids error accumulation drifts that plague other algorithms. Example results are shown in Fig. 1.31.

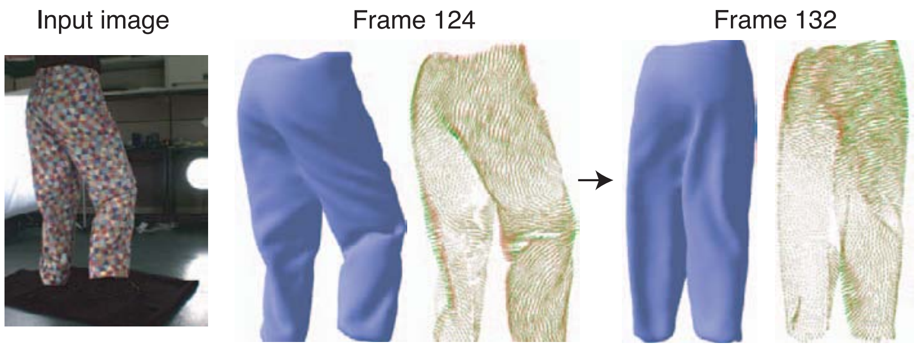


Figure 1.31: Two tracked frames for a dancer wearing textured clothing, showing the resulting 3-d mesh and motion fields. Adapted from [78].

The algorithm by Courchay et al. [77] also uses a fixed-connectivity triangle mesh to represent shape. They use an optimisation procedure with parallels to the method above, minimising the weighted sum of a data attachment term, a surface smoothness term and a velocity smoothness term. The data attachment term compares mesh projections with the original images between pairs of cameras in a single frame and between pairs of frames. The surface smoothness term calculates total mesh area, and penalises stretching or shrinking of the mesh, while the velocity smoothness term penalises changes in velocity, minimising a metric of the gradient of the velocity field. As with the previous algorithm, an initial mesh is required to get started. The algorithm first uses low resolution versions of the images to track a coarse mesh over the sequence. The coarse mesh is optimised for the first two frames, then the velocity is used to initialise the next frame. (Both velocity and acceleration are used to initialise later frames.) The optimisation is performed for a sliding time window of a few frames over the entire sequence, then the process is repeated at increasing image and mesh resolutions.

The earlier algorithm by Neumann and Aloimonos [81] represents shape using a time-varying 'subdivision surface'. This is a polygon mesh with associated rules for recursive refinement: vertices and faces are updated using these rules to approximate a smooth surface. This hierarchy of meshes, and motion vectors between vertices, is obtained in a coarse-to-fine optimisation framework, like the previous algorithm, using normalised cross-correlation as a similarity measure between the projections of surface patches. The shape is initialised in each frame by visual hulls produced by object silhouettes. This algorithm provides a high degree of automation, but has the drawback that it can only be applied to slowly moving surfaces under constant illumination.

These algorithms are very effective for tracking a wide range of deforming objects, but are ill-suited for the hoverfly images in this project. The wing texture is very limited and the nature of the backlighting results in images that are very different from those that the algorithms are designed for. Each algorithm requires that the object is Lambertian, with reflected light producing predictable textures based on surface normals. In the hoverfly images the light that reaches the camera passes through the membranes, and minimal reflection is observed. The algorithms would most likely be unable to cope with the high degree of occlusion in the hoverfly images combined with the small number of views. The Furukawa and Ponce algorithm, for example, discards views for which texture samples around a projected vertex have low cross-correlation coefficients with other views. There must therefore be more ‘good’ views than ‘bad’ to prevent all views from being discarded. The hoverfly sets, however, typically only have two views (out of four) with good wing visibility in each frame - and even then there are many frames in which a wing region is only visible in one view or not visible at all. Meeting the challenges posed by the hoverfly images therefore requires an alternative approach.

1.2.5 Summary

None of the techniques covered here can cope with the full range of challenges posed by the hoverfly data. Those techniques that offer solutions for high self-occlusion, for example, are either limited to 2-d tracking (using ‘mesh shrinking’ [63]) or they assume a highly-textured Lambertian surface with enough viewpoints to ensure complete surface visibility. Table 1.1 lists some key tracking techniques covered in this review, with a breakdown of whether they satisfy the various requirements of the hoverfly data. While no technique offers a complete tracking solution, the strategies they adopt in terms of shape representation, segmentation, feature and region matching, occlusion handling, and combining shape and motion estimation can inform a new approach tailored to this problem.

Matching points between wing images is particularly challenging due to the low texture and occlusion, but the literature shows that well-chosen shape and motion models can provide valuable assistance. Algorithms designed for tracking paper-like surfaces (e.g. [63, 67]) show the advantages of using a flat and inextensible template and assuming local smoothness, while motion models can be as simple as extrapolating velocity and acceleration measurements from previous frames (as in [77]). Direct, pixel-based registration techniques, as explored in Section 1.2.3, show the potential to cope with the poorly textured wings, as opposed to feature-based methods which don’t use every pixel.

Further support for matching between wings can come from edges and wing boundaries. Edge detection not only helps segmentation, as discussed in 1.2.1, but can also provide features for the reconstruction of 3-d curves, as demonstrated by Schmid and Zisserman [33]. As the hoverfly data comes from calibrated cameras, the use of edges to extract silhouettes provides a 3-d bounding volume (the ‘visual hull’), as well as 2-d and 3-d constraints for matching points between images.

	SHAPE	TEXTURE	OCCLUSION	INITIALISED	CAMERAS	BASELINE	CALIBRATED	
Tracking Techniques	gives 3d reconstruction	can cope with limited texture	can handle self-occlusion	doesn't require training set	uses multiple viewpoints	suitable for wide-baselines	uses camera calibration	REFERENCE
TPS robust point-matching	✗	✓	✗	✓	✗	✓	✗	[18]
Active shape/texture models	✗	✓	✗	✗	✗	✗	✗	[55]
Optical Flow	✗	✓	✗	✓	✗	✗	✗	[61]
Monocular 2d mesh tracking								
pixel-based	✗	✓	✓	✓	✗	✗	✗	[63]
feature-based	✗	✗	✗	✓	✗	✗	✗	[67]
Monocular 3d mesh tracking								
learned deformations	✓	✗	✓	✗	✗	✗	✓	[72]
feature-based	✓	✗	✗	✓	✗	✗	✓	[74]
depth from separations	✓	✗	✗	✓	✗	✗	✓	[75]
Multi-view tracking								
coarse-to-fine mesh	✓	✗	✓	✓	✓	✓	✓	[77]
using PMVS	✓	✗	✓	✓	✓	✓	✓	[78]
'subdivision surface'	✓	✗	✓	✓	✓	✓	✓	[81]

Table 1.1: Tracking techniques, indicating strengths and weaknesses for hoverfly data.

1.3 Objectives

The task of creating an algorithm capable of tracking hoverfly wings can be broken down into a number of objectives, illustrated in Figure 1.32.

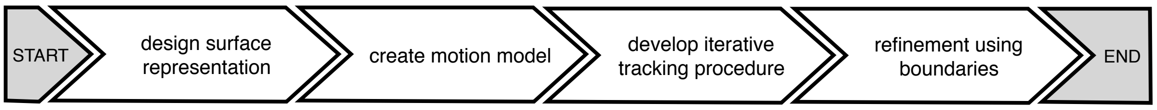

START
design surface representation
create motion model
develop iterative tracking procedure
refinement using boundaries
END

Figure 1.32: Tracking algorithm objectives.

The first objective is the selection of a suitable surface representation. The goal of the thesis is to capture the shape and deformation of the entire wing, so the surface representation must be able to give a dense set of measurements; allow large, unpredictable deformations; and be suitable for imposing constraints such as inextensibility. The wing surface could be described by the positions of a collection of points or in a continuous form as a 2-d manifold. Chapter 2, which deals with this objective, introduces the latter form: a continuous parameterised wing surface.

Chapter 2 also handles the second objective: the creation of a predictive motion model. It should be simple enough to be able to adapt to unpredictable motions, yet strong enough to give useful predictions to aid shape reconstruction. The chapter presents a novel prediction algorithm tailored to insect flight, called the 'normalised stroke model'.

Chapter 3 covers the third objective, developing the core tracking procedure. A robust tracking procedure must take full advantage of available constraints such as surface and motion smoothness. Measurements must be accurate enough to enable good shape predictions, and outliers must be removed when self-occlusion prevents successful matching.

The fourth and final objective is the topic of Chapter 4. Additional image features, wing boundary edges, are brought in to allow further refinement of the shape measurement in each frame. This involves the reconstruction of boundary curves in three dimensions, followed by a novel strategy for incorporating them into the tracking results.

With a tracking framework in place, Chapter 5 applies it to a challenging flight manoeuvre dataset, to test the versatility of the algorithm. The accuracy of the algorithm is quantified in Chapter 6, which examines the errors introduced at each stage of processing. Finally, Chapter 7 revisits the objectives, presents conclusions, and discusses the scope for further work.

Chapter 2

Shape and Motion Models

2.1 Overview

This chapter lays the groundwork for wing tracking in three stages: the segmentation of fly silhouettes, the extraction of reference wing shape and the development of a simple model for motion estimation.

The segmentation of silhouettes provides shape information and epipolar constraints, and allows background features to be ignored. They will be a key component of later sections on outlier detection and the identification of boundary points.

The reference wing shape consists of reference 3-d surface and a corresponding template image, which are related by a map between identifiable keypoints. Observation of the movement patterns of keypoints (within a suitable coordinate frame) enables the final step of this section: the creation of a motion model which can predict upcoming keypoint locations.

The two hovering flight sequences are referred to as ‘Set 15’ and ‘Set 23’, retaining their names from the data collection phase. Set 15 is the set from which most of the examples are drawn in this chapter, so all figures refer to this set unless specified otherwise. Exactly the same processing methods were used for Set 23. For brevity, the left and right wings of these sets will be referred to as 15L, 15R, 23L and 23R.

2.2 Silhouette extraction

Silhouette extraction was accomplished through a combination of edge extraction and morphological processing, with steps illustrated in Figure 2.1. Edges were extracted using Matlab’s implementation of the Canny filter (with lower and upper thresholds 0.001 and 0.2, respectively). Broken edges in the binary edge image were closed using a morphological closing operation (dilation then erosion with a ‘disk’ structuring element with a 5 pixel radius). The areas bounded by the edges were then filled (using Matlab’s *imfill* function) and the largest connected area (identified by Matlab’s *bwconncomp* function) was taken as the fly silhouette.

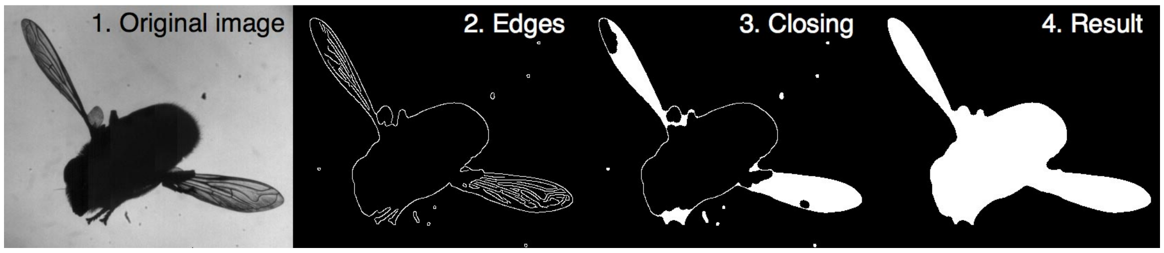


Figure 2.1: Silhouette extraction steps. (Note: in 2 and 3, images are inverted.)

The steps and parameters of this algorithm were built up using trial and error for set 15, but proved to work well for later sets. In choosing the Canny thresholds, there was a trade-off between being low enough to be sensitive to soft edges of the fly boundary (in regions with a relatively dark background or regions around abdomen hairs) and being high enough to discriminate against subtle background features. The 'closing' step adds another dimension to the trade-off as it can close gaps in edges but has the drawback of smoothing the boundary shape. A completed boundary was prioritised at the expense of pulling in more background features. This results in some background features occasionally becoming attached to the fly silhouettes, but as will be seen in later chapters, it is better to have pixels incorrectly labelled as foreground (i.e. the fly) than background. Example silhouettes are shown in Figure 2.2.

2.3 Reference image, keypoints and 3-d wing shape

In order to build up a reference wing shape to assist tracking, the first step was to identify a set of recognisable wing landmarks. Twelve points, shown in Figure 2.3, were identified at vein intersections. These were chosen as they are well-spread, easily recognised, are present in the wings of different hoverflies, and cover the entire wing (with the exception of the alula).

Images were then identified that could represent the texture of each wing. (These template images are used later for direct image registration.) Separate images were chosen for each wing. The selection criteria were that the wing had to be fully and clearly visible, appearing to be flat and facing the camera, and, if possible, with no dark background features overlapping with the wing membranes which might throw off registration. In each case, images from the downstroke were deemed most suitable: the wing appeared most planar midway through the downstroke for each fly. The selected template images are shown in Figure 2.4.

The image coordinates of the 12 keypoints were manually identified for each template. Identification was performed using a Matlab script for image magnification and the conversion of the location of mouse clicks to subpixel image coordinates. Keypoints identified in this way will be referred to as 'clicked' points. As each keypoint is located at the centre of

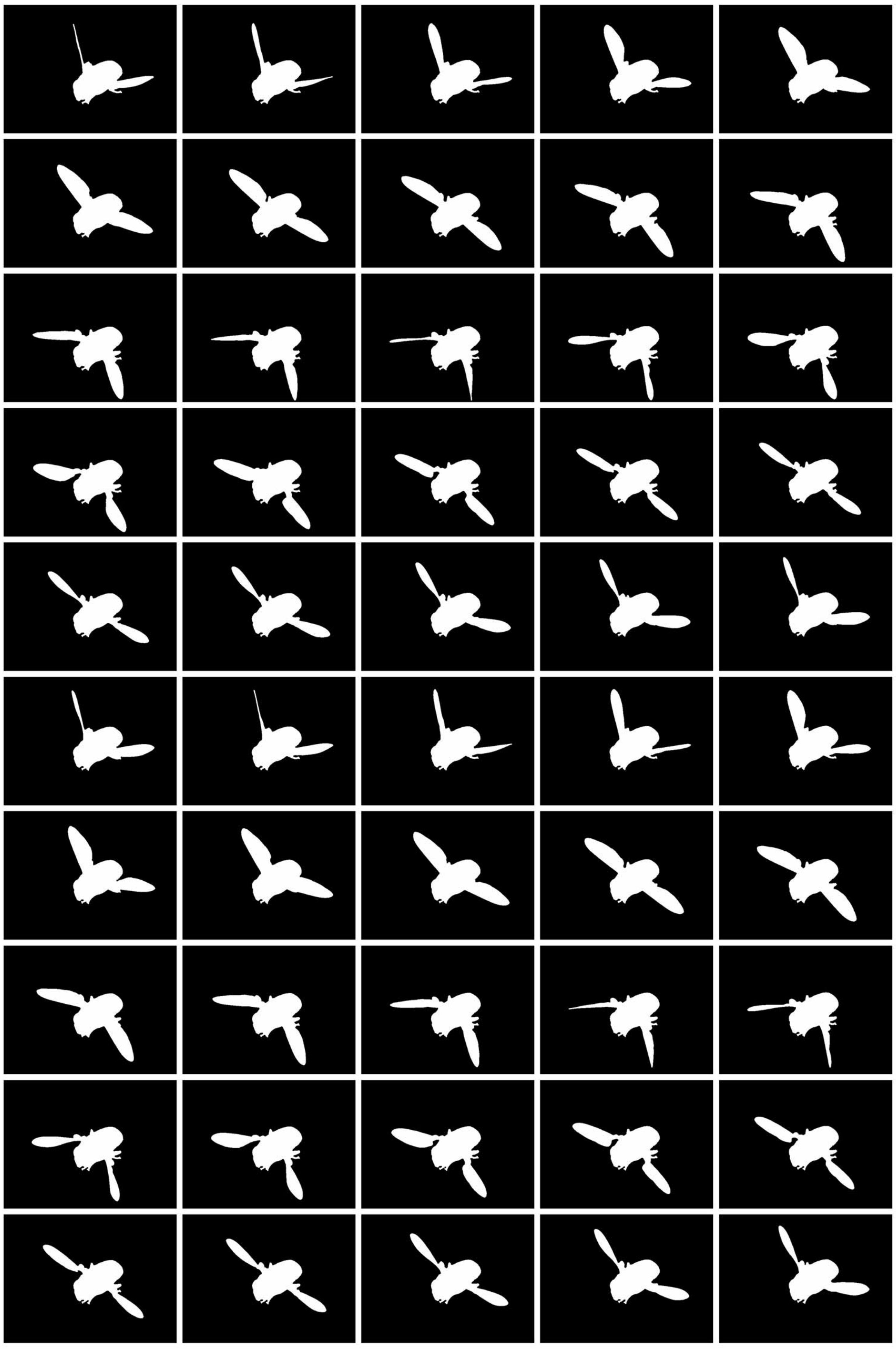


Figure 2.2: Fifty consecutive silhouettes, ordered in rows from left to right. (Frame 1-50, camera 1)

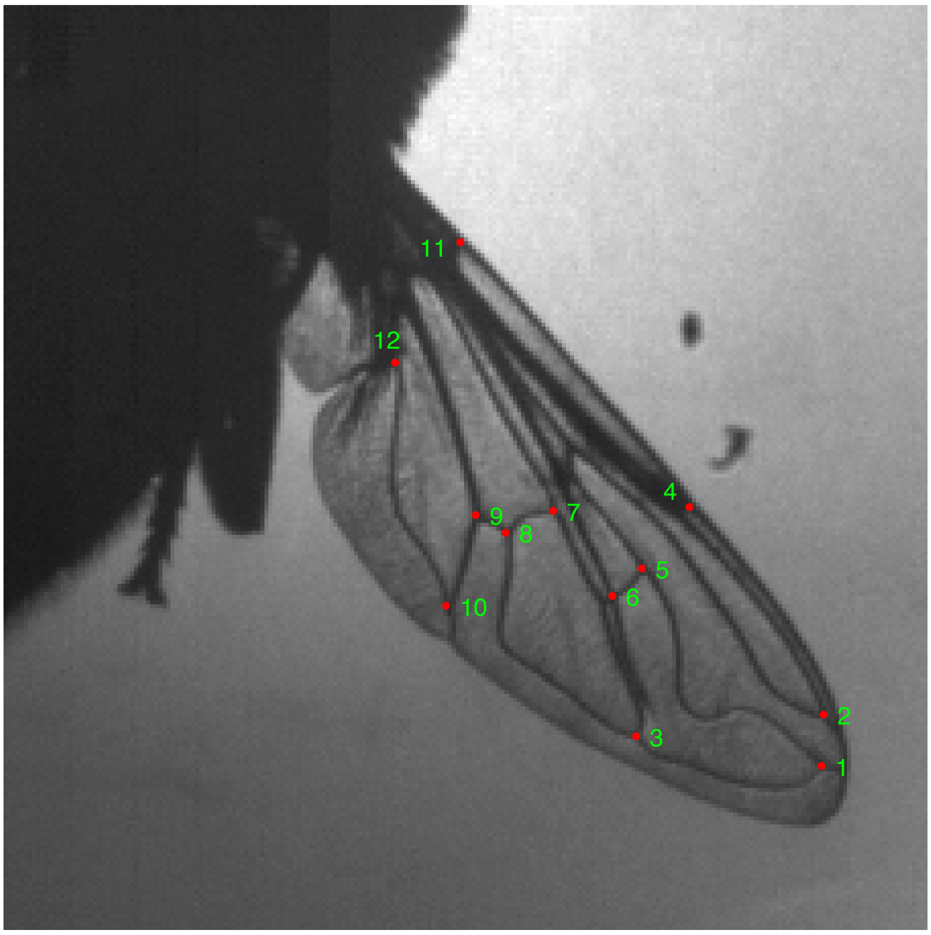


Figure 2.3: 12 wing keypoints.

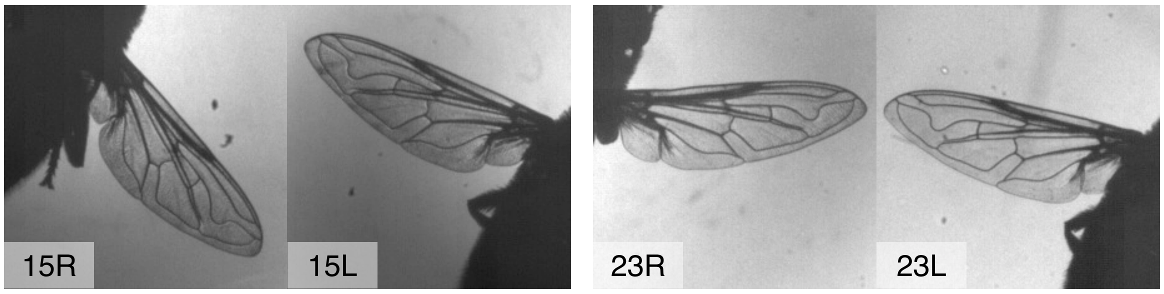


Figure 2.4: Wing templates for left and right wings of sets 15 and 23.

a vein intersection, they could be localised by eye to within an accuracy of approximately half a pixel (varying with the pixel area of the feature in each image, perspective distortion and local contrast).

Three-dimensional information was acquired for the wings in the form of millimetre separations between keypoints. To find these separations frames were first selected in which the wing appeared to be very flat. For three of the four wings the template frames were chosen, but for one wing (15R) there was an even more unambiguously flat frame where the wing appeared perfectly edge-on in one view, projecting into the image as a very thin strip only a few pixels wide at its thickest point. By clicking on each keypoint (where visible) in each camera, and performing a 3-d intersection for each of these points[1], 3-d location estimates were found. The reprojections of these points in each image were examined and any points whose reprojections seemed poor or (in the case of the thin strip view) lay outside of the wing boundary were identified. Such errors would most likely result from imperfect keypoint identification. A script was written to nudge these points in 3-d by 0.02mm at a time along each of the three axes, in order to make minor adjustments to the 3-d locations.

Rather than assuming the points all lay in a perfect plane, the use of thin-plate splines as a surface representation allowed for some wing curvature. These thin-plate splines (calculated using Matlab's *tpaps* function, with a smoothing parameter value of 1 so the data is interpolated) mapped between 2-d points on a 'general template' (from a view of the perfectly flat wing in set 15R mentioned above) and the improved 3-d points. To visualise the TPS wing surface, wing spar (i.e. vein) pixels were manually traced in the general template, as shown in Figure 2.5. By mapping these into the 3-d scene using the TPS surface and then projecting them back into the camera views, they provide a qualitative way of evaluating the surface fit: a good fit will result in the traced spars projecting back onto the correct locations in each view. Figure 2.6 demonstrates this, showing the spars resulting from the reference surface fit for the set 15R.

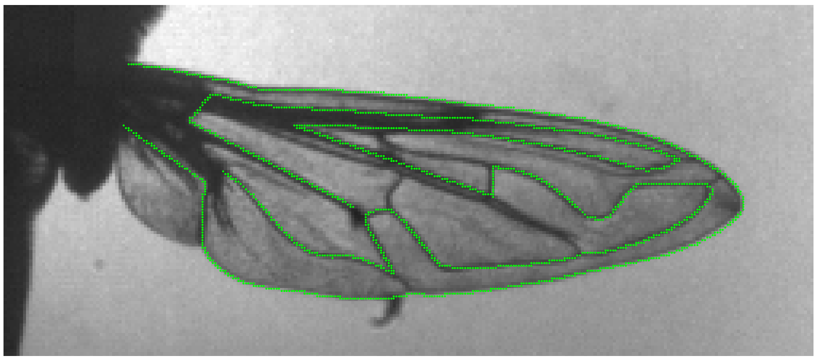


Figure 2.5: Traced edges on general template (Set 15R, Frame 12, Camera 3), used in 3-d wing visualisation.

[1]Intersection is performed using the linear triangulation method described in [29, pp. 312-313], and implemented with accompanying code at http://www.robots.ox.ac.uk/~vgg/hzbook/code/

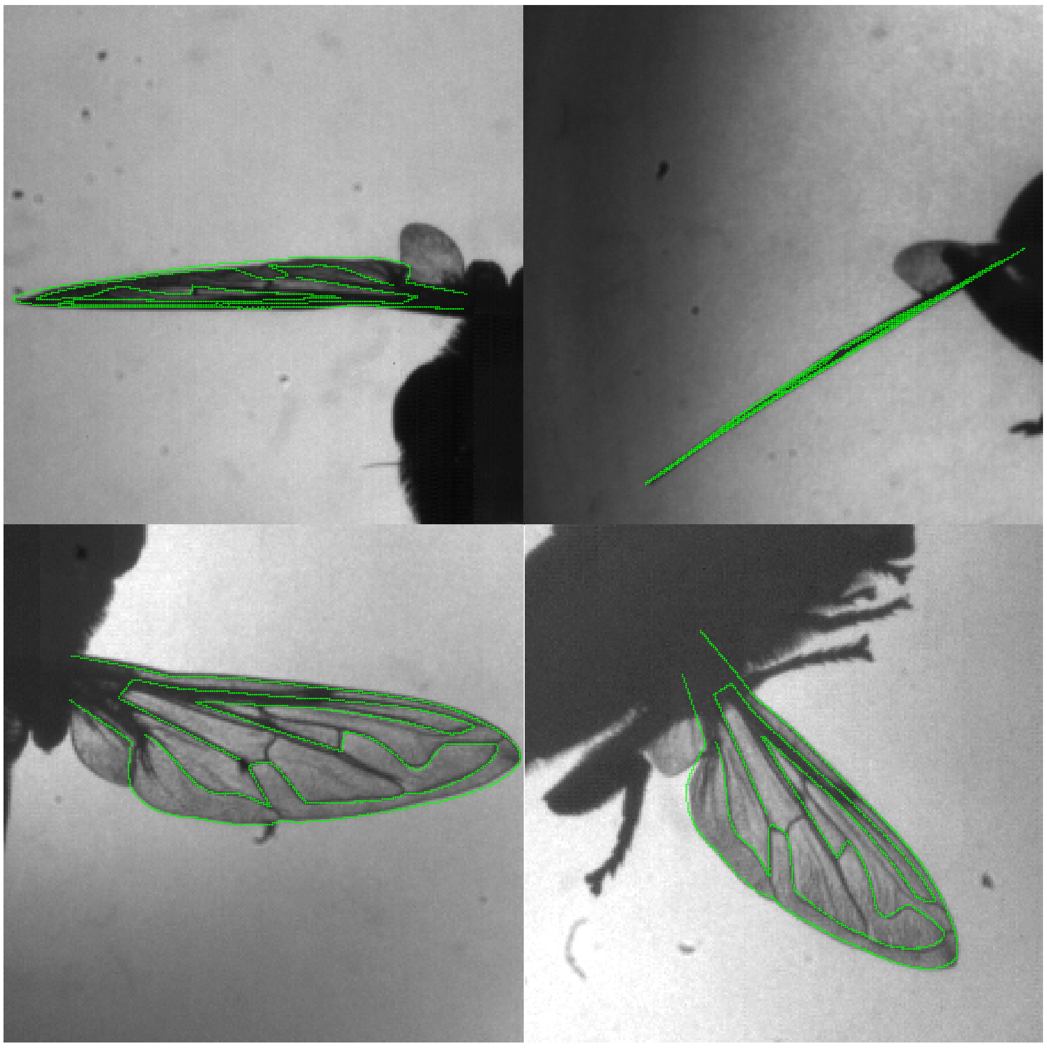


Figure 2.6: Spar projections for the 3-d reference shape for the set 15R.

The quality of these projections shows the power of a small subset of keypoints to encode a good approximation of wing shape. A larger number of template points could have been used to attempt a closer fit, but for the purposes of estimating separations between the 12 chosen keypoints the fit appears adequate. Later enforcement of these separations will allow for some imperfection in their estimation.

In order to estimate separations, paths were constructed along the wing surface. For each pair of points this involved finding the line segment between them on the general template, pulling out 100 equally separated points along this segment, then mapping them onto the 3-d wing surface using the previously constructed TPS map. Adding up the straight-line separations between these points gave an estimate of the separation between keypoints in millimetres. For the flat template of set 15R an alternative estimation method would be to assume the keypoints are mapped onto a common plane which is found by estimating a homography from their 2-d locations. For generality, however, this TPS mapping method is appropriate as the presence of a flat wing frame cannot be guaranteed. Table 2.1 shows estimated separations for set 15R.

pt 1	pt 2	pt 3	pt 4	pt 5	pt 6	pt 7	pt 8	pt 9	pt 10	pt 11	pt 12	
0	0.78	2.79	4.39	3.90	3.93	5.55	5.80	6.28	6.05	9.48	8.85	pt 1
	0	2.70	3.69	3.33	3.44	4.99	5.31	5.80	5.71	8.82	8.29	pt 2
		0	3.43	2.38	2.02	3.58	3.54	3.96	3.43	7.75	6.70	pt 3
			0	1.14	1.69	1.93	2.58	2.99	3.65	5.15	4.88	pt 4
				0	0.56	1.67	2.08	2.56	2.86	5.63	4.97	pt 5
					0	1.66	1.88	2.36	2.44	5.79	4.94	pt 6
						0	0.70	1.06	1.95	4.17	3.31	pt 7
							0	0.48	1.27	4.33	3.16	pt 8
								0	1.22	4.05	2.75	pt 9
									0	5.18	3.66	pt 10
										0	1.90	pt 11
											0	pt 12

Table 2.1: Millimetre separations between template points (rounded to 2 d.p.) for set 15R.

2.4 Constructing a wing coordinate frame

In order to describe the motion of points on the wing it is necessary to construct a suitable frame of reference in which to examine them. The wing shoulder was taken as origin, and basis vectors as indicated in Figure 2.7, following the choice of Zbikowski et al [82]. The principal axis of the fly's body forms the x-axis (with head on the positive side of the axis); the 'downwards' direction perpendicular to this forms the z-axis; and the y-axis is given by the 'outwards' direction, perpendicular to both.

To estimate these basis vectors, points on the 'head' and 'tail' of the fly were clicked in each view in an early frame, using epipolar lines to assist in finding suitably corresponding

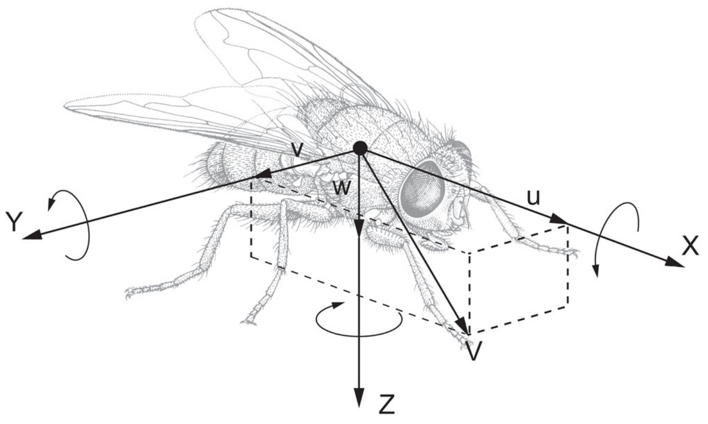


Figure 2.7: Illustration of X,Y,Z axes for wing coordinate frame, from [82]

points (as there was limited texture to provide guidance). Intersection gave 3-d locations, the line segment between them was projected into each view. If the segment seemed badly oriented (i.e. not following the central axis of the fly's body) new points were generated. In this way a vector parallel to the 'forwards' direction was obtained. To get a vector approximately parallel to the 'outwards' direction, points were clicked on the wingtips, assuming symmetry of the wings about a plane bisecting the fly. (An illustration of these points is given in Figure 2.8.)

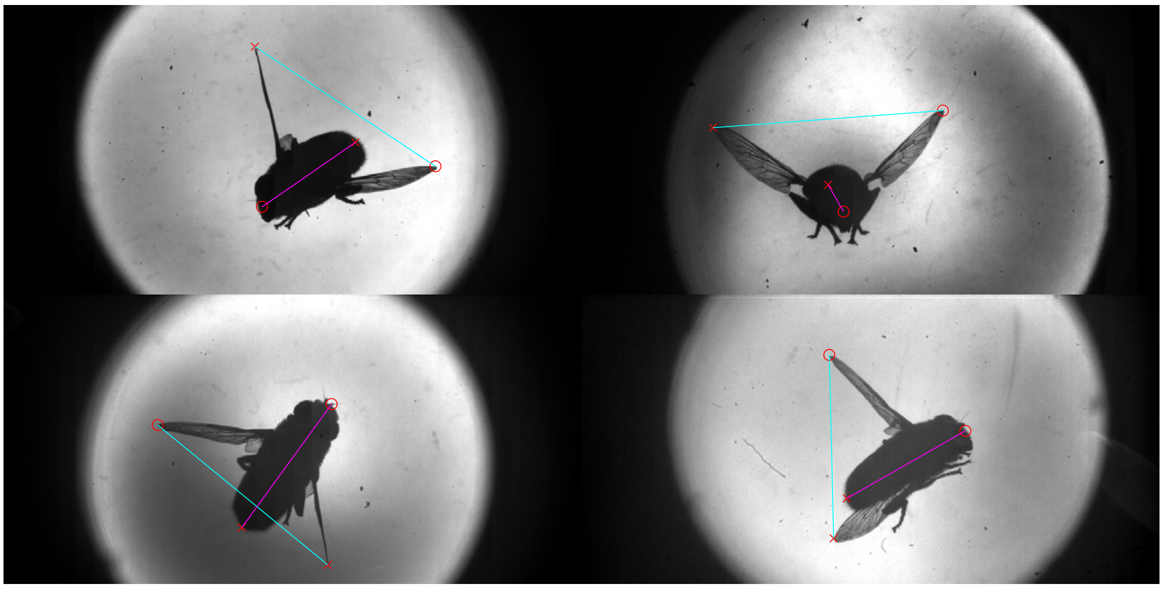


Figure 2.8: Lines between manually identified points used for fly 'forwards' and 'outwards' direction, required to construct axes.

A 1 unit vector in the 'forwards' direction was taken as a basis vector for x (call this $\vec{u}$, as in Figure 2.8). To get a vector in the outwards direction, the estimate from the wingtips (call this $\vec{v_0}$) was nudged to the closest vector orthogonal to $\vec{u}$ using the following formula:

$$\vec{v_{\text{new}}} = \vec{v_0} - \langle \vec{v_0}, \vec{u} \rangle \vec{u} \tag{2.1}$$

This was normalised to get a 1 unit vector ($\vec{v}$), which formed the basis for y. The third basis vector ($\vec{w}$) for z was given by $\vec{u} \times \vec{v}$, as it is a right-hand coordinate system. These vectors span $\mathbb{R}^3$ and provide a change of basis matrix A (a rotation matrix) where:

$$A = [\vec{u}|\vec{v}|\vec{w}] \tag{2.2}$$

The 3-d shoulder point was estimated in a similar manner to the head and tail. Although there is limited texture information to work with, epipolar lines can guide the selection (especially when the shoulder is clearly visible in one view, such as view 2 of set 15R), as well as the observation that a line along the leading edge of the wing should pass through the shoulder (an assumption used in [21]). The shoulder position was estimated in both the first frame and the last, and the movement between these points was taken to be the motion of the body during the sequence. For simplicity, a constant linear velocity was assumed between these two points, so that the fly's movement is a translation along the line between them. Subtle changes in velocity and minor rotations are visible in the image sequences, but as will be discussed later, perfect motion extraction is not essential.

In order to obtain a coordinate system centred that stays centred on the moving shoulder, the coordinate system transformation for a given frame must subtract the current position of the shoulder. If the shoulder position estimate for the first frame f_0 is X_{S_0} and motion vector between frames is $\vec{m}$, then the position at frame f is simply:

$$X_S(f) = X_{S_0} + (f - f_0)\vec{m} \tag{2.3}$$

For a point p in standard world coordinates at frame f, its position relative to the moving wing shoulder is given by:

$$p_{\text{wing}}(f) = A^\top (p - X_S(f)) \tag{2.4}$$

This is a transformation between the static world axes and desired moving axes which encode positions as lengths in intuitive forwards, outwards and downwards directions. An illustration of the axes is provided in Figure 2.9.

With these new axes, it is easier to understand the movement patterns of keypoints because of their intuitive representation and that the effects of rigid body motion have (largely) been removed. The motion of template keypoint 1 (near the wingtip, clicked for every frame in the sequence) is shown in Figure 2.10 in the original axes, with the effects of the moving body causing a clear drifting effect. In Figure 2.11, this same 'point trace' is viewed from the moving wing axes, in which the (quasi-) periodic motion patterns are obvious. In both figures frames are marked in which the wing is either at the top of the upstroke or bottom of the downstroke, as judged by eye from the movement of point 1 in the image sequences.

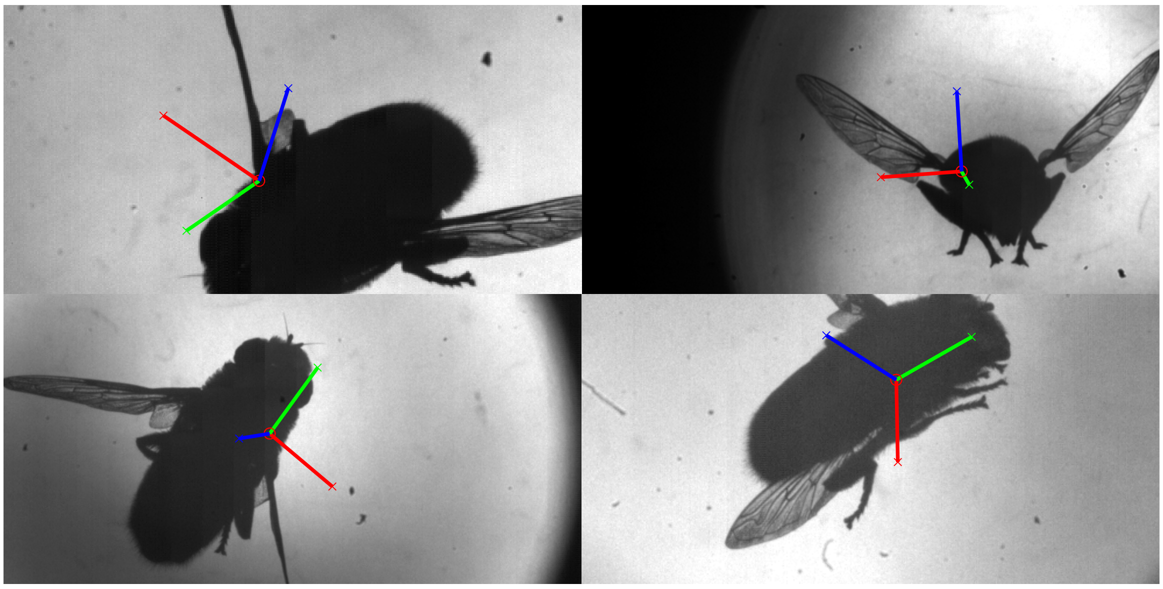


Figure 2.9: Illustration of fly axes projected onto fly images, centred on shoulder: forwards, out and up vectors (green, red, blue) correspond to X, Y and -Z.

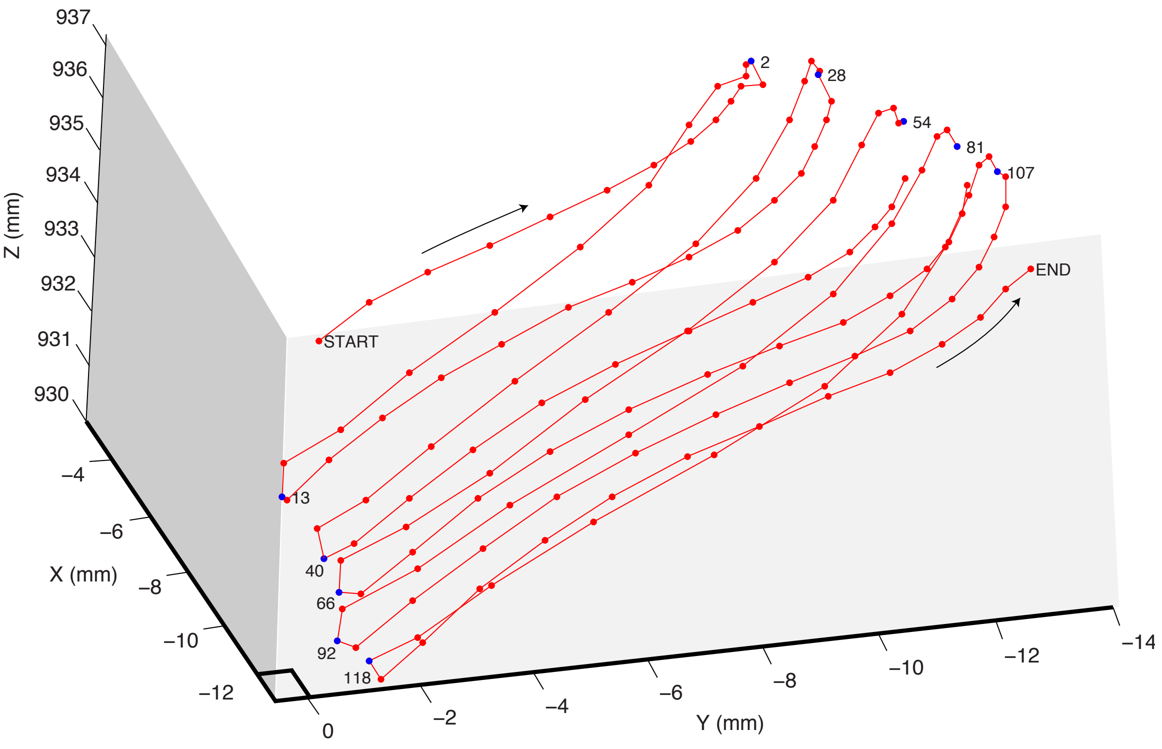


Figure 2.10: 3-d locations for point 1 (near wingtip) in original world coordinates - identified from clicked points in all frames.

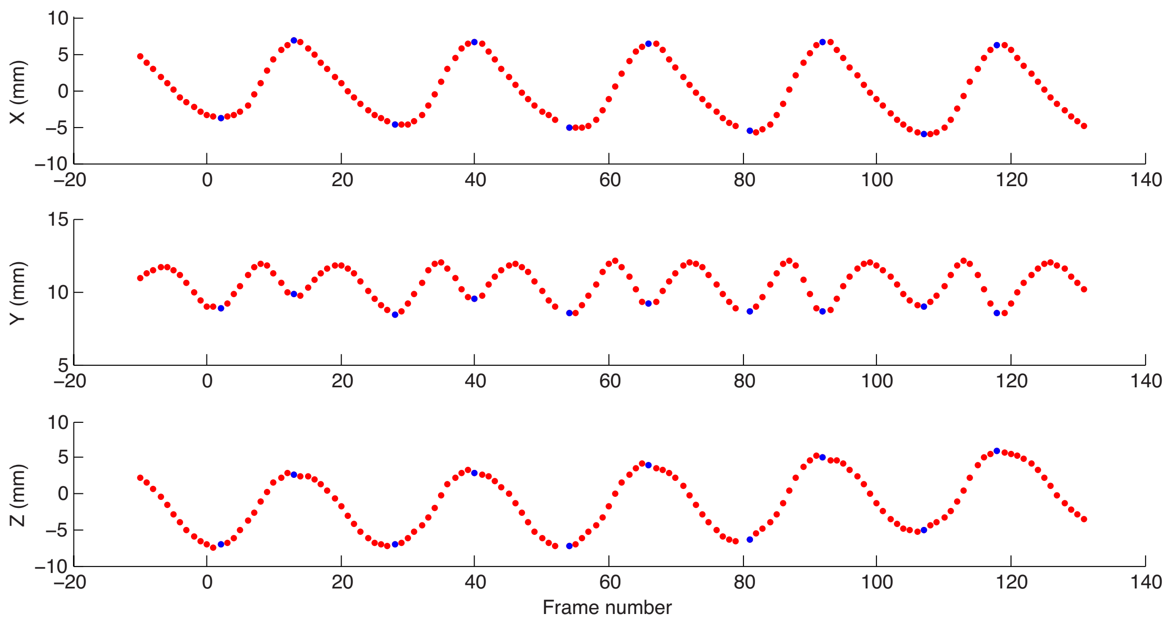


Figure 2.11: X, Y, Z values in wing coordinates for point 1 across all wingbeats.

2.5 Patterns in spherical coordinates

As points on the wing surface are essentially rotating around the wing shoulder, representing their position in spherical coordinates allows motion patterns to be simplified further. Figure 2.12 illustrates the relationship between cartesian and spherical coordinates. Theta (θ, the azimuth) is the angle from the x-axis in the x-y plane. It will be 0 for a wing point directly in front of the fly and 90 degrees for a point directly out from the shoulder, and can be thought of as the angle ‘outwards’ from the front of the fly. Phi (ϕ, the elevation) is the angle from the x-y plane. It can be thought of as the ‘downwards’ angle to a surface point. Finally R (the radius) is the distance between a wing point and the wing shoulder.

In Figure 2.13, the positions of point 1 across the sequence are shown in spherical coordinates. It is clear from the plots that the point takes its maximum and minimum azimuth and elevation values at the stroke extrema. During the downstroke the point moves forwards and downwards (theta decreases, phi increases) and during the upstroke the point moves backwards and upwards (theta increases, phi decreases). Meanwhile radius values (R) remain within a narrow range of values approximately 12mm from the wingtip. This indicates that the point is very close to tracing a spherical path.

Minor differences can be observed between the strokes. For example the amplitude of the theta waveform appears to increase during successive wingbeats. At the same time, the phi waves seem to drift higher. This latter difference is most likely an artifact of oversimplification of rigid body motion. Assuming the drift indicated an accelerating forwards roll, the spherical coordinates were replotted after introducing a suitable rolling motion to the wing coordinate frame - using peak height differences as current roll angle. Figure 2.14

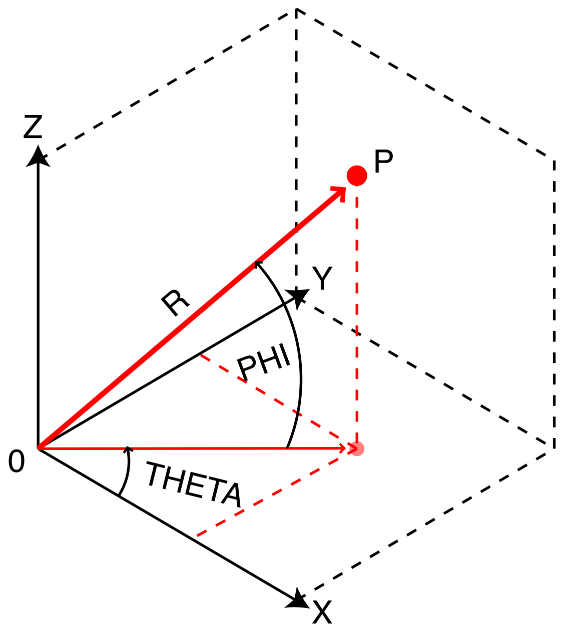


Figure 2.12: Relationship between cartesian (X,Y,Z) and spherical (THETA,PHI,R) coordinates.

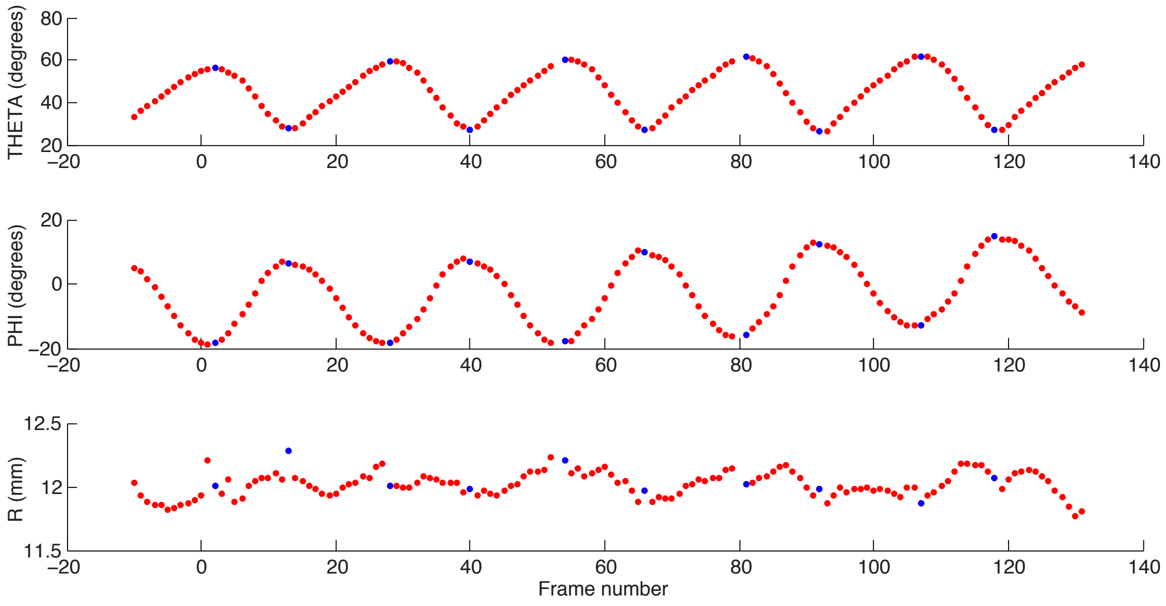


Figure 2.13: Spherical coordinates from wing axes for point 1 across all wingbeats.

shows that this does remove the drift from phi without significantly impacting theta and R. While manual coordinate frame adjustments such as this can increase the periodicity (and therefore predictability) of the observed waveforms, a prediction model was sought that can work with inter-stroke differences.

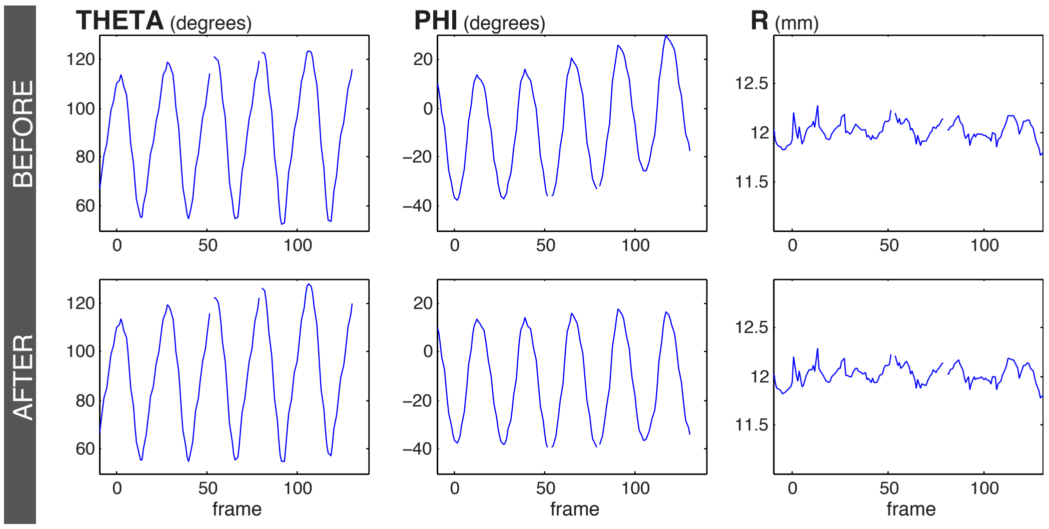


Figure 2.14: Spherical coordinates before and after removal of accelerating roll.

To investigate the motion of other parts of the wing, the other keypoints were clicked throughout the first wingbeat. The observed spherical coordinate patterns are similar, with theta and phi appearing to be reflections of each other (while R remains approximately constant during the beat), with the main difference being that their extrema lag to varying degrees behind point 1. These theta and phi values are shown in Figure 2.15. Gaps exist for certain points in a number of frames where the point could not be identified in more than one camera. (This would either be caused by occlusion or poor local contrast.)

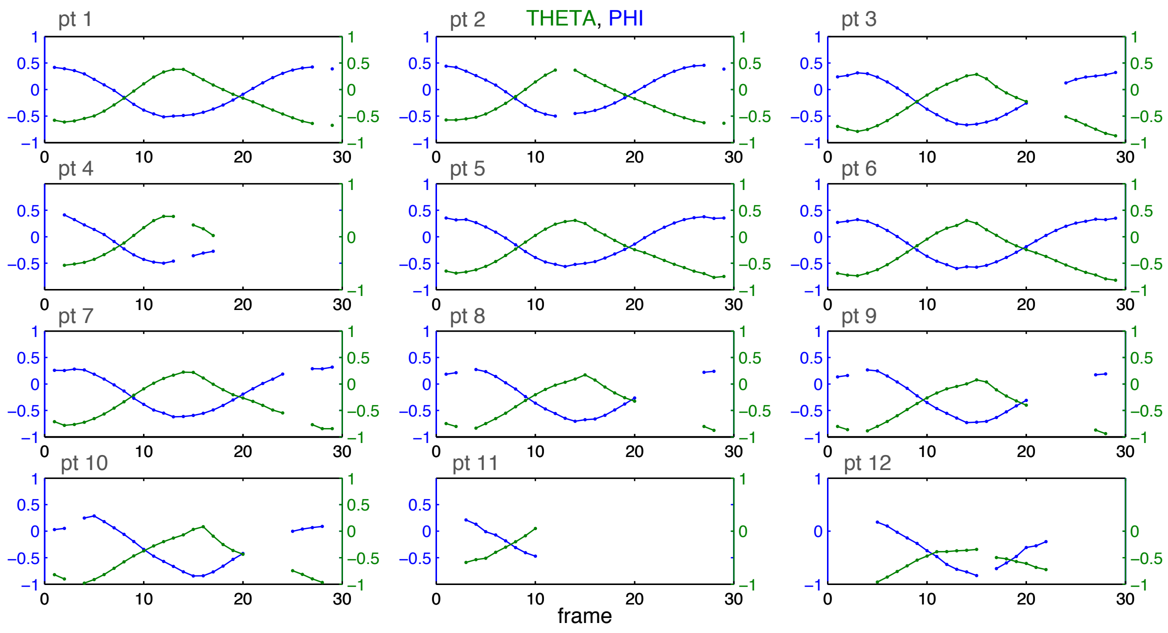


Figure 2.15: PHI and THETA (in radians) for 12 wing keypoints.

2.6 Building a prediction model

The approximately sinusoidal nature of the theta and phi values suggest that a simple prediction model could fit sine waves to individual wing surface points, using some set of position measurements. But the repeating patterns are not in fact sinusoidal: if they were, the points would move back and forth along an arc rather than the more complex figure-of-8-like motions that they trace out (shown in Figure 2.16). A sine-fit approach that assumes perfect periodicity would also fail to adapt to wingbeat differences, and prediction accuracy would deteriorate under conditions such as the amplitude variation and drift observed for point 1 in set 15R.

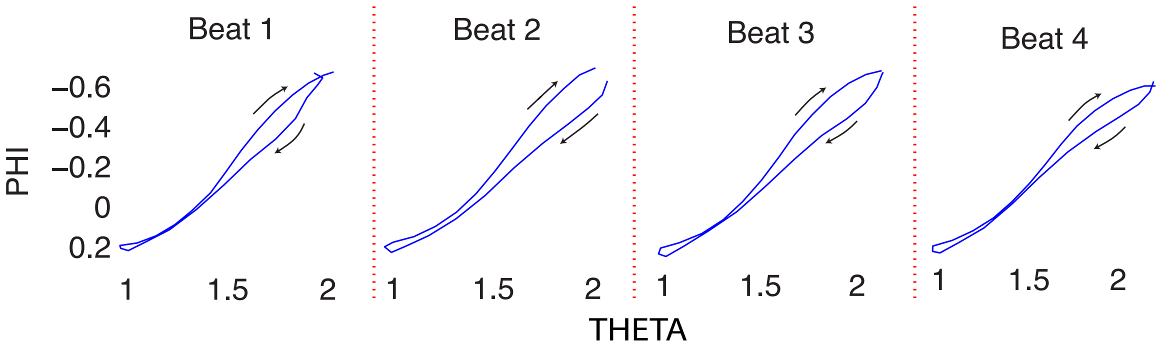


Figure 2.16: PHI and THETA traces on sphere for first four wingbeats of point 1.

The assumption that consecutive wingbeats look very similar, and that gradual trends emerge from wingbeat differences can be used within a prediction model. One approach would be to attempt to develop a quasi-periodic fitting procedure that encapsulates trends in a small number of parameters, estimated from previous wingbeat measurements, and can cope with noise and gaps in the data. This is a complicated procedure (as demonstrated by Eilers [83], who investigated such fitting to noisy data) that requires careful manual supervision and is very sensitive to the selection of initial parameter values.

A much simpler procedure was chosen that maximises generality at the expense of requiring some manual intervention, and makes use of the similarity of adjacent wingbeats. This will be referred to as the 'normalised stroke model'.

The normalised stroke model represents theta values throughout a stroke as a spline and predicts the values for the next stroke (of the same type) by stretching the spline horizontally and vertically to interpolate clicked points at the extrema. Phi values are predicted in the same way. R values are obtained by linear interpolation between start and end values (obtained from the clicked extrema).

For example, the clicked points of set 15R beat 1 provide a discrete set of 3d measurements that can be used to predict 3-d positions for the two strokes of beat 2. To predict the downstroke for point 1, a smoothing cubic spline was first fitted to the downstroke theta values[2]. Next, the domain and the range of the spline were scaled in a process that will be referred to as 'normalisation'. The domain (set of frame numbers) is linearly scaled such that the frame at the first extremum (top of the previous upstroke) is scaled to 0, and the frame at the second extremum (bottom of the downstroke) is set to 1. Thus frames in between are scaled to positive numbers less than one. The range (theta axis) is also linearly scaled such that the value achieved at the first extremum is 0 and at the second is 1.

By clicking on point 1 in frames identified as the corresponding extrema of the next downstroke, 3-d points (and thus theta, phi and R values) were obtained that this stroke should interpolate. This interpolation was performed by rescaling the normalised spline. First the domain is linearly scaled so that 0 and 1 map to the new start and end frame of the stroke, and the range is linearly scaled so that 0 and 1 map to the end and start frames theta values. Figure 2.17 illustrates each of these steps. Downstroke phi values, and upstroke phi and theta values are predicted in the same way.

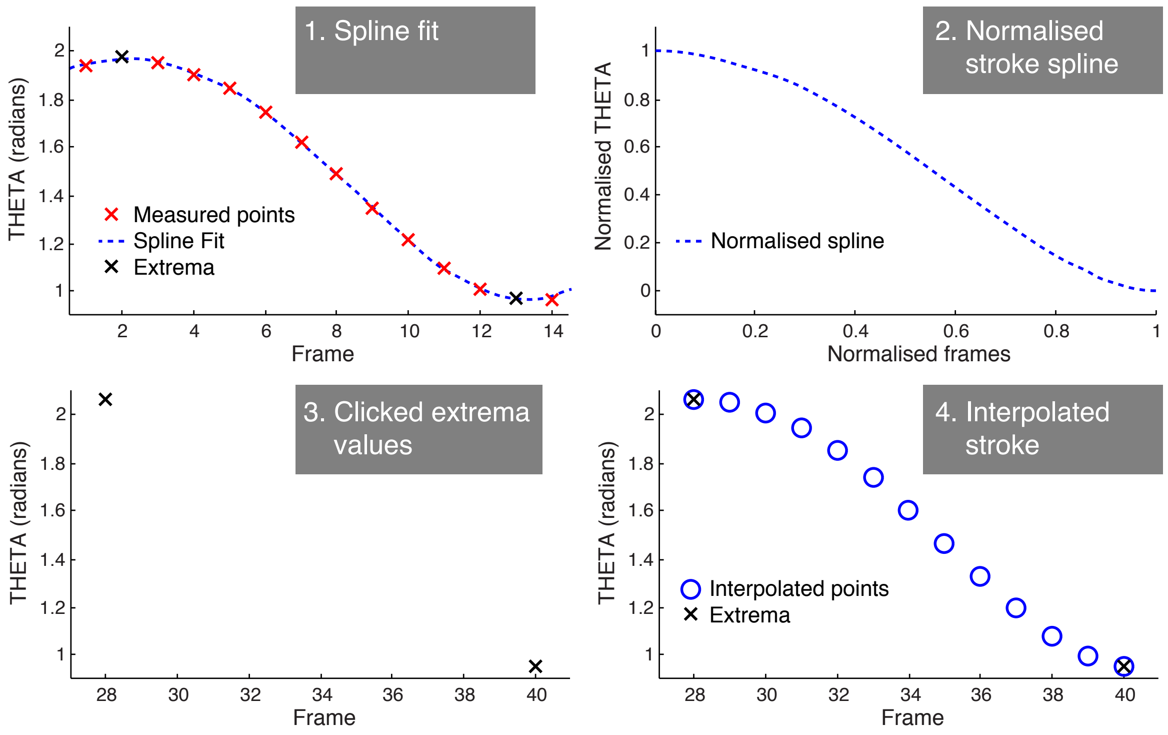


Figure 2.17: Normalised stroke model steps, for predicting second downstroke theta values (point 1)

The success of this spline normalisation and scaling method relies on theta and phi taking their maximum and minimum values at or near the (manually identified) extrema. If the difference in values at extrema in the reference stroke is δ_1, and the difference for the predicted stroke is δ_2, then the ratio δ_1/δ_2 gives the 'stretching' factor in the rescaling step. This is the desired scaling for the case where δ_1 and δ_2 are good estimates of the stroke amplitudes, but will cause unwanted amplification or shrinking otherwise. For this reason

[2]using Matlab's *csaps* function with default smoothing, weights and roughness parameters, and extrapolated using *fnxtr* for zero-valued second derivatives outside the basic interval.

Set 15 extrema	2	13	28	40	54	66	81	92	107	118
15R clicked frames	2	13	28	40	53	66	81	92	107	118
15L clicked frames	3	11	29	41	55	68	81	91	107	117

Set 23 extrema	12	25	39	52	66	79	93
23R clicked frames	11	24	40	51	67	78	94
23L clicked frames	14	24	37	51	68	78	95

Table 2.2: Extrema and measurement frames for each hovering set.

it is important that extrema are identified with care. As mentioned previously, extrema for point 1 were estimated by eye: choosing the frames where the point seemed to reach its farthest ‘forward’ location and farthest ‘backward’ location. While this choice can be ambiguous (depending on time resolution), this approach was chosen in order to minimise the number of manual ‘clicks’ required. For simplicity, these same extrema frames were used when making predictions for other points, assuming the delay before their ‘true’ extrema was small compared to the stroke duration.

A helpful property of the normalisation process is that the frames at which points are measured do not have to be the extrema themselves, but can be frames near the extrema. This allows for the eventuality that keypoints are occluded or otherwise unidentifiable at extrema frames - a common occurrence in the data sequences used. Using these alternative measurement frames requires identifying their normalised frame value. For Set 15R, for example, the fourth stroke extrema are frames 40 and 53, but points were clicked at frames 40 and 54 (due to inadequate visibility in frame 53). In this case, frame 53 would be equivalent to normalised frame 1.0, while frame 54 is equivalent to 1.0679. The spline values at these points gives the normalised theta and phi values for frames 53 and 54. The rescaling process can be carried out accordingly so that the measurement at frame 54 is interpolated.

Table 2.2 shows the frames identified as extrema and those used for keypoint measurement in each of the hovering sets.

This process predicts the future locations of a single point on the wing surface. In order to predict an approximate shape of the whole wing, the predictions for a small subset of the wing keypoints were combined. Five points (1, 3, 5, 8 and 10) were chosen from the previously identified set of twelve. They are well-spaced, easier to identify and less frequently occluded than several of the other points, and cover a large area of the wing. The set is small enough to be quick to find at extrema frames but large enough to capture a reasonable approximation of wing shape. A wing surface can be estimated from these five keypoints by constructing a TPS map in the same way as used for the reference 3-d wing. Figure 2.18 shows the result of this process when used to predict spherical coordinates for each of these points in the second wingbeat, given measured (i.e. ‘clicked’) locations in the first.

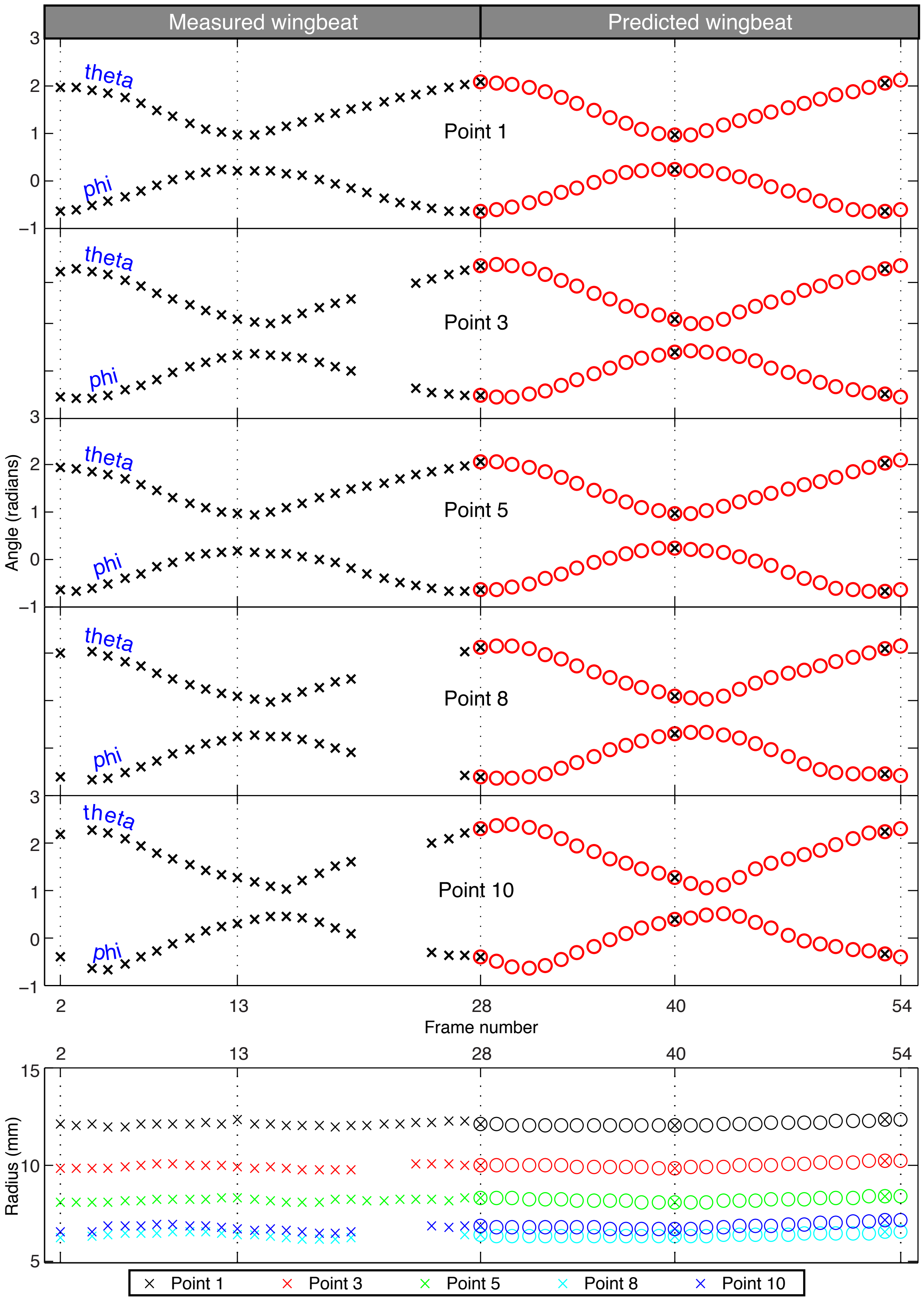


Figure 2.18: Spherical coordinate predictions for 5 keypoints. Crosses indicate ‘clicked’ points, and circles indicate predictions from normalised spline model.

The normalised stroke model is well suited to scenarios in which stroke amplitude changes or coordinates drift between strokes, and allows any stroke shape to be captured without the need for an underlying function to be explicitly specified. Predictions from this model can be used to guide measurements of the true wing-shape. In the next chapter, an algorithm will be introduced which does this, and sets up an iterative framework of stroke prediction and stroke measurement to obtain wing shapes over an arbitrary number of wingbeats.

2.7 Discussion

In developing the above procedures for segmentation, building a reference shape and making shape predictions, various alternative approaches were tested.

Silhouettes

Silhouette extraction may have benefited from pre-processing which removes image features from the background, such as the black dust which occasionally overlaps with the wing membranes. Appendix B describes the development of such a background subtraction method. This method effectively flattened out the background intensities in individual images (and compensated for illumination variation between images) but it created unwanted artifacts in the 'foreground' fly pixels. This made it unsuitable as either an aid to segmentation or a tool to create uniform wing appearance across images. If the hoverfly sequences had also been accompanied by camera images of the background without the hoverfly present, a version of the background subtraction method could have been used to improve silhouettes (by removing strong edges that aren't on the fly). As it stands, the current extraction method produces silhouettes that in most images stick very close to the fly boundary. Some silhouettes have significant gaps or 'appendages', as shown in Figure 2.19, and a very small number cut out sections of the wings (on frames where the wings stray into the dark boundary section), but even these images provide some useful information, as will be seen in the following chapters.

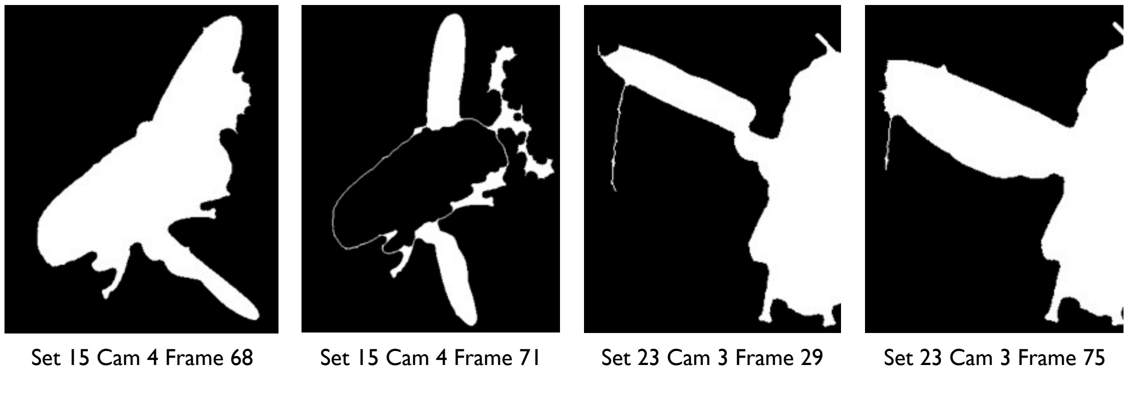


Figure 2.19: Examples of poorly segmented silhouettes.

In extracting the silhouettes, similarity of silhouettes between adjacent frames and between frames at the same stroke phase in adjacent wingbeats (i.e. temporal coherence) was not used. A method such as the combination of an active contour with watershed pre-segmentation (as proposed by Gouze et al [53], mentioned in Chapter 1) could make use of some of this temporal information. Whether it could yield boundaries as accurate as those obtained from the image processing procedure presented here would require further research.

Reference shape and texture

The reference wing shape and texture would ideally be constructed using an image of the wing taken under controlled conditions which ensure perfect flatness of the wing surface and uniform illumination which would reveal any texture present on the wing membranes. As only one image is required, a higher resolution camera could be used. This could be calibrated so as to allow separations between keypoints to be measured in millimetres directly from the image. The strategy adopted here obtained these distances for each wing by traversing a thin-plate spline surface which had been verified to be a good fit by eye using projected spar points. This is a more convoluted process in which the level of precision is difficult to quantify. Measurement accuracy will be discussed in detail in Chapter 6. In the absence of independently obtained flat wing measurements, the strategy could be extended so that point separations are obtained from surface fits to multiple frames. This would increase the amount of manual work required for each set (which is why only one frame was used) but could help increase the accuracy of the separation estimates.

Another choice made to limit manual intervention was the use of a single 'general template' for spar visualisation purposes for multiple wings. That this worked well for each of the wings tested shows that venation patterns are extremely similar between hoverflies. The general template is used for spar visualisation only, so the choice to use just one template shouldn't introduce errors into the process, however future work on other hoverflies may require a different general template if their wings are substantially different.

Choice of features

The choice to use five manually identified keypoints to make motion predictions removed the possibility of a fully automated procedure. Full automation would require that keypoints be automatically and accurately identified between images. Tests were performed with a number of affine covariant feature detectors and descriptors[3], with a particular focus on SIFT matching[4], as this showed a good ability to find and match features between consecutive images of bright, face-on wing images. As with other features, SIFT feature detection and matching can be controlled by parameters that determine how 'strong' features must be in order to be included and how similar the descriptors must be to be matched. In trial

[3]using code provided by the Visual Geometry Group at Oxford University on their Affine Covariant Features webpage at http://www.robots.ox.ac.uk/~vgg/research/affine/

[4]using a matching routine provided by the VLFeat open source library at http://www.vlfeat.org

and error testing of parameter values, it was found that only the most bright and face-on images yielded a dense set of matches, but even these always contained outliers (including some wing features that are erroneously matched to features on the fly body). As the wing image area decreases in subsequent frames, the number of features found and correctly matched quickly drops to zero. Figure 2.20 shows an example failure of SIFT matching between large area wings in different views with different illumination properties: only two matches are found, and one of these is an outlier. In this image, the five keypoints used in the prediction model are easily identified by eye.

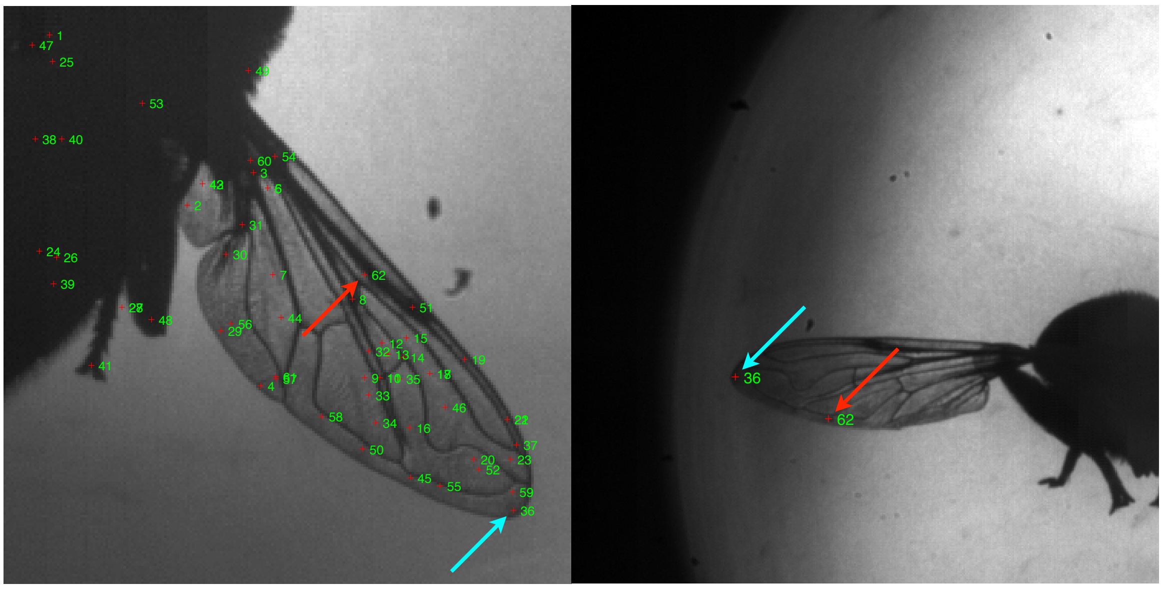


Figure 2.20: Example SIFT matching result. Left image: reference image showing all SIFT features found. Right: Of 2 matches found, one is correct (36), and one is incorrect (62).

The disappointing performance of SIFT on this data set shows that the texture information present in these backlit images is not adequate for a naive feature matcher to localise wing points robustly. There may be scope for further research into increasing the number of matches using something like Ferrari et al's 'image exploration' technique [11], which anchors on an initial feature set to find others. This however would require a good set of reliable matches for initialisation. It may be easier to find such sets if the images can be suitably processed to compensate for differences in background illumination and spar contrast, so further work would be required in image pre-processing as well.

The PMVS multi-view stereo algorithm of Furukawa and Ponce [40] was also tested to see if it could combine automatic feature recognition (of Difference-of-Gaussian and Harris features) across multiple calibrated views to produce 3-d shape. Figure 2.21 shows 3-d patches obtained for a frame from set 15. The algorithm was successful in building up some of the wing surfaces, and for the most part managed to weed out background features, but there are large gaps in both wings for which patches have not been found. The patches themselves are also quite large in relation to the wing size, so can't be expected to provide a very high resolution mesh. Furukawa and Ponce's 3-d motion capture algorithm [78], which takes a dense PMVS mesh as starting point, was therefore deemed inappropriate for these hoverfly data sets.

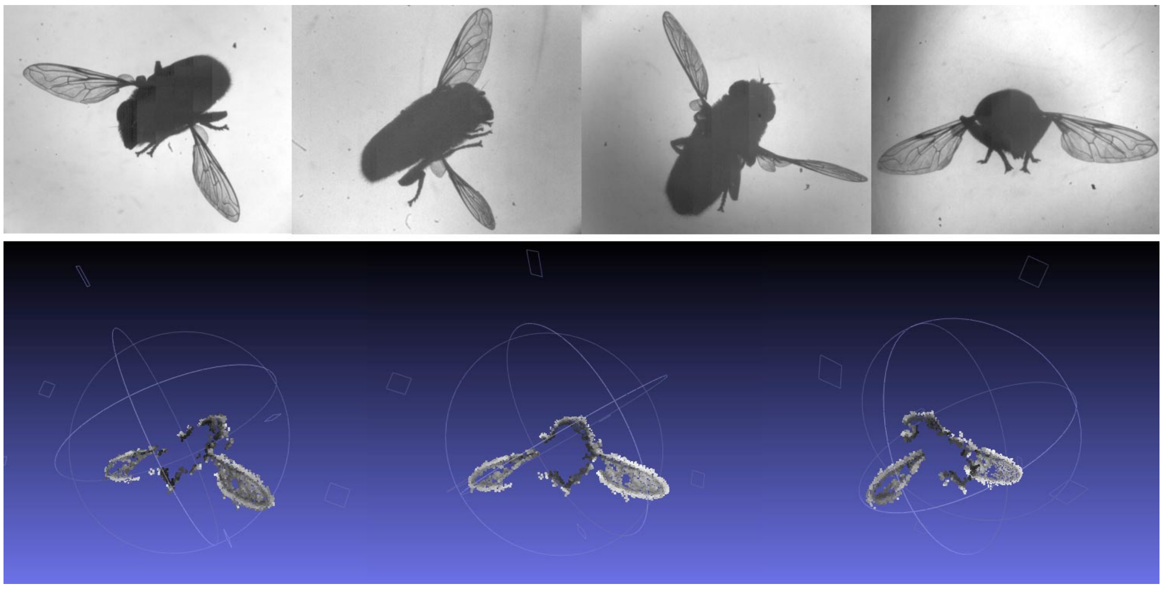


Figure 2.21: PMVS algorithm results. Top row: input images (Set 15, frame 16). Bottom row: three views of 3-d patch model.

Wing coordinate system

In setting up the wing coordinate system, a number of assumptions were made for the sake of simplicity and limiting the manual intervention. The position of the wing shoulder is only measured twice for an entire sequence (at the start and end frames) and was linearly interpolated for intermediate frames. A decent estimation of the wing shoulder position is necessary in order for the assumption that points have a constant ‘radius’ coordinate to hold. This simple estimate was effective for the data sets examined, but for longer sequences in which there is a less steady motion, additional measurements will be required. This may be a matter of clicking on additional intermediate frames, but there is also potential for future work into tracking the wing shoulder. Starting points for such a task would include attempting to identify it as a very narrow strip of the visual hull, or attempting to track it using one of the visual tracking techniques mentioned in Chapter 1 (such as optical flow).

For set 15R another method of estimating the wing centre was tested. Using the assumption that keypoint 1 traces out a spherical path relative to the moving shoulder, a non-linear least squares estimation algorithm (Matlab’s *lsqnonlin*) was used to simultaneously fit a sphere centre and constant linear velocity that agreed with ‘clicked’ data for keypoint 1 over several wingbeats. This method requires considerably more manual intervention, however, and so was not used in the final procedure.

Choice of motion model

The use of scaled splines to represent stroke motions of keypoints was based on the similarity of adjacent wingbeats. Before progressing to this solution, a number of alternative prediction methods were tested that predict a keypoint location in a single frame based on least squares sine wave fits to phi and theta coordinates over recent frames. This would be consistent with the assumption that the coordinates behave locally like sine waves, and would not be affected by movement characteristics like stroke amplitude variation or drift. These fits were attempted using a varying number of measurements (from clicked data) and by using a number of ways to initialise the sine wave (such as constant initialisation, or using the fit from the previous frame, or from the equivalent stroke in the previous frame). These fits often gave good predictions for mid-stroke frames, but could give poor results near the extrema.

By employing limited manual supervision near stroke extrema, where overall wing feature visibility is at its lowest, the normalised stroke model provides a simple and effective solution for predicting wing motion.

Chapter 3

Keypoint tracking for wing interior

3.1 Overview

This chapter develops the core wing tracking algorithm, which captures the deformation of wing interior over an arbitrary number of wingbeats. The full algorithm, given the pre-processing of Chapter 1, consists of the steps in this chapter (which will be referred to as the 'keypoint tracking algorithm') and the steps in the next chapter which extract the wing boundary (or 'rim') and use these boundary points to get a better estimation of wing shape.

The keypoint tracking algorithm begins with an iterative procedure consisting of three processes which will be referred to as prediction, tuning, and smoothing. These are applied in sequence on a stroke by stroke basis, and each generates 3-d shape information for the wing. The prediction process uses the motion model from Chapter 2 to obtain starting estimates for wing shape for a given stroke, given measured shape two strokes earlier. The tuning process refines these predictions using image registration with a template. Keypoint locations are extracted from the results and used as inputs to the smoothing process, which identifies outliers, smoothes over the results and fills in missing points. This smoothing process operates on 'tuned' measurements from the current stroke and two adjacent strokes, and the resulting shape is then used to create new predictions for the next corresponding stroke. Figure 3.1 illustrates the progression of the sequence for *N* wingbeats.

This iterative procedure is followed by a process that introduces the wing inextensibility constraint to identify further outliers and create an implicit wing fit. The algorithm then concludes with a final smoothing stage. Each process will be discussed in turn in the following sections.

3.2 Prediction

As Figure 3.1 shows, the algorithm begins with keypoint predictions for the first three strokes. For the sake of keeping the preliminary manual steps to a minimum, a single set of reference stroke patterns was used for predicting these initial strokes. This consisted

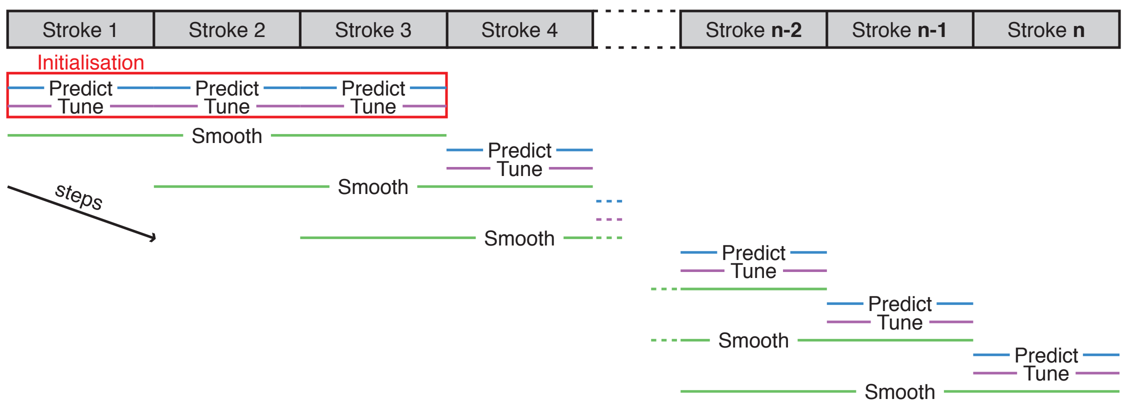


Figure 3.1: Order of prediction, tuning and smoothing processes for a full wing sequence.

of the theta and phi splines generated for the first downstroke and upstroke of the five prediction keypoints (1, 3, 5, 8 and 10) of set 15R. If the reference patterns had not been used, approximately 600 points would need to be 'clicked' for each wing (for 5 points in 4 cameras in 30 frames).

The assumption underlying the use of reference patterns is that different hoverfly wings and their hovering flapping motions are very similar. Variance in body and wing size would therefore not significantly affect theta and phi angles, given a well-constructed wing co-ordinate frame. The radii of the points may well change, but the linear interpolation of measured radius values should account for these differences.

For a new flight sequence, the first step is to establish whether it begins with a down-stroke or an upstroke. This determines the treatment of the first three strokes. If it begins with an upstroke, for example, predictions use the reference upstroke, then reference down-stroke, then reference upstroke again. Figure 3.2 shows an example of prediction results half-way between clicked points in the first stroke of set 23L.

From the start of the fourth stroke onwards, predictions are based on shape estimates from the smoothing process rather than reference patterns. Stroke 4 is based on stroke 2, stroke 5 is based on stroke 3, and so on. So that prediction (and tuning) estimates are only recorded once for each frame, the extrema frames are classified as belonging to one and only one stroke. Strokes were therefore divided such that they include their first extremum, but exclude the second.

3.3 Tuning

The tuning process refines the predictions for each frame in a given stroke, one frame at a time. The five keypoint predictions are used to warp images from this frame onto the reference image (see chapter 2). By registering these images, the full set of 12 keypoints (or indeed an arbitrary number of 2-d template points - see Section 3.7) can be localised in

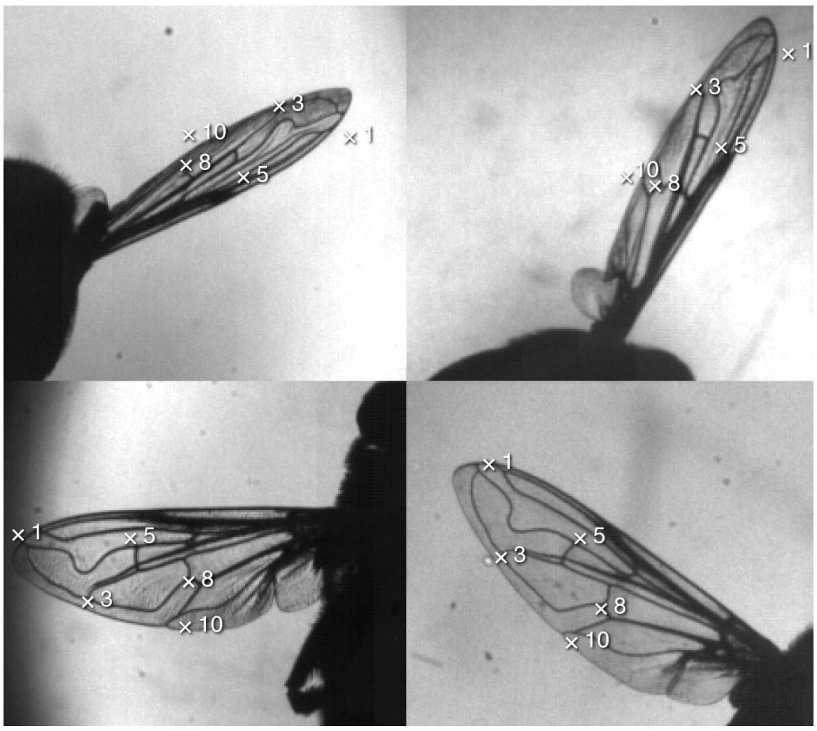


Figure 3.2: Projections of predicted keypoints for Frame 19, Set 23L (Stroke 1).

the warped image, and then localised in the original image by unwarping. Performing this for two cameras, provides an estimate of the 3-d locations for each keypoint.

By restricting the process to two cameras texture information was discarded from half of the views. In the hoverfly sequences analysed, however, there are very few frames with more than two views suitable for registering with the template. Successful registration requires that a large area of the wing surface is exposed to the camera. By including views with poor wing visibility, the registration attempt will lead to incorrect correspondences, and therefore poor 3-d localisation. This creates an additional burden on outlier detection. The outlier detection method, described in Section 3.4.2, discards keypoints for a given frame if marked as an outlier - so an accurate 2-d localisation will be discarded if it happens to be paired with an inaccurate one. It is therefore important to select good views at the tuning stage.

3.3.1 Camera Choice

A ‘good’ camera view is one in which a large wing area is visible to the camera. This allows greater visibility of wing features as they will be well-spread and the membranes will be approximately orthogonal to the background illumination, maximising contrast. Such views therefore have the best chance of successful registration with a wing template. The projections of the 3-d predicted keypoints can provide a measure of wing area. The polygonal area defined by the hull of the prediction keypoints was used: point 1 $\rightarrow$ point 5

$\rightarrow$ point 8 $\rightarrow$ point 10 $\rightarrow$ point 3 $\rightarrow$ point 1.

Figure 3.3 shows the polygons that correspond to the predictions from Figure 3.2. In this case the largest polygons are in views 3 and 4, so these two views will be used for image registration. A further distinction is made between the selected two views: the ‘best’ view (highest polygon area), in this case view 4, is marked for special treatment.

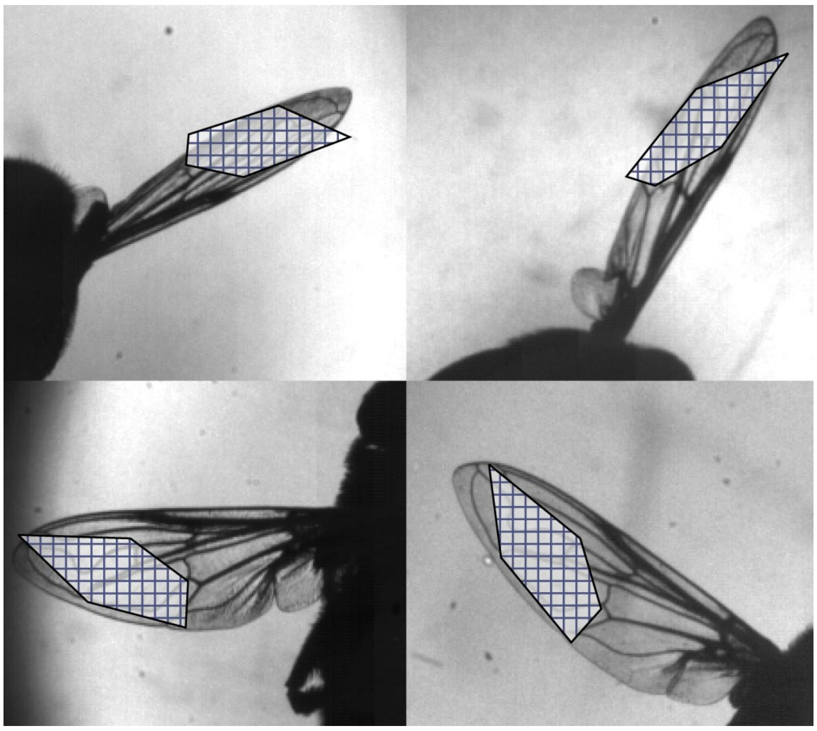


Figure 3.3: Polygonal area from predictions indicates ‘best views’. for Frame 19, Set 23L (Stroke 1).

3.3.2 Wing Segmentation

As a preliminary step before wing image registration, bounding boxes for the wings in each frame were identified, and used to crop the images so they would show only one wing. This served three main purposes: first, it provided a simple way of separating left and right wings so they could be processed independently; second, the similar appearance of the wings may adversely affect image registration if they’re both present; and third, less processing is required in the registration step if there is a smaller number of pixel displacements being optimised.

In the first frame of the sequence, points were clicked near the centre of the left wing. A bounding box (of 227×227 pixels) centred on this point was observed and, if necessary, moved by clicking again. Then the same was done for the right wing. The same box locations were viewed in the next frame, and adjusted if required, but for most frames this was not necessary.

Figure 3.4 shows wing bounding boxes that have been identified for a frame in set 15. Figure 3.5 shows the cropped right wing images from all four views over the course of a wingbeat.

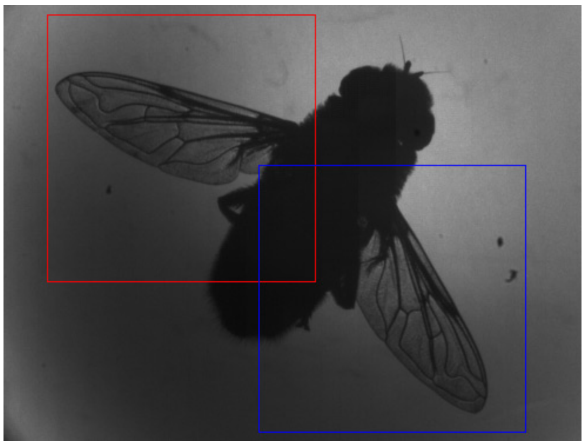


Figure 3.4: Manually identified bounding boxes containing left (red) and right (blue) wings - camera 3, frame 111.

3.3.3 Template registration

As mentioned previously, prior to registration the wing images must be warped according to correspondences between predicted keypoints and template keypoints. This will be referred to as a ‘coarse warp’, since this only provides a very rough alignment which depends on the accuracy of the predictions. Of the two views used for registration, the ‘best’ view will be referred to as view A, and the other as view B (and corresponding images as image A and image B). The steps in this process are illustrated in cartoon form in Figure 3.6, and described below.

The coarse warp is first performed on image A. The transformation for this warp uses a thin-plate smoothing spline that maps the five prediction keypoints from the template to the projections of their predicted 3-d locations in view A. This spline was calculated using the *tpaps* function (as before) with default smoothing parameter. It provided the ‘backwards’ transformation required by Matlab’s *imtransform* function to warp the image.

Next, the coarse-warped image is registered with the template. This provides the apparent displacement of each pixel in the template image, called the *deformation field* or *optical flow*. An implementation of the discrete Markov Random fields registration algorithm by Glocker et al. [61][1] was used. This algorithm fits the deformation field using ‘image pyramids’ (blurred and downsampled versions of the initial images) and a control grid of varying size. The coarse-to-fine hierarchy starts by estimating the deformation vectors at an initial sparsely sampled set of uniformly distributed control points on the lowest resolution image (the top level of the pyramid) and interpolates intermediate vectors using cubic b-splines. The control points are then sampled at increasing densities, adding updates to the deformation field. The next image level in the pyramid is used to start the process again. This is an

[1] *Drop* software, authored by Glocker and Komodakis is available at http://www.mrf-registration.net. Results in this thesis were generated by version 1.05.

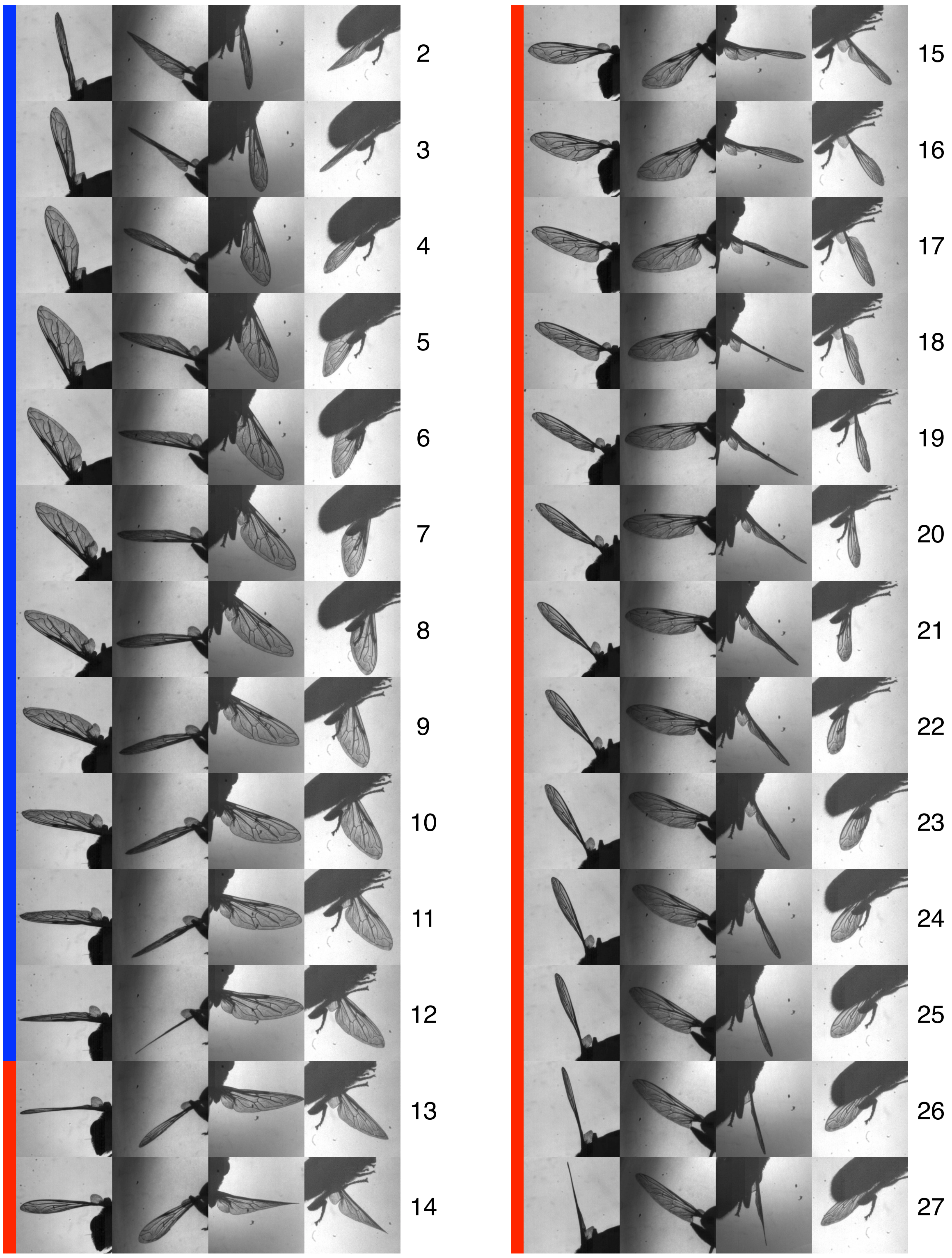


Figure 3.5: Right wing in each view for all frames of 15R beat 1 (frames 2 - 27). Downstroke and upstroke indicated with blue and red, respectively.

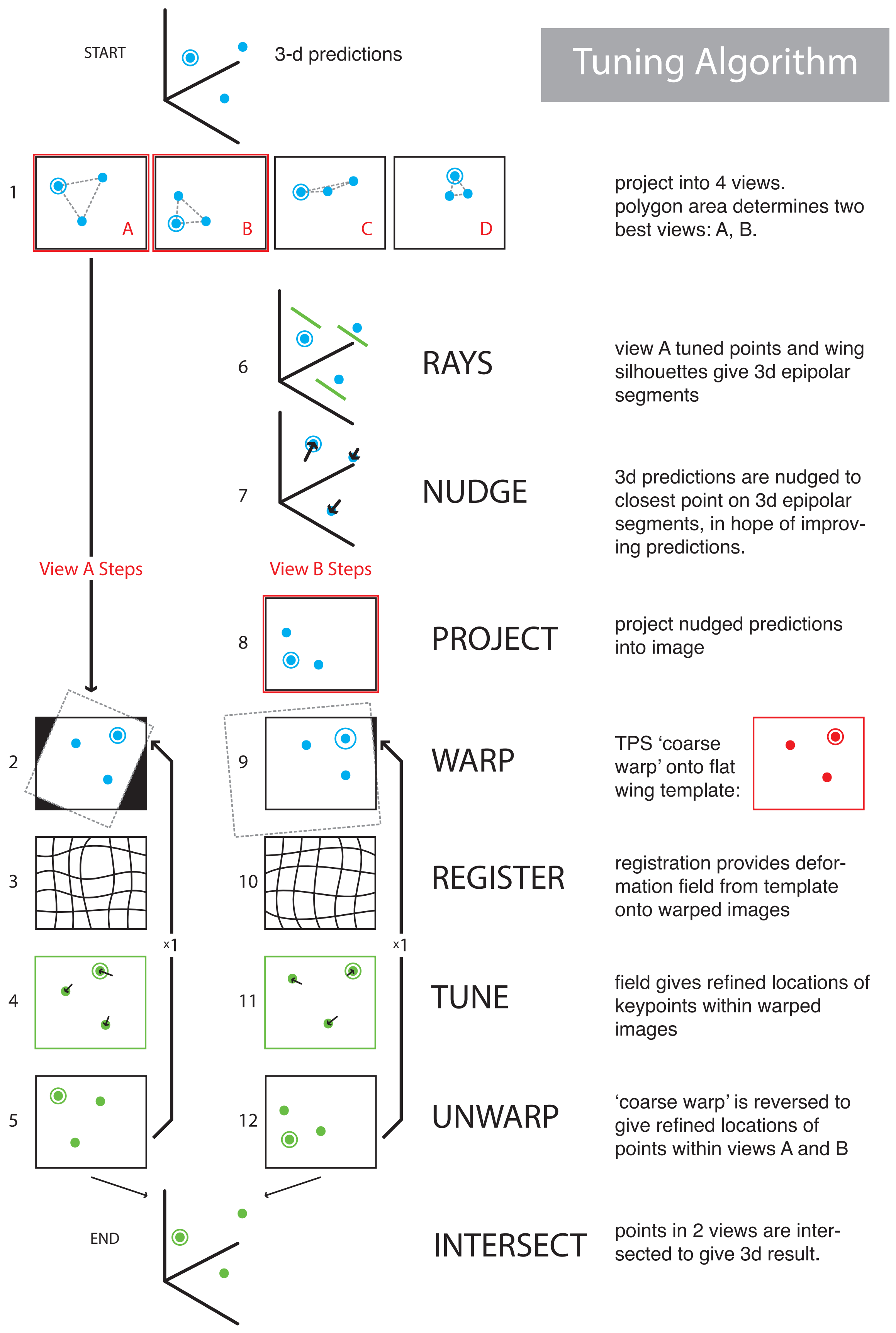

Tuning Algorithm
START
3-d predictions
1
A
B
C
D
project into 4 views. polygon area determines two best views: A, B.
6
RAYS
view A tuned points and wing silhouettes give 3d epipolar segments
7
NUDGE
3d predictions are nudged to closest point on 3d epipolar segments, in hope of improving predictions.
View A Steps
View B Steps
8
PROJECT
project nudged predictions into image
2
9
WARP
TPS 'coarse warp' onto flat wing template:
3
10
REGISTER
registration provides deformation field from template onto warped images
×1
×1
4
11
TUNE
field gives refined locations of keypoints within warped images
5
12
UNWARP
'coarse warp' is reversed to give refined locations of points within views A and B
END
INTERSECT
points in 2 views are intersected to give 3d result.

Figure 3.6: Cartoon representation of steps in tuning process.

'uncertainty-based morphing' technique in that, at each stage, uncertainty in the solution is used to locally adjust its precision. In particular, the number of possible motion vectors at each point (i.e. the density of the quantisation of the flow space) is selected according to this uncertainty.

There are a number of user-selectable parameters which the algorithm uses to estimate optical flow. This includes the functions used for the data term and the smoothing term in the cost equation, which the algorithm seeks to minimise, as well as the number of image pyramid levels, grid levels, and control point spacing, among others. Normalised cross correlation (NCC) was chosen as the image similarity measure. This is included in the data term, called the 'normalised correlation coefficient', which is the negation of the absolute value of the normalised cross correlation, as shown in Equation 3.1.

$$E_{\mathrm{DATA}}(\tau) = -\left| \frac{\sum_{x\in\Omega}(I_1(x)-\mu_1)(I_2(\tau(x))-\mu_2)}{\sqrt{\sum_{x\in\Omega}(I_1(x)-\mu_1)^2 \sum_{x\in\Omega}(I_2(\tau(x)))^2}} \right| \tag{3.1}$$

In this equation, I_1 is the coarse-warped image, I_2 is the template and μ_1 and μ_2 are their respective mean intensities. τ signifies the 2-d displacement field and Ω is the set of pixel locations. This is used alongside the smoothing term which adds a penalty based on the magnitude of the difference between displacement vectors at neighbouring control points. A three-level image pyramid was used and started with an 8×8 control point grid. Three grid levels were used, with a three-fold increase in resolution and five optimisation iterations (using the FastPD optimiser) at each step. The complete set of parameters, which were used for each data set, is provided in Table 3.1. These were selected by trial-and-error, examining the effects of different values in a GUI version of the software, which was provided with reference wing images from 15R. A single wing registration takes approximately five seconds.

The algorithm also allows a boolean 'mask' image to be provided, so that intensity patterns in the 'off' pixels are excluded from the optimisation. This is useful for the purposes of this algorithm, as a simple segmentation of the wing pixels from the background in the template can prevent background features from corrupting the flow calculation. This mask was created for each template by cropping the boolean silhouette image of the template frame to the wing's bounding box and manually switching off non-wing pixels using image editing software.

Once the displacement field between the template and coarse-warped image is established, the displacements of the twelve (subpixel-localised) keypoints are interpolated (using cubic spline interpolation with Matlab's *interp2* function). These displacements are added to the template keypoint locations to give the locations of these keypoints in coarse-warped image. Finding their locations in image A is a matter of applying the same TPS map which was used as the backwards transformation for the coarse warp. These 'tuned'

Parameter	Value	Description
image levels	3	number of image pyramid levels
grid levels	3	number of control grid levels
number of iterations	5	optimisation iterations per level
grid spacing	32.43	initial control point spacing (pixels)
data term	NCC	Normalised Correlation Coefficient similarity measure for data term
lambda	0.1	weighting between data and smoothness term
minimum dimension	0	minimum size for any dimension within an image pyramid
label space refinement	0.33	label space rescale factor after each iteration of optimization cycle
distance term	TAD	(Truncated) Absolute Difference regularisation term
truncation value	0	truncation value for distance term (to allow flow discontinuities)
local labels	OFF	local label set computation
interpolation method	spline	cubic b-spline interpolation for computing dense deformation field
inverse projection method	spline	inverse projection using cubic b-splines
linked maximum	ON	maximum displacement is linked to the control point spacing
incremental regularisation	ON	regularise updates to the displacement field rather than entire field
field update mode	additional	adds the displacement field update after each iteration
sampling mode	sparse	sparse, rather than dense, displacement space sampling
image margins	0	image border margins (in mm) exluded from data cost computation

Table 3.1: Parameters used in Drop software (V1.05)

keypoint locations are then used to initiate the process again in the hope of additional improvement, generating a coarse warp with 12 keypoints instead of 5, before re-registering, updating, and inverting the warp.

Each of these steps, which are also used for view B, is illustrated using results from a frame in 15R in Figure 3.7. View B has additional initialisation procedure, however, as illustrated in steps 6, 7 and 8 of Figure 3.6, described in the next section.

3.3.4 Ray segments for updating predictions

The tuning process for view A provides 2-d keypoint locations, but in combination with fly silhouettes in other views, can give 3-d information. The visual ray from an updated keypoint in view A will have one or more depth intervals that is consistent with the silhouettes from the other views (given valid silhouettes and that the point lies on the wing in view A.) This principle was used in the visual hull algorithm of Boyer and Franco [49]. ‘Nudging’ the 3-d prediction onto the closest ray segment point therefore gives the closest point within the fly’s visual hull that projects to the tuned point. This method was used to update each of the five keypoint predictions, where possible, before tuning view B.

To obtain the depths along the visual ray for a tuned point, its epipolar lines were first determined in the other three views. In each view the epipolar line was used to create a boolean image of ‘line pixels’ that could be intersected with the boolean silhouette image (where pixels outside the wing bounding box are switched off). The line pixels were obtained as follows. If the epipolar line is given by the homogeneous coordinates $l = [a;b;c]$, then its gradient is given by $-a/b$. If the magnitude of this gradient is less than or equal to

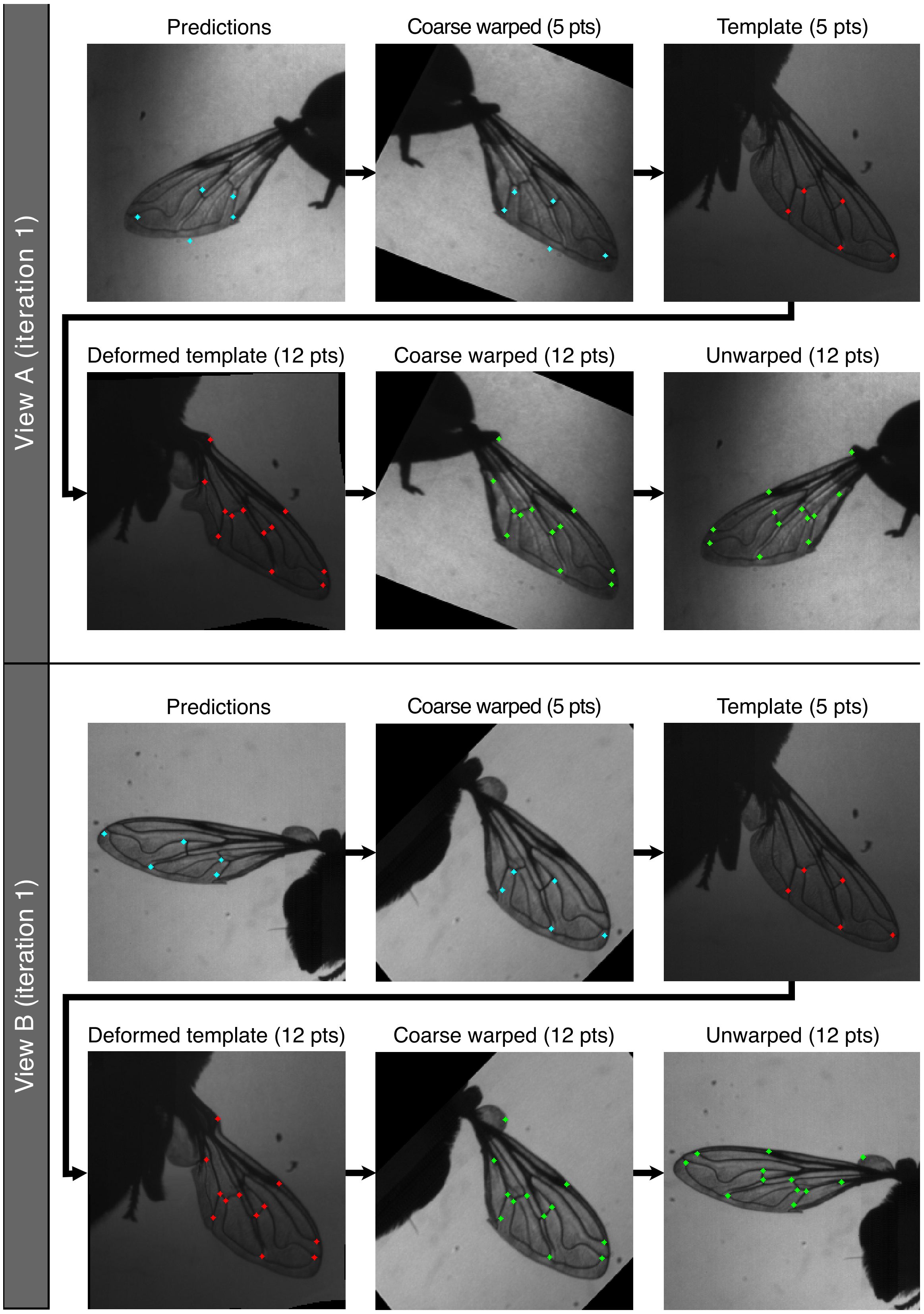


Figure 3.7: Illustration of point and image transformations in tuning process - 15R Frame 42. (View A from camera 2, View B from camera 1. Colours correspond to those in Figure 3.6.)

one (a ‘shallow’ line), a single pixel is switched on for each column (i.e. x value) in the image. The row of this pixel is given by the solving for y using the equation of the line, and rounding to the nearest integer. The process is similar for gradients with magnitude greater than one (‘steep’ lines): a single pixel is switched on for each row, and columns are determined by solving for x and rounding.

With the line pixels ordered by increasing x-value for shallow lines and y-value for steep lines, a boolean array was constructed, indicating whether these pixels belonged to the silhouette. ‘Entry’ and ‘exit’ points of the silhouette are given by the difference between adjacent elements in this array: 1 indicates an entry, -1 an exit. (By padding the boolean array at the start and end with an ’off’ value, it was ensured that ’on’ pixels at the start and end of the array would count as an entry and exit, respectively.) Subpixel coordinates were then obtained for these entries and exits on pixel boundaries. For entries, the boundary with minimum y-value for steep lines, and minimum x-value for shallow lines was used. An example entry/exit localisation is shown in Figure 3.8, which shows an epipolar line corresponding to tuned keypoint 1 in camera 2 of 15R frame 42 (view A of Figure 3.7).

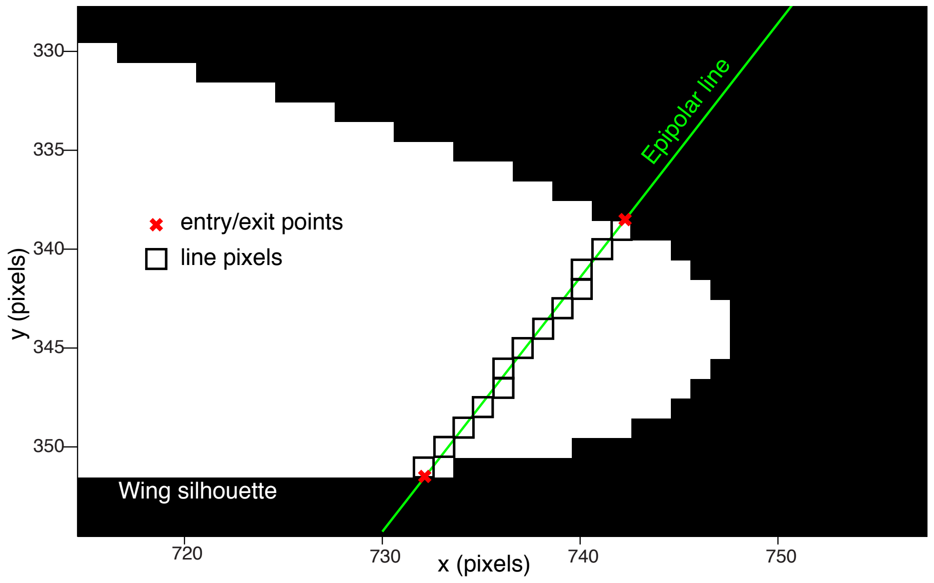


Figure 3.8: Epipolar entry/exit points through silhouette - 15R frame 42, camera 3.

In this way pairs of entries and exits were obtained for each camera, each corresponding to the limits of depth intervals on the visual ray. The depth at each of these points was obtained by finding the intersections with keypoint 1 and subtracting the view A camera centre. To determine the depth segments consistent with all views, an array of each of the depths was sorted, in increasing order; entries were replaced with ’1’ and exits with ’-1’; and a cumulative sum was calculated: a sum of 3 indicates a valid entry point, and the next point is the corresponding exit. Figure 3.9 shows depth intervals for keypoint 1 in the previous figure.

Finding the closest point (X_c, say) on the ray from the 2-d tuned point (x_t) to the 3-

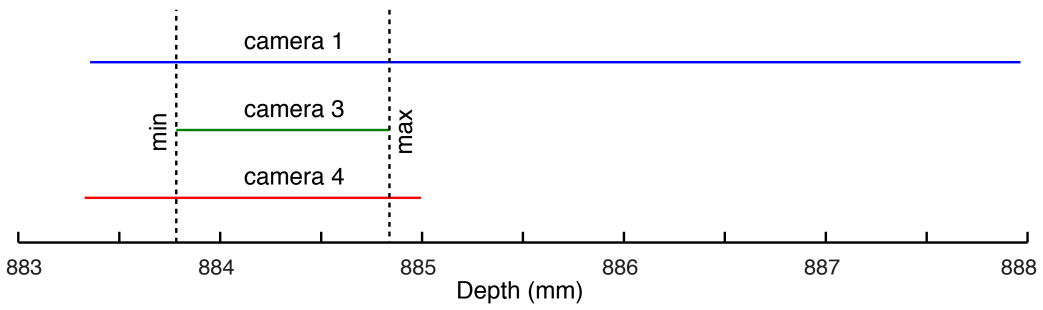


Figure 3.9: Depths intervals between silhouette entry/exit points, from tuned point 1 in 15R frame 42, camera 2.

d predicted point (X_p) requires a vector along the ray, which will be referred to as the 'sightline vector' ($\vec{v}$), and the camera centre (X_0). The sightline vector $\vec{v}$ can be obtained by:

$$\vec{v} = M^{-1} x_t^* \tag{3.2}$$

Here M is the first 3×3 submatrix of the the camera's 3×4 projection matrix P and x_t^* is the tuned point in undistorted homogeneous coordinates [29, p.162]. Using the property that the vector between X_c and X_p must be orthogonal to $\vec{v}$, the closest point is given by:

$$X_c = X_0 + \frac{\langle X_p - X_0, \vec{v} \rangle}{|\vec{v}|^2} \tag{3.3}$$

If this point is contained by one of the ray intervals, it is used as the 'nudged' point, otherwise the closest of the ray segment boundaries is used.

This process of updating the predictions before tuning view B is motivated by the smaller polygon area of its keypoints. As mentioned earlier, the larger polygon area for view A indicates a greater chance of improving 2-d keypoint predictions by registration. This 'nudging' method allows these improvements to be transferred to view B, thereby increasing the chance of the coarse-warp lining up the wing image with the template, and improving the likelihood of successful localisation. Figure 3.10 shows keypoint predictions that have been updated using this process, in view B of the frame used for recent figures (15R frame 42).

Figure 3.11 shows the results of the tuning process for this frame, including the 2-d tuned keypoints for both views after 2 registrations, and the projections of the 3-d results from intersection. The points in this frame have been well localised despite imperfect 5 keypoint predictions. The exceptions are the points near the wing shoulder, keypoints 11 and 12. For point 11, the presence of the leg in view A and the alula in view B appears to have had a distorting effect on the registration results. For point 12, the high degree of occlusion in this region in both views prevented good localisation.

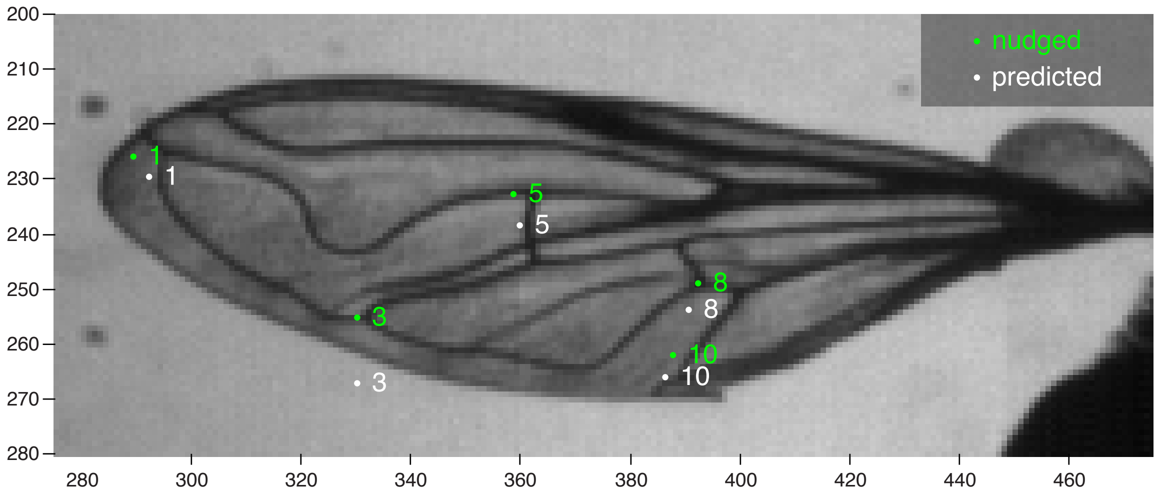


Figure 3.10: Updated predictions for view B, using ray segments based on view A tuning results. (15R frame 42, camera 1).

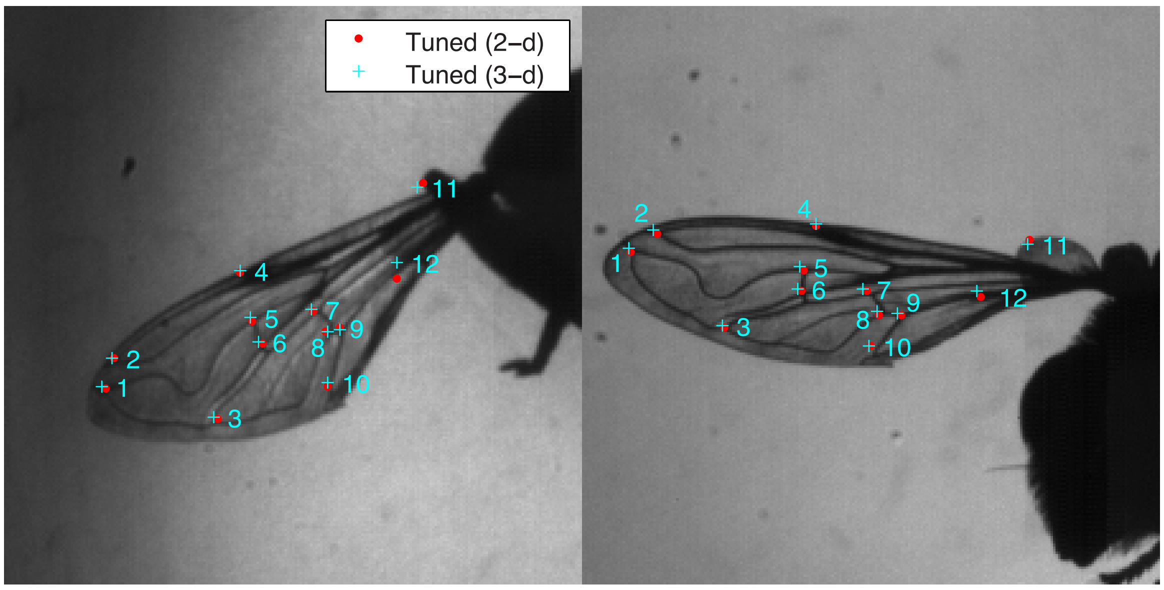


Figure 3.11: Tuning results for 15R frame 42: 2-d localisations and 3-d projections. (Left: view A, Right: view B.)

3.4 Smoothing

The tuning process provides 2-d locations of the 12 wing keypoints in two views, and, through intersection, their 3-d locations in each frame. The next challenge is to identify outliers and to use smoothness of motion to attempt to improve the results.

Figures 3.12 and 3.13 show tuning results for a frame in which one of the views (view B) achieved a significantly better registration. As a result, some keypoints appear to have been very well localised in 3-d (in particular 1, 2, 3, 5, 7 and 11) - in spite of bad predictions - and others very poorly localised (8, 9, 10, 12). A metric is required to separate out these points automatically. That points 3 and 10 lie completely outside the wing boundary in both views suggest the use of a silhouette-based constraint. Another useful property is that good correspondences should have visual rays that intersect or pass very close to each other. The use of both of these properties is discussed below.

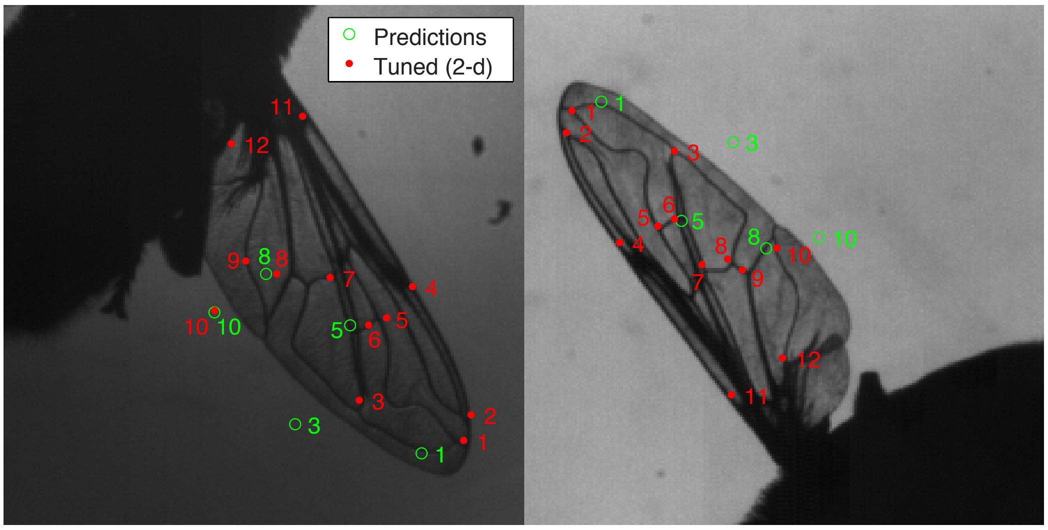


Figure 3.12: 2-d tuned keypoints and the projections of the initial 3-d predictions (15R frame 85).

3.4.1 Sightline distance

The minimum distance between two rays is given by the following equation (from [84]):

$$\text{distance}_{\min} = \frac{|\langle X_a - X_b, \vec{v_a} \times \vec{v_b} \rangle|}{|\vec{v_a} \times \vec{v_b}|} \tag{3.4}$$

Here X_a and X_b are camera centres that correspond to the sightline vectors $\vec{v_a}$ and $\vec{v_b}$.

Given any pair of 2-d tuned points, this equation, combined with the sightline vector equation (3.2), can be used as a metric for match consistency. This property will be referred to as sightline distance. Figure 3.14 shows the 2-d tuned points from the previous example coloured according to sightline distance. The colormap (inspired by traffic lights) uses green for highly consistent points, moving through amber and red as the sightline dis-

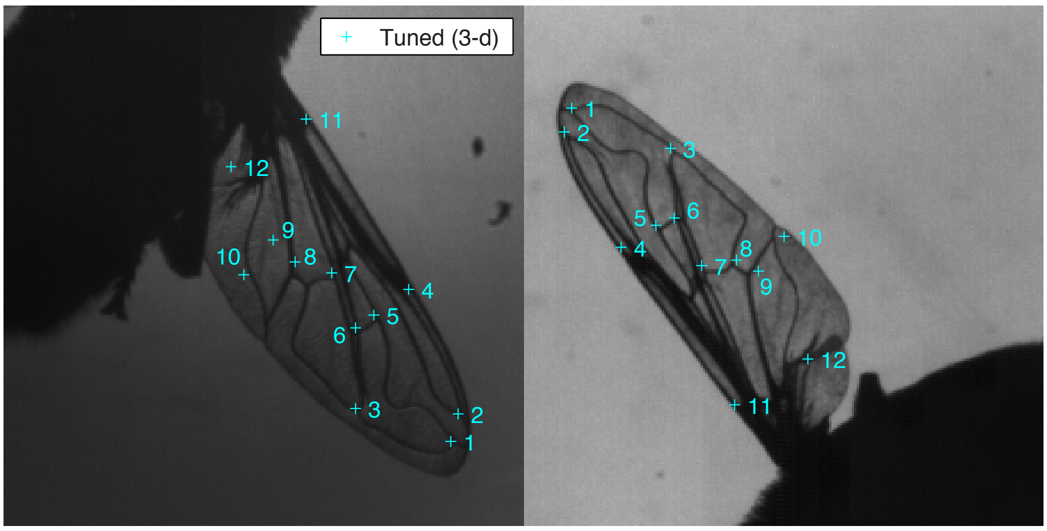


Figure 3.13: Projections of the 3-d tuning results, produced by intersection (15R frame 85).

tance (with units in millimetres) increases. In this example sightline distance successfully discriminates between good and bad matches: the keypoint pairs marked with low sightline distance colours have been correctly localised at the appropriate vein intersections. (One instructive exception is keypoint 6 - with one of the lowest sightline distances - has been 'tuned' onto the wrong intersection, albeit one very close to the correct intersection.)

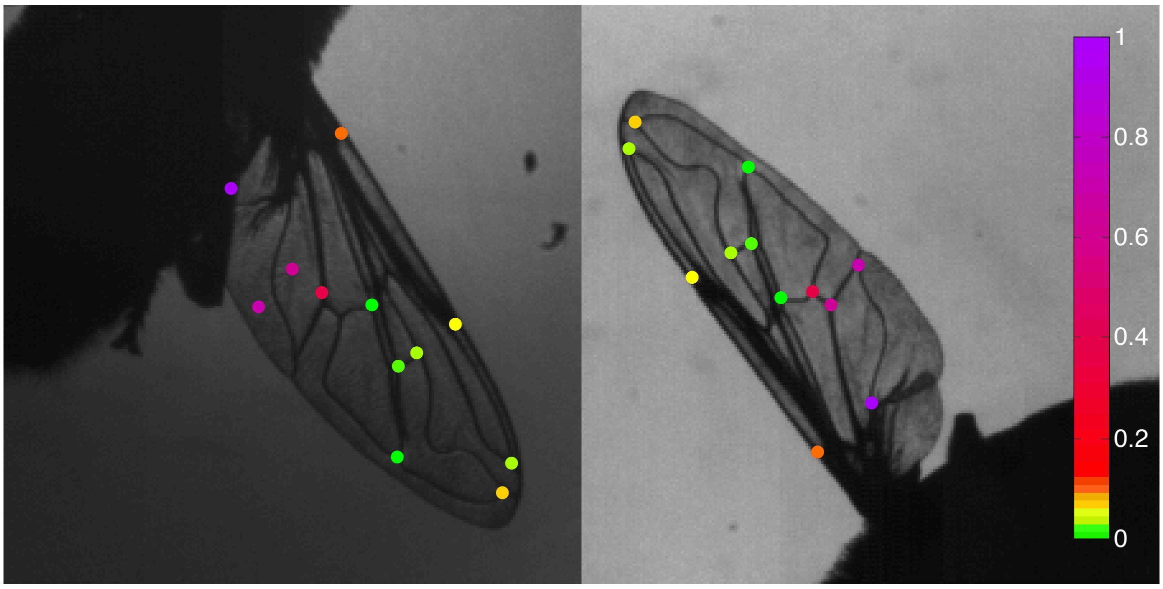


Figure 3.14: 2-d tuning results, coloured according to sightline distance of keypoint pairs (15R frame 85).

As a test for outliers, identifying keypoint pairs with sightline distance above some suitably high threshold can result in the elimination of false positives, but false negatives can slip through at any threshold. Any pair of points which have been coincidentally tuned onto a pair of corresponding epipolar lines will have a 0 mm sightline distance. Figure 3.15 shows such a pair. In this example the keypoint was badly localised on both views, resulting in a 3-d point that projects a considerable distance from the true location, and completely off the wing in one view. For this reason a new constraint was devised which is closely related to sightline distance but should catch many of these egregious false negatives: *epipolar segment distance*.

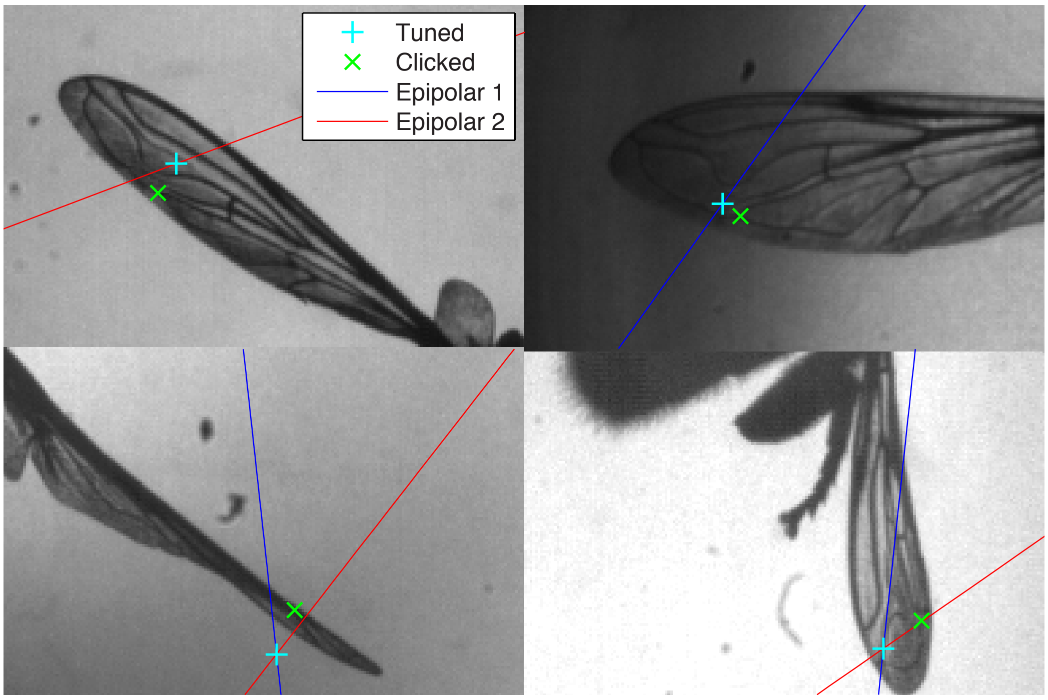


Figure 3.15: Bad correspondences with very low sightline distance. Views A and B are top right and left respectively. Projections of 3-d tuned keypoint 3 and its clicked location are shown for comparison, along with epipolar lines from 2-d tuned points. (15R frame 20)

3.4.2 Epipolar segment distance

This constraint uses the ray segments found in the prediction update process of Section 3.3.4. These ray segments correspond to the 2-d tuned points of view A. Their projections in the other views will be referred to as 'epipolar segments' as they demarcate the regions of the epipolar lines that corresponding points must lie near, in order to be consistent with the silhouettes from the other views.

Epipolars from view A are used based on the assumption that greater confidence can be placed in the tuning results in this view. Figure 3.16 shows epipolar segments from a view in which keypoints have been well-localised by tuning. As a consequence of the very narrow wing silhouettes in two of the views, the epipolar segments are very short. This confines acceptable tuning results from view B to small areas.

The 'epipolar segment distance' is the minimum separation in pixels between a tuned keypoint in view B and the corresponding epipolar segments from view A. This constraint was used for outlier detection in a given frame by rejecting any keypoint for which this distance is greater than a chosen threshold, d. This is illustrated in Figure 3.17.

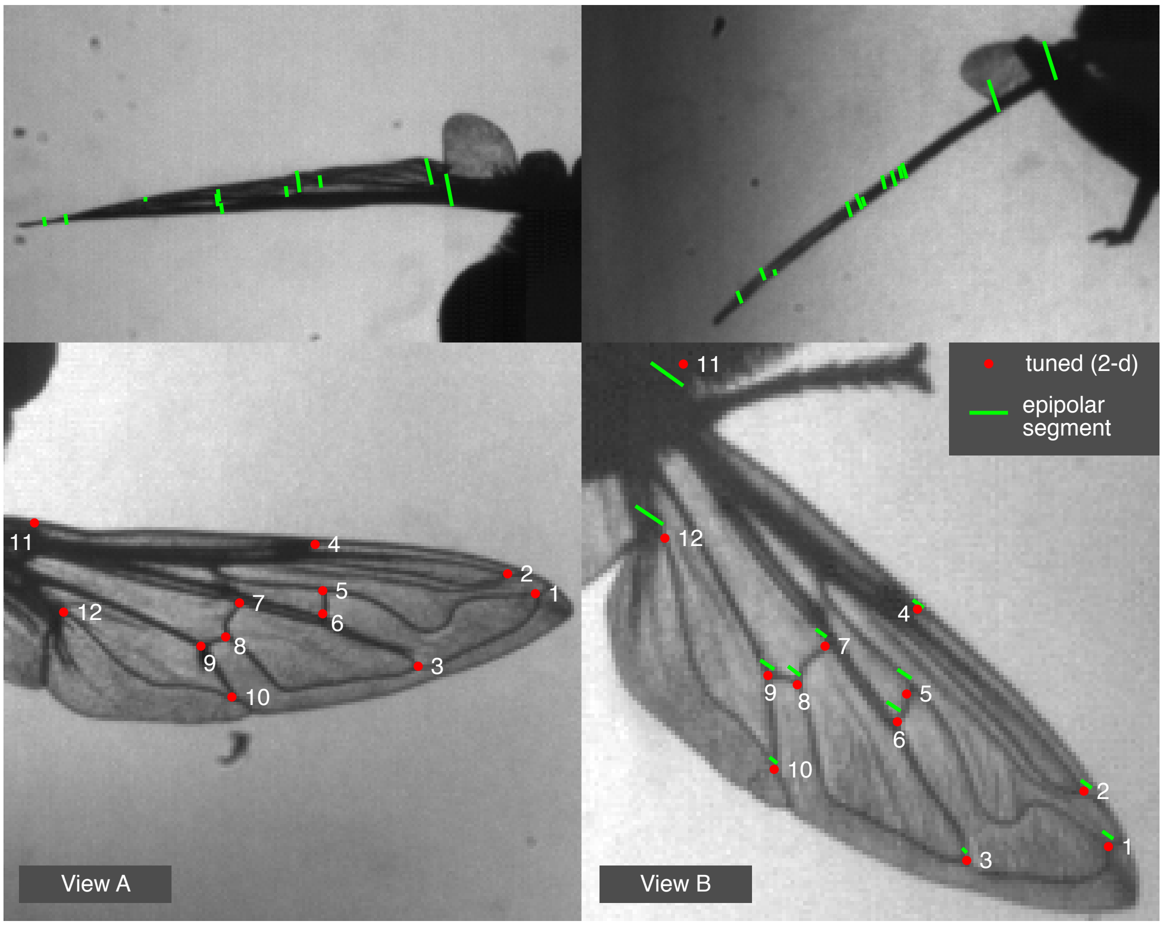


Figure 3.16: Epipolar segments from tuned keypoints in view A. Well-matched points in view B should be found near corresponding segments. (15R frame 39)

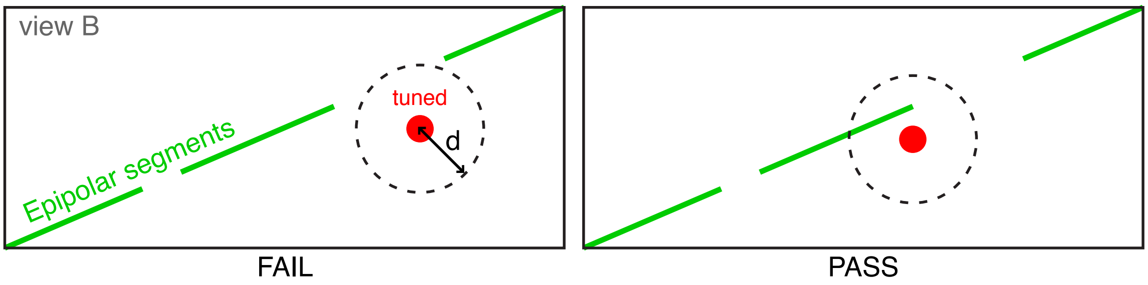


Figure 3.17: A point is identified as an outlier if it is not within *d* pixels of any point on the epipolar segments.

This test was performed as follows. If a tuned point p, with coordinates (x_t, y_t), is within d pixels of the epipolar line, then a circle of radius d around p will intersect the epipolar line at either one or two points. If one or both intersection points are within an epipolar segment, or if a segment lies between the intersection points, then the tuned point is within d pixels of a segment, and so passes the test. Otherwise it is rejected.

If the epipolar line has homogeneous coordinates (a, b, c), then the intersection points will be at (x, y) satisfying:

$$\begin{aligned} ax + by + c &= 0 \\ (x - x_t)^2 + (y - y_t)^2 &= d^2 \end{aligned} \tag{3.5}$$

The x-values can therefore be found as the solution to a quadratic equation:

$$\begin{aligned} \alpha x^2 + \beta x + \gamma = 0 \\ \text{where} \quad \alpha &= 1 + \frac{a^2}{b^2} \\ \beta &= \frac{2a}{b}\left(y_t + \frac{c}{b}\right) - 2x_t \\ \gamma &= x_t^2 - d^2 + \left(y_t - \frac{c}{b}\right)^2. \end{aligned} \tag{3.6}$$

By comparing these x-values to the x-values at the endpoints of each epipolar segment, one can then determine whether tuned points pass or fail.

3.4.3 Three-stroke smoothing

The smoothing process uses an epipolar segment distance of 5 pixels as an outlier test and uses sightline distance to weight each point for smoothing: the higher the sightline distance, the greater the smoothing around this point.

At this stage keypoints are treated independently. Their spherical coordinates are also treated independently, with theta, phi and R smoothed separately. Smoothing over three strokes, rather than one, ensures a good smoothing result in the vicinity of the extrema for the 'middle' stroke and smooth transitions between neighbouring strokes. A cubic smoothing spline (*csaps*) was used, with weights for the error measure given by the reciprocal of the sightline distance. Doubling the sightline distance therefore halves the weight. Outliers were excluded from the fit, so tuned points in these frames have no influence on the smoothed result.

Figures 3.18 and 3.19 show smoothing results for theta and phi (6 keypoints in each figure) for the second stroke in set 15R (an upstroke). Figure 3.20 shows the radius values for each point, smoothed in the same way. Each point is coloured according to sightline distance as before.

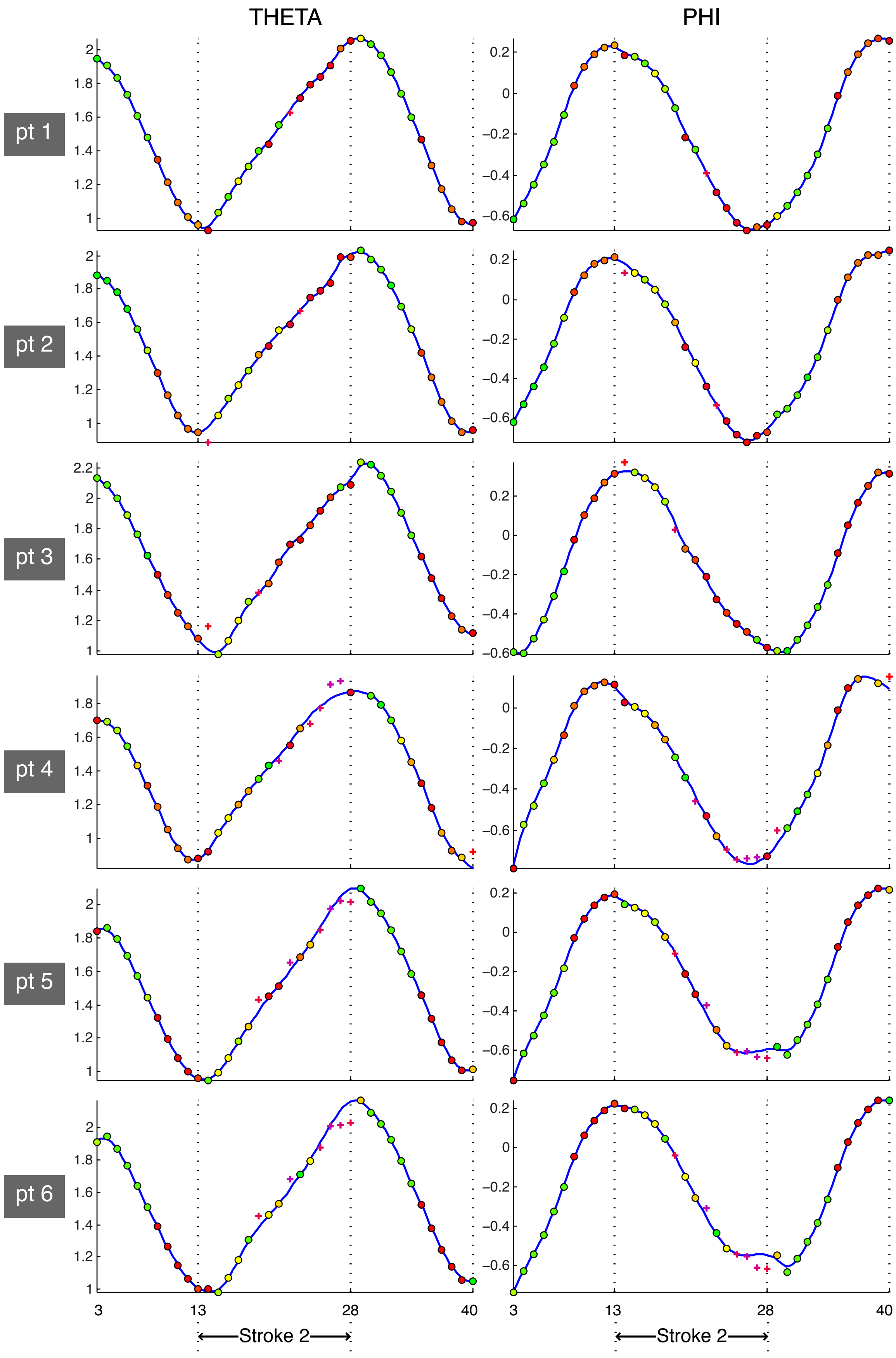


Figure 3.18: Smoothed theta and phi values for 6 keypoints. Colours reflect sightline distance, and crosses indicate outliers. Smoothing spline is shown in blue. (15R points 1-6)

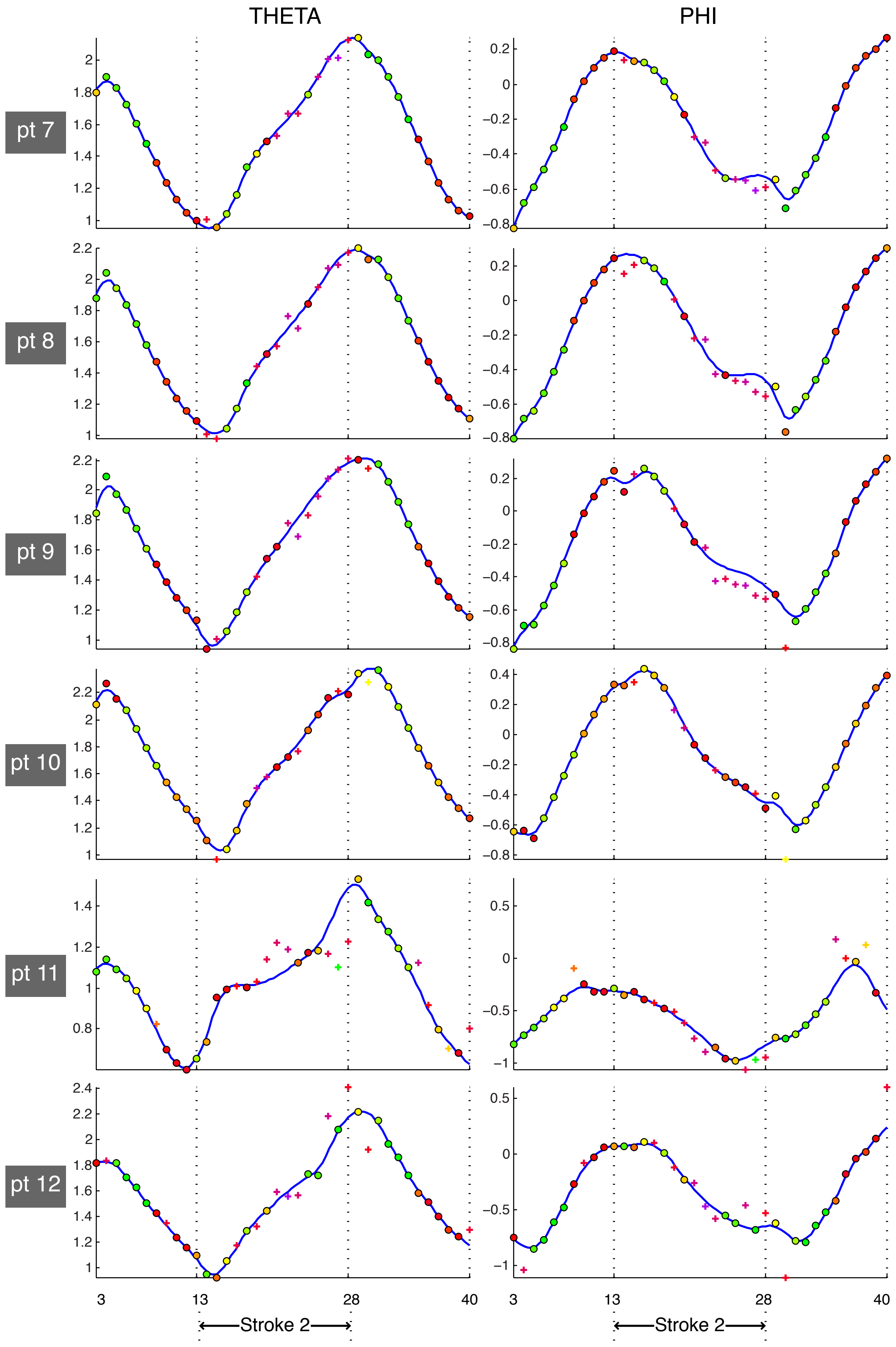


Figure 3.19: Smoothed theta and phi values for 6 keypoints. Colours reflect sightline distance, and crosses indicate outliers. Smoothing spline is shown in blue. (15R points 7-12)

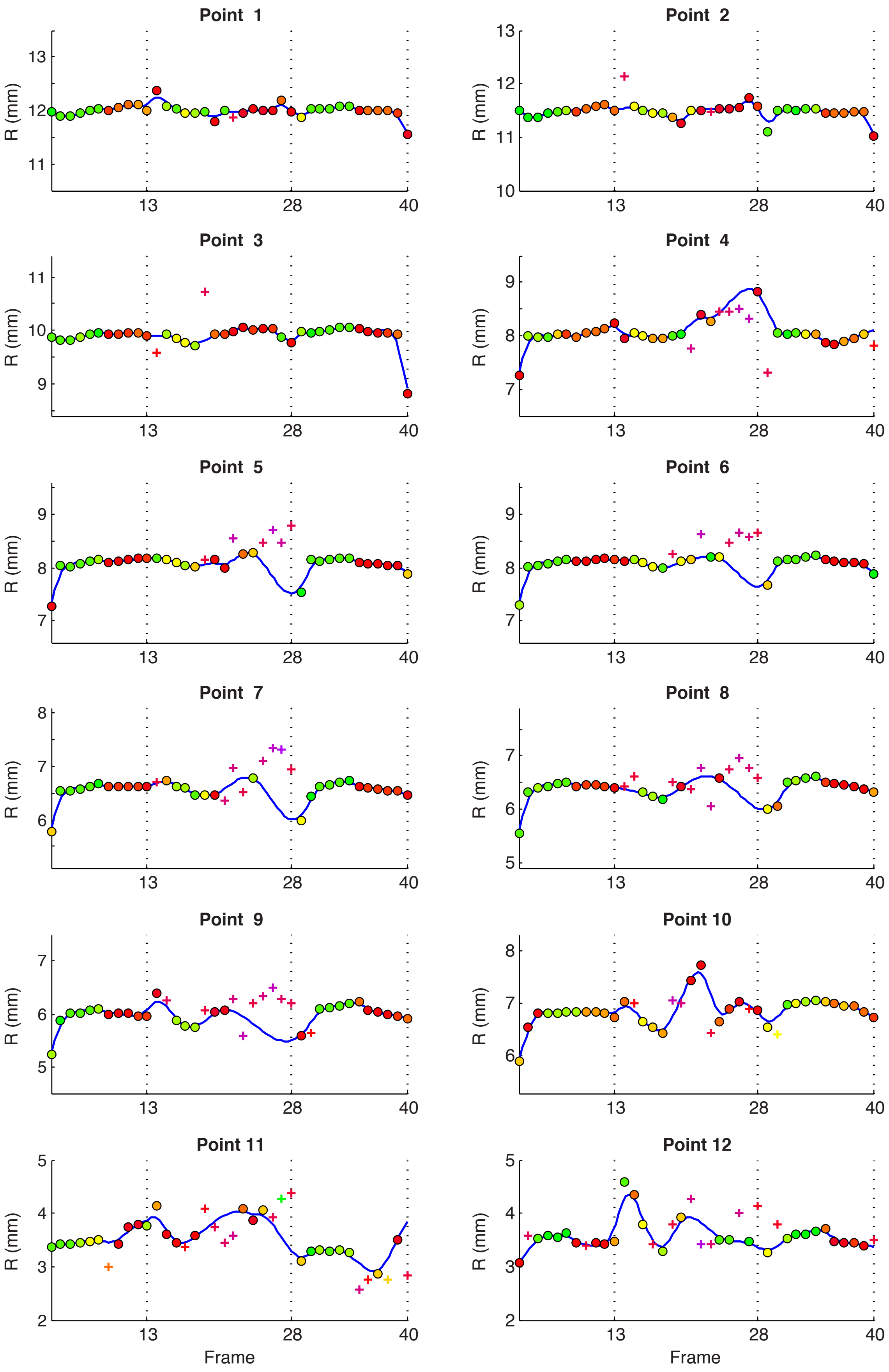


Figure 3.20: Smoothed R values for 12 keypoints. Colours reflect sightline distance, and crosses indicate outliers. Smoothing spline is shown in blue.

This smoothing process produces a new set of points at each frame, with coordinates given by the spline. In the example of smoothing stroke 2 (the first stroke to be smoothed), the new points are used to predict stroke 4. Tuning stroke 4 then allows stroke 3 to be smoothed, and so on until the final stroke (stroke N) has been tuned, allowing the smoothing of stroke $N-1$.

3.5 Wing fitting using inextensibility

The prediction, tuning and smoothing loop described in previous sections does not enforce 3-d spatial relationships between wing keypoints. This enforcement was introduced after obtaining smoothed 3-d traces of each keypoint for the full image sequence. For each frame this begins with gathering the 3-d locations for the 12 keypoints and using them to create a wing surface, using the same procedure as used in Chapter 1 to build a reference wing surface. To recap, this uses 2-d to 3-d thin-plate splines to map from the 2-d keypoint locations in the wing-specific reference image to the new set of 12 3-d points. This map can be used to send any point in the reference image to 3-d space, and so forms an implicit representation of the wing. In this instance thin-plate smoothing splines were used, with a smoothing parameter ($p = 0.9$ as input to the *tpaps* function) chosen to give a good balance between smoothness and proximity to keypoints. In the following steps, this map construction will be referred to as 'wing fitting'.

As before, the distance between keypoints on this surface was calculated by stepping along a path of 100 points between them, and adding up the linear distances. The resulting array of keypoint separations was subtracted from the reference separation array, and the L^1 norm was used to quantify this disparity, i.e. finding the sum of the absolute values of each element. This provides a measure of the extent to which the wing fit violates the inextensibility constraint. This error measure was used in a process which creates many different wing fits by sampling subsets of the 12 keypoints, attempting to identify points which together produce a good wing fit, and weeding out outliers. For this a modified version of the RANSAC ('random sample consensus') algorithm was created with the following recursive steps:

1. Fit wing to subset of 5 keypoints.
2. Find all keypoints within 0.1mm of their locations in the wing fit. This is the current 'consensus set', C.
3. If C is largest consensus set yet encountered (and hasn't been examined previously), fit wing to C.
4. Get error measure for this fit.
5. If this is the lowest error encountered for a consensus set of this size, save this error, fit and consensus set as the 'best'.
6. Repeat for all (792) subsets of 5 points.

In a sentence, this algorithm looks at all 5 point subsets, and finds the one with the smallest inextensibility violation among those that yield the biggest consensus set'. This 'best' consensus set is the set of inliers, and they provide the wing fit for this frame. The other points are outliers, which are discarded and replaced by the corresponding 3-d location from the wing fit, as illustrated in Figure 3.21.

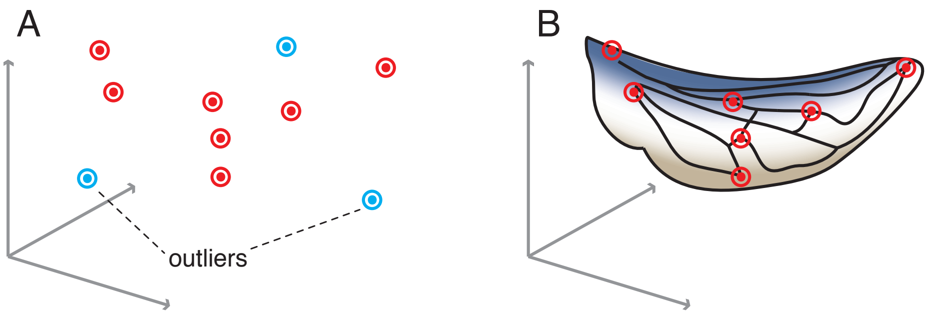


Figure 3.21: Outlier rejection for wing fitting. A: Keypoints are divided into inliers and outliers based on inextensibility constraint. B: Inliers can be used for determining 3-d shape.

Figure 3.22 shows an example of a frame in which this fitting process produces a good wing shape in spite of two wildly inaccurate keypoint locations. These points have been successfully rejected as outliers.

This process produces an improvement in wing projections for most but not all frames. Figure 3.23 shows an example of a frame in which outlier rejection was detrimental. In this example, the six keypoints in the region with greatest bending (near the trailing edge towards the shoulder) have all been rejected, producing an 'over-flattening' of the wing. This highlights the trade-off between minimising inextensibility violation and maximising the number of points retained for the wing fit. The design of the fitting process - in particular, the choice to prioritise consensus set size over error measure - aims to strike a good balance between these competing factors.

3.6 Final smoothing

With wing fits established independently for each frame, the remaining step is to bring back in temporal continuity by smoothing over the keypoint 3-d positions a final time. Whereas the previous smoothing process weighted according to sightline distance, this one uses the inextensibility error measure from the wing fitting process. For inlier keypoints (i.e. points used in the wing fit), the weight used is the reciprocal of the error measure for the frame. For outlier keypoints, the weight is equal to half of the lowest inlier weight. This uses the assumptions that the most reliable points are those which are inliers in low error fits; that reliability drops with fit error; and that outliers are least reliable but should be treated as equivalently unreliable as the wing fit is not guided by a measurement at these points. This goes some way to counteracting the problem (demonstrated in Figure 3.23) that wing regions lacking inliers may be poorly represented by the fit, using nearby frames

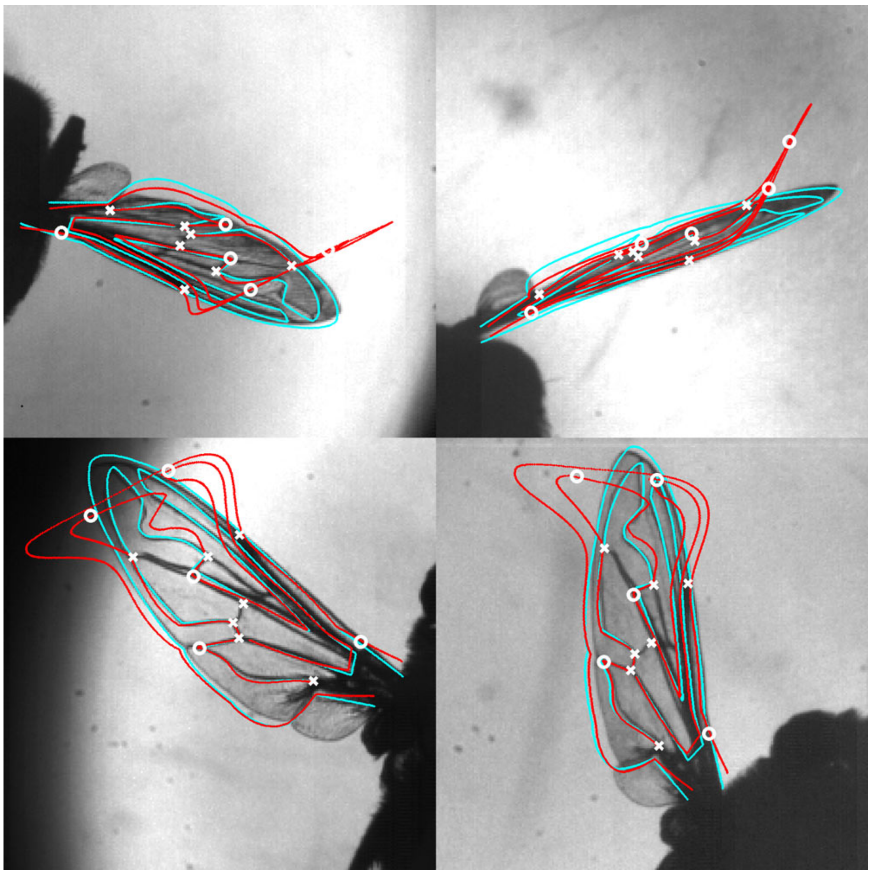


Figure 3.22: Improved wing fit (cyan) using inlier keypoints (x) and rejecting outliers (o). Error measure reduced from 48 to 7 mm. The three outliers closest to the shoulder appear very close to the improved fit when projected, but their 3-d separations exceed the 0.1 mm threshold. (23L frame 23)

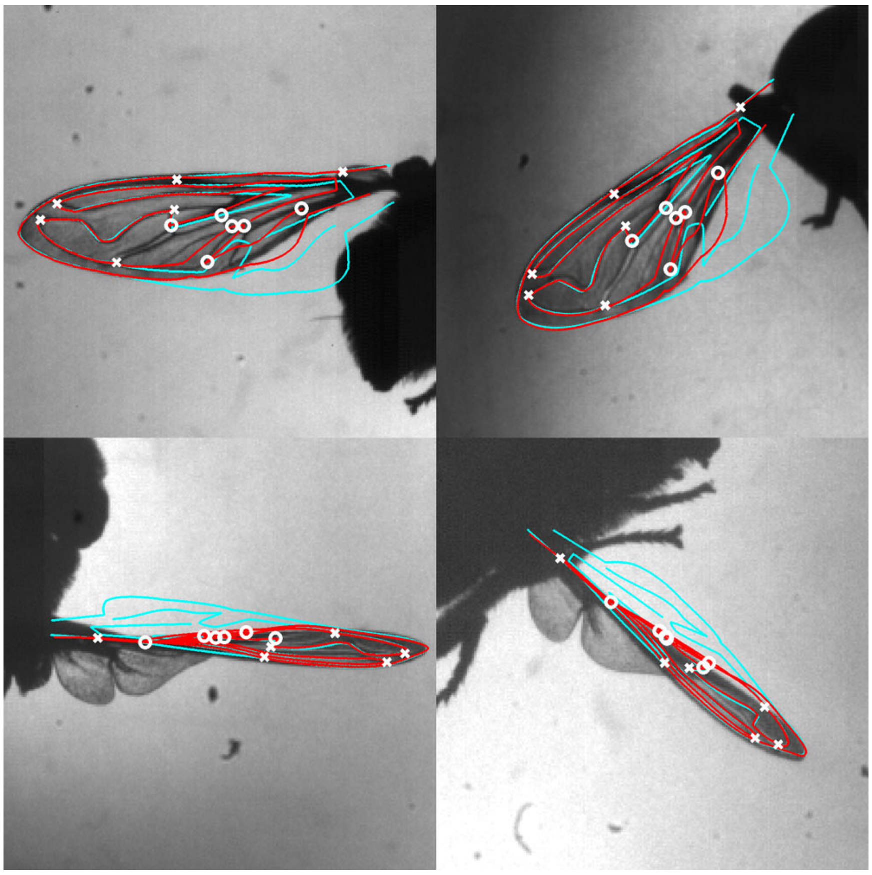


Figure 3.23: New wing fit (cyan) has improved point separations (15 compared to 40 mm) but projections are inferior. (15R frame 94)

for improved localisation.

As in the previous smoothing process, a cubic smoothing spline was used for each of the spherical coordinates for each of the keypoints. The smoothed values produced in this step are the final output of the keypoint tracking algorithm, covered in this chapter. Figure 3.24 shows the spherical coordinate results for keypoint 1 of set 15R. Figure 3.25 illustrates each of the steps of the algorithm.

3.7 Discussion

This chapter has detailed the steps required to obtain estimates of a set of interior points over an arbitrary number of wingbeats. These points can be used directly to produce a parameterised wing surface, or, as will be demonstrated in the next chapter, they can be combined with points on the wing boundary to obtain an improved wing shape. The algorithm requires manual supervision in the form of clicking a small subset (5) of these points in a single frame in each stroke.

Why only 12 points?

It is important to note that while only 12 points were tracked, the algorithm is not restricted to this number. The tuning process, which takes five 3-d predictions as input, is responsible for pulling out the desired number of points. As many points can be pulled out as there are pixels on the wing in the template image. This follows from the ability of the registration step to obtain a dense deformation field. The deformation field gives a map from every pixel in the template to the coarse-warped image, so composing this with the inverse of the coarse warp provides a map from every pixel in the template to the corresponding location in the current image.

In tests of 'dense' tuning, the locations of one in every twelve pixels in the template image were used, as illustrated in Figure 3.26. By sampling the tuning deformation fields at these locations, 3-d points were obtained for each of the sampled pixels, and the sightline distances associated with the correspondences. Figure 3.27 shows the 2-d tuning results for these pixels, coloured according to sightline distance. This is a companion to Figure 3.14, which showed 12 keypoint locations for the same frame. A close-up of view A is shown in Figure 3.28.

The smooth gradient of sightline distance 'colours' shown in these results is indicative of the smoothness of the surface containing the points, which is a consequence of the smoothness of the deformation fields. It also demonstrates that match consistency varies smoothly over the points on the wing, so that poorly matched points will be surrounded by other poorly matched points, and well-matched points will be surrounded by other well-matched points. (The same goes for points with misleadingly low sightline distances.) Tuning can therefore be thought to fail not just for individual points, but for entire regions

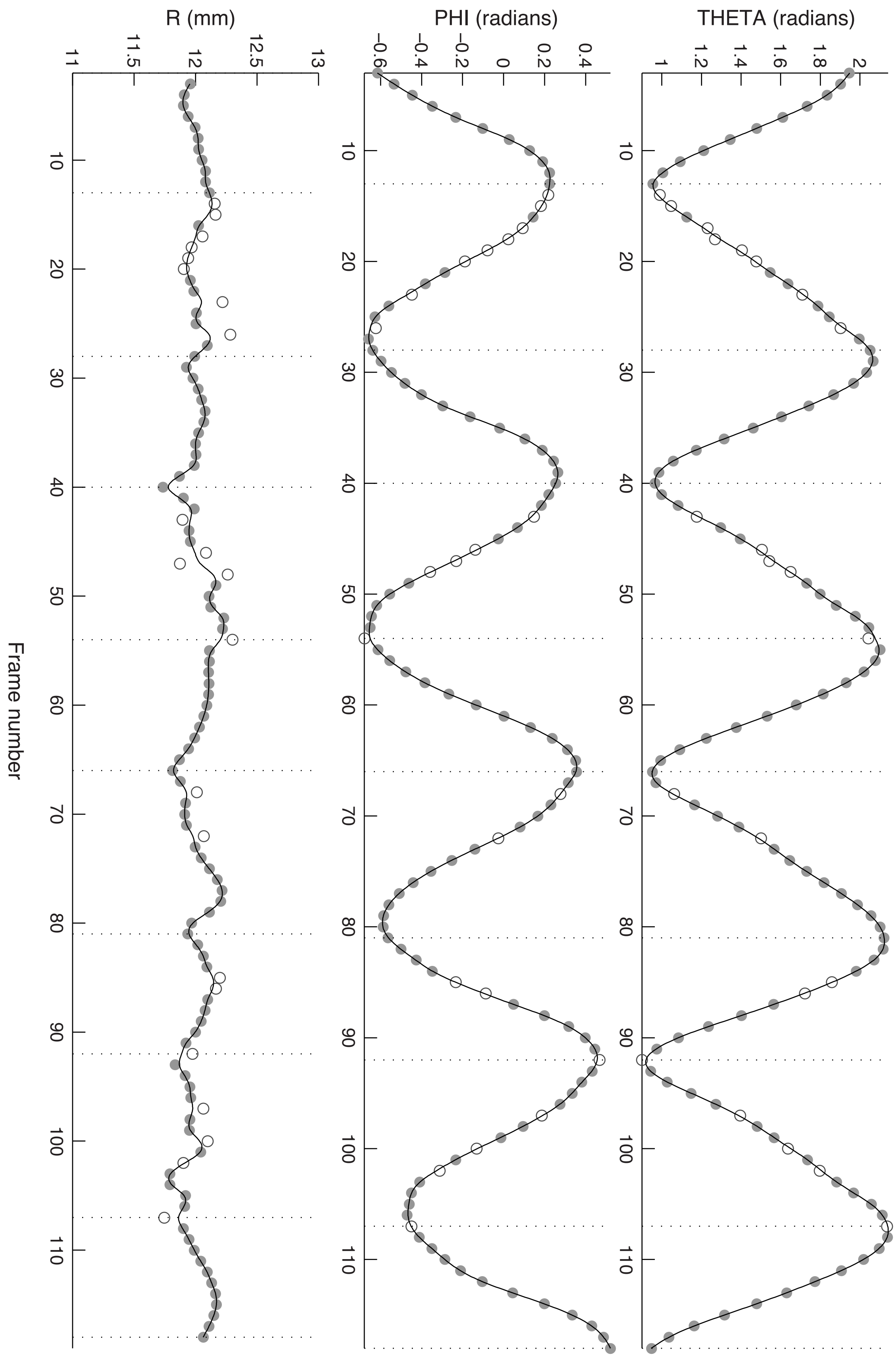


Figure 3.24: Final smoothing results for keypoint 1 over all frames in set 15R. Gray circles indicate wing fit inliers, white circles are outliers.

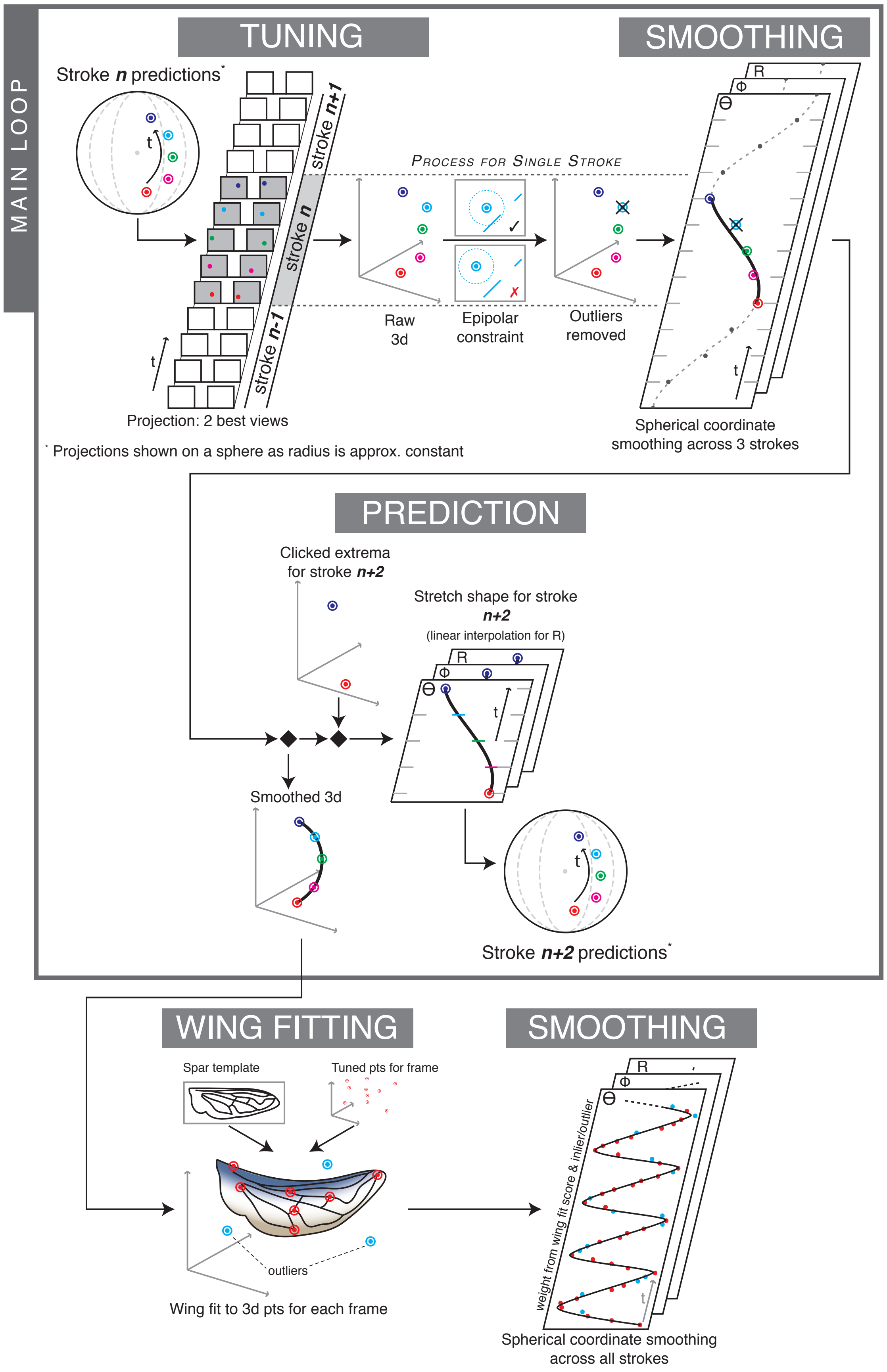

MAIN LOOP
TUNING
SMOOTHING
Stroke n predictions*
stroke n+1
stroke n
stroke n-1
t
Projection: 2 best views
* Projections shown on a sphere as radius is approx. constant
PROCESS FOR SINGLE STROKE
Raw 3d
Epipolar constraint
Outliers removed
R
Φ
Θ
Spherical coordinate smoothing across 3 strokes
PREDICTION
Clicked extrema for stroke n+2
Stretch shape for stroke n+2
(linear interpolation for R)
Smoothed 3d
Stroke n+2 predictions*
WING FITTING
SMOOTHING
Spar template
Tuned pts for frame
outliers
Wing fit to 3d pts for each frame
weight from wing fit score & inlier/outlier
Spherical coordinate smoothing across all strokes

Figure 3.25: Overview of steps in keypoint tracking algorithm. The main loop is simplified by showing the processing of a single keypoint, rather than all 12.

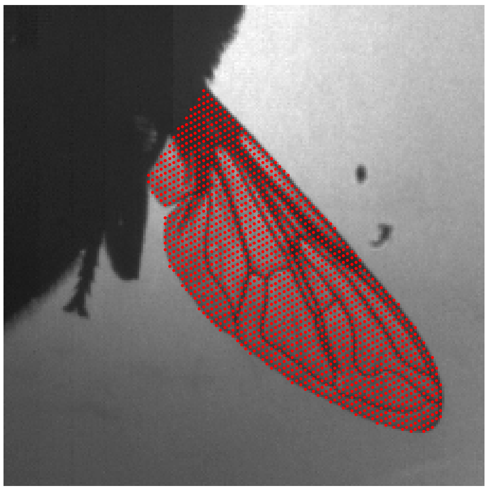


Figure 3.26: Densely sampled points on 15R template.

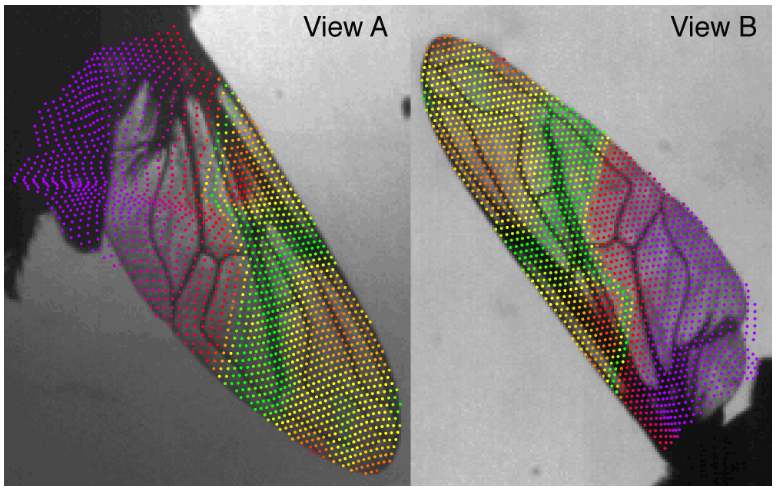


Figure 3.27: Dense 2-d tuned points coloured by sightline distance. (15R frame 85)

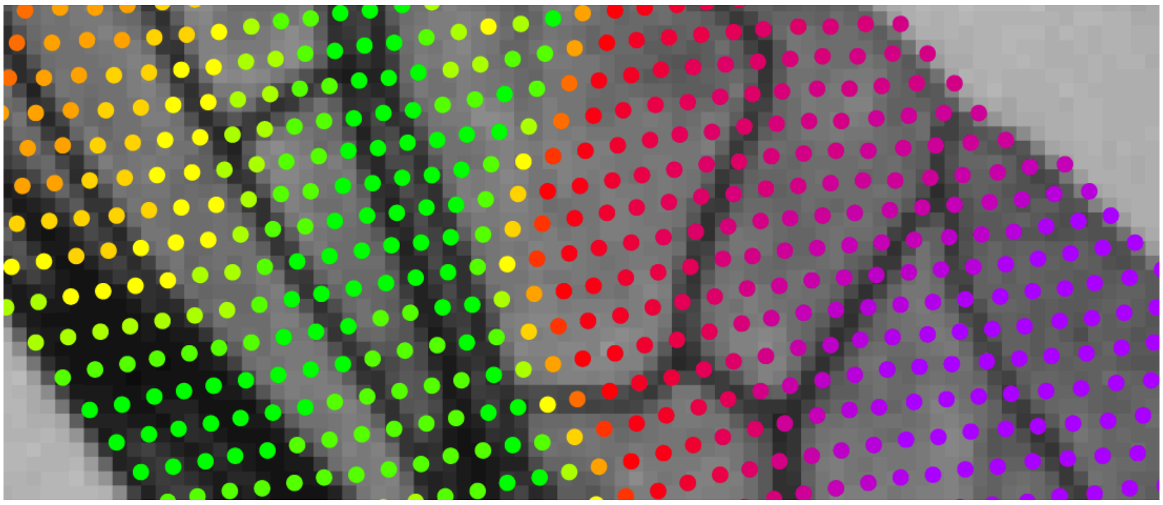


Figure 3.28: Wing detail showing sightline distances from tuning results of densely sampled template points, rather than keypoints. (15R frame 85 view B)

of the wing. For this reason it makes sense to use a sparse sample of keypoints which is small enough for outlier detection to be effective, but large enough and suitably spread to provide a good representation of wing shape.

Another reason to track only a small number of points is the computational cost of the wing fitting process. This step requires that all 5 member subsets of the points be examined. A wing surface is created for these 5 points, then another is created for all those points sufficiently close to the surface (if this surface hasn't already been created). For 12 points the number of subsets is 792 (12 choose 5), so up to 1584 surfaces are examined per wing per frame. Increasing to 20 points, say, would require 31008 surfaces per wing per frame, and each surface would require almost three times as many paths to be interpolated and measured for the error measure (190 compared to 66). If computation time scales proportionately to these increases, the fitting for the 116 frames of set15L, which took 27 minutes using a 2.53GHz Intel Core 2 Duo processor (other sets similar), would take over two days. This is a substantial burden without any guarantee of an increase in measurement accuracy.

A final reason to stick with extracting and tracking the 12 keypoints is that they are readily identifiable by eye, meaning that a 'ground truth' of sorts can be established for their 3-d locations by clicking in each image and intersecting the results. As the prediction, tuning, smoothing and wing fitting steps of the algorithm each provide 3-d locations for the 12 keypoints, each stage can be assessed based on how close the 3-d output is to the clicked locations. Detailed analysis of this accuracy quantification will be provided in Chapter 6.

Prediction

The goal of the prediction process is to provide five 3-d keypoint estimates which determine the coarse warp in the tuning process. A set of predictions for a given frame can be said to be 'good enough' if this coarse warp leads to a successful image registration. (There are many case-specific factors which influence this level of success, so the required accuracy is extremely difficult to quantify.) The predicted points' locations are also very important in that they determine camera choice for the tuning process - an inappropriate selection of views can mean the difference between success and failure in the tuning process.

The reference stroke patterns (from 15R strokes 1 and 2), used in the prediction of the first three strokes, proved suitable for each of the data sets examined. They might not, however, be appropriate for all data sets. For a set in which the tuning of these initial strokes is unsuccessful, it will be necessary to click points the five points in the first wingbeat.

Tuning

In choosing how to select appropriate views for tuning, an alternative approach of estimating wing areas from silhouettes rather than predictions was considered. This involves obtaining the 'minimum' image of the sequence (see Appendix B) and manually identifying a line which bisects the fly in each view, separating the left wing from right. The pixel area of the appropriate half of the silhouette in a given frame relative to the minimum area over all frames gives a crude approximation of the current wing area in this frame (assuming the wing is just a narrow strip in the frame with lowest area). Over short frame sequences this might be an acceptable alternative, but the drift of the body that will be present in most hovering sequences will distort these estimates over long sequences. If the wings can be reliably segmented in the silhouettes, however, this more direct approach would be preferable to using predicted point polygons.

Another design issue for the tuning process was the choice of how many times to perform the registration for each view. Two was the number settled on as the tuned points from the second iteration seemed to produce either minor refinements only or was unable to correct poor localisation from the first iteration.

Smoothing

In developing the smoothing process, two alternative constraints for outlier detection were considered. The first is a radius-based constraint. This involved removing any point whose radius coordinate was outside an allowed range of values: 3 standard deviations around the radii from beat 1 (as measured for each individual keypoint). Figure 3.29A shows a point for which this worked well. For some points, however, this is overly restrictive, as the radius values behave more erratically. Figure 3.29B shows an example of such a case. In general, assuming that radius values stay approximately constant over multiple strokes is risky: it requires that rigid body motion be measured with enough accuracy that the wing coordinate frame remains well-centred on the shoulder; and it doesn't allow for the possibility of changes in flapping motion (for example, during a manoeuvre) affecting radius values. In order to avoid the risk of eliminating any well-tuned points in this way, this test was not pursued.

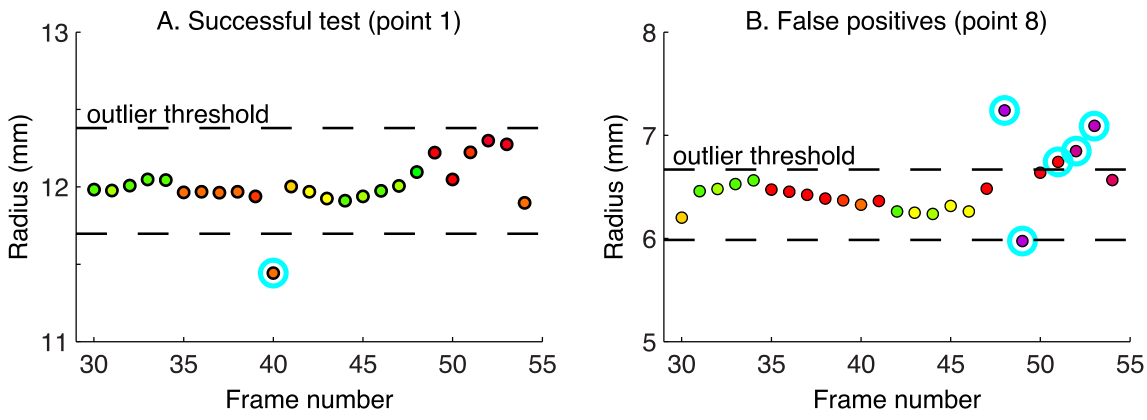


Figure 3.29: Successful and unsuccessful applications of radius thresholding test. Outliers circled in cyan; points coloured by sightline distance. (Set 15R)

A more straightforward silhouette-based outlier test was also examined. This projected

3-d tuning results into each view, and eliminated any point which lies outside any of the silhouettes. This is effective for catching many of the tuning outliers, but suffers two drawbacks compared to the epipolar segment distance test. The first is that it is occasionally overly restrictive: some points project slightly outside boundaries due to only slightly imperfect tuning, and are better treated as inliers (and weighted by their sightline distance). The second is that in some cases it is not restrictive enough, letting through some points that would be rejected by the epipolar segment distance test. For these reasons the segment-based test was chosen instead. Figure 3.30 shows the points which are removed as outliers by both tests. In the case of the epipolar segment test, a range of distance values were included. These results helped inform the choice of the distance limit: at 5 pixels there are still a considerable number of outliers being removed, without creating very large gaps in the data that could prevent a good spline fit.

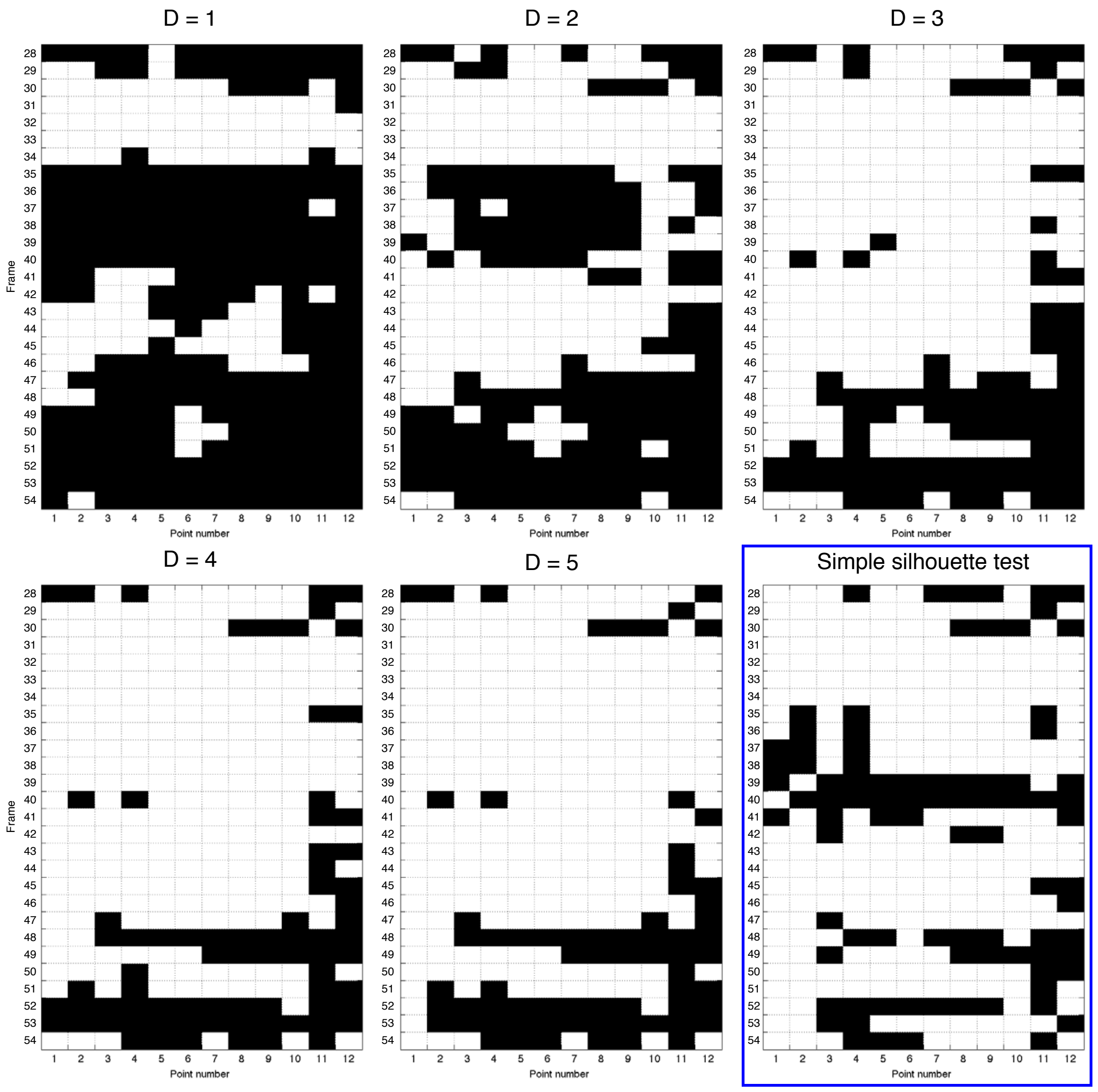


Figure 3.30: Inliers/outliers (white/black) for 12 points over wingbeat for various distance values. Bottom right shows simple silhouette test results. (Set 15R beat 2)

Chapter 4

Incorporating the Boundary

4.1 Overview

This chapter describes the final steps of the algorithm, which focus on two objectives. The first objective is to obtain points on the rim of the wing, which are referred to as the *wing boundary*. The previous chapter provided a method for estimating the positions of keypoints spread over the wing surface. In the process, fly silhouettes and the epipolar constraint were used to verify keypoint matches, but they can also be used more directly to establish boundary points.

The second objective is to create a wing surface using both internal point estimates and boundary point estimates. For this, a thin-plate spline surface representation is created for each frame, which encodes the 3-d location of every point in a reference image. This is the final output of the algorithm. These wing surfaces are shown for a full data set at the end of this chapter.

4.2 Boundary splines

The first step towards extracting a 3-d boundary is to extract the 2-d silhouette perimeters from each of the views. Among these perimeter pixels will be a number of points corresponding to the desired wing boundary, but also many points from the rest of the fly body, points on dust particles (due to imperfect segmentation), and points on the interior of the wing wherever folding causes self-occlusion. By correctly matching and intersecting points within these perimeters, sections of the 3-d boundary can be found.

To represent these perimeters as a continuous and ordered set of points, smoothing splines were fitted to them using the following steps. The pixels on the outer edge of the fly (and wings) were obtained using a simple morphological operation on the silhouette images - Matlab provides a suitable operation with the function *bwboundaries*. Next the wing bounding boxes, previously used for cropping images for tuning, were used to crop out all the edge pixels except those around the wing of interest. In order to get subpixel localisation of edge points, the Canny edge detector response was observed around these pixels, and the local maximum along the local edge orientation was found by bilinear interpolation. The functions *canny* and *nonmaxsup* (with radius 1.2 pixels) by Peter Kovesi [85] were used for this task, and a further Kovesi function *edgelink* was used to link together adjacent edge points into chains. These chains provide the discrete steps around the wing (and legs, and anything else that might be inside the bounding box) and are suitable for spline fitting.

A cubic smoothing spline (Matlab's *csaps* function with 0.5 as the smoothing parameter) was used to connect the subpixel locations from the longest chain (which will necessarily contain the wing). The domain parameter (t) was specified such that its value at each of the edge points is equal to the sum of the distances between preceding control points. This allows one to step along the spline at approximately 1-pixel intervals by using integer values for t. Splines were fitted in this way for every wing in the image sequence. An example frame is shown in Figure 4.1, showing the splines in each view. This frame also illustrates the challenge posed by self-occlusion: even by eye it is very difficult to separate pixels on fold edges from those on leading and trailing edges (i.e. the wing boundary). A detail from camera 1 in this frame is shown in Figure 4.2, including the spline and the original silhouette perimeter pixels.

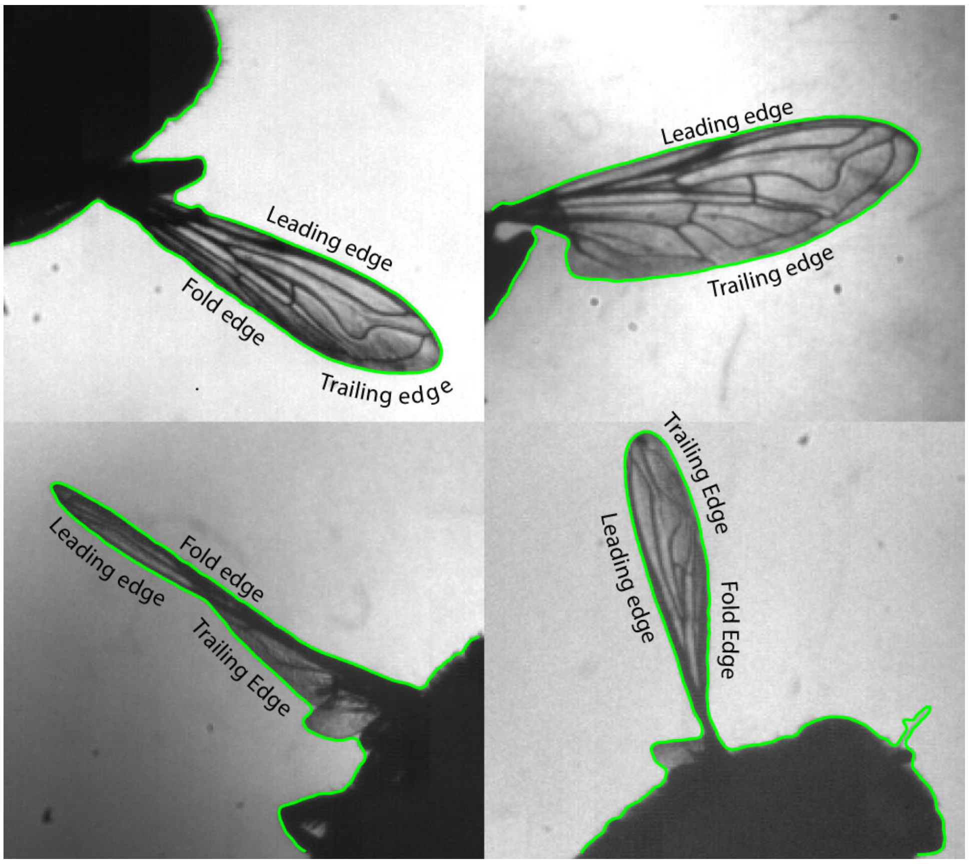


Figure 4.1: Boundary splines shown for 15L frame 21.

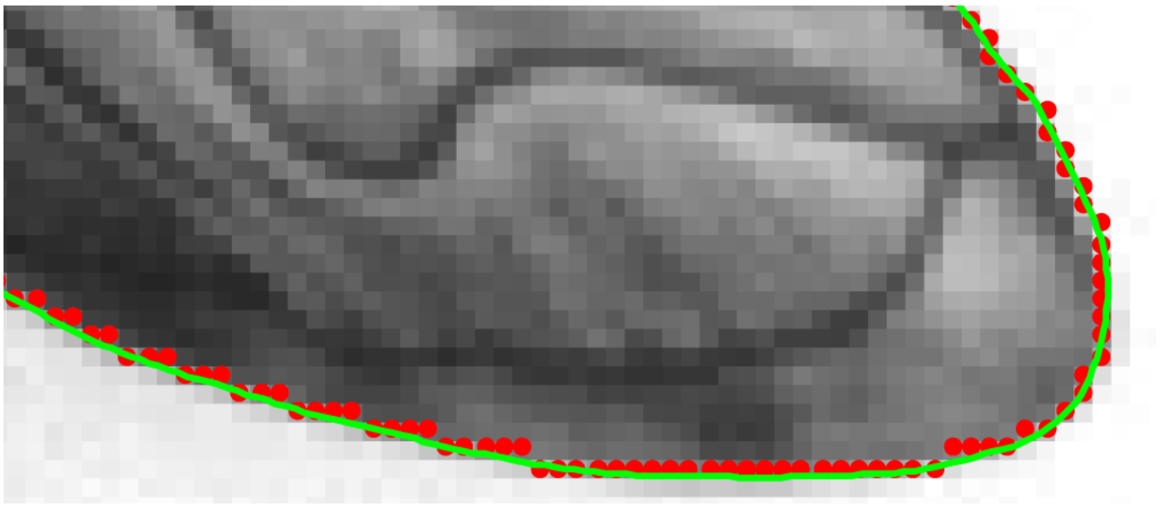


Figure 4.2: Smoothing spline (green) original silhouette perimeter pixels (red) - 15L frame 21 camera 1.

4.3 Epipolar intersections

The next step is to search for correspondences between points along the splines. As with the tuning process, the first step was to identify the 'best' view (again, call this view A) based on the hull area of five keypoints. Instead of using the motion model predictions for these points, the results of the keypoint tracking algorithm were used. This reduces the chance of picking a poor view. This view is most likely to contain a spline that sticks to the wing boundary, as it should be the most face-on view, with the least self-occlusion from folding or twisting. (There could well be occlusion from other parts of the fly however, in particular the legs.) For this reason matches were only sought between the spline points in view A and those in other views.

To generate a set of putative correspondences, $t = 0$ was taken as the starting point on the view A spline, and epipolar lines were projected from this point into the other views. As each spline is a sequence of cubic polynomials defined on intervals of t (whose start and endpoints are called 'breaks'), the intersection points can be found by testing each cubic function for an intersection with the line in the appropriate interval. The cubic function on the interval $t_j \leq t < t_{j+1}$ can be written in the form:

$$\begin{aligned} x &= m_1(t-t_j)^3 + m_2(t-t_j)^2 + m_3(t-t_j) + m_4 \\ y &= n_1(t-t_j)^3 + n_2(t-t_j)^2 + n_3(t-t_j) + n_4 \end{aligned} \tag{4.1}$$

If the epipolar line is given by the homogeneous coordinates $l = [a; b; c]$, then at the intersection(s):

$$ax + by + c = 0 \tag{4.2}$$

By substitution we obtain the new cubic polynomial equation:

$$(\frac{a}{c}m_1 + n_1)T^3 + (\frac{a}{c}m_2 + n_2)T^2 + (\frac{a}{c}m_3 + n_3)T^3 + (\frac{a}{c}m_4 + n_4 + \frac{c}{b}) = 0 \qquad (4.3)$$

where $T = t - t_j$. If r is a real root of this equation, and $r + t_j$ lies between t_j and t_{j+1}, then the epipolar line intersects the spline at $t = r + t_j$.

Every polynomial interval was checked in this way, collecting every intersection point in each of the views, before advancing to the next break, collecting new intersection points, and continuing until the end of the spline was reached. Figure 4.3 shows a complete set of intersection points for an example frame.

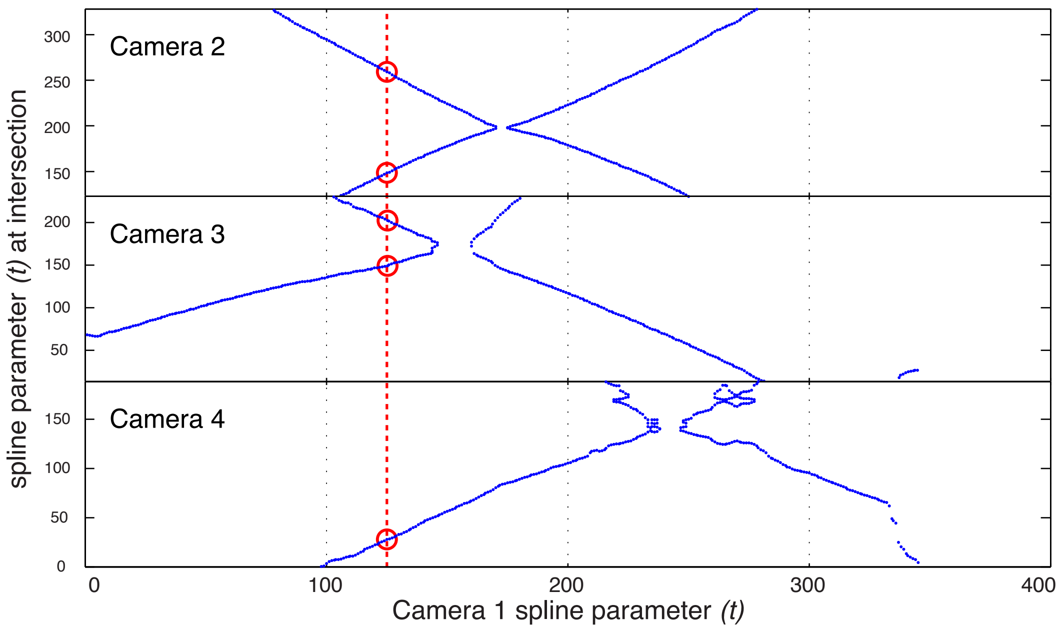


Figure 4.3: Spline parameters corresponding at intersections with epipolar lines from spline points in camera 1 (view A). Red circles show 5 intersections corresponding to the control point at $t = 125$. (15R frame 6)

4.4 Curve Matching

Each intersection point paired with its spline point in view A provides a putative correspondence whose visual rays intersect. Two tests were performed on each point to assess match quality. The first is the epipolar segment distance test, used once again as a test for outliers. As before, the 2-d location of the point in view A is used to generate the epipolar segments, and the point in the other views must be within a set distance (d) of the segments. A distance of $d = 3$ pixels was used. This is more restrictive than the value used in the tuning process (which allowed for misregistration error) but still allows for small edge localisation error and calibration error.

The second test assigns a score to each point using a similarity measure for pixel neighbourhoods around this point and the view A point. This measure comes from the curve matching algorithm of Schmid and Zisserman [33], described in Section 4.4 on page 94. They demonstrated that curvatures and tangents at projections of a curve point in two views determine the osculating plane at that point. This osculating plane induces a homography between the two views, which can map pixel neighbourhoods from one view to the other. Normalised cross-correlation on these neighbourhoods provides a similarity measure.

If $\boldsymbol{x}$ is a spline point in view A, with putative match $\boldsymbol{x}'$ (both in homogeneous coordinates), with curvatures $\{\kappa, \kappa'\}$ and tangent lines $\{\boldsymbol{l}, \boldsymbol{l}'\}$, normalised such that $l_1^2 + l_2^2 = 1$, then the homography (H) is given by the equation:

$$H = \boldsymbol{A} + \mu \boldsymbol{e}' \boldsymbol{l}^\top \tag{4.4}$$

where:

$$\begin{aligned}
\mu &= \frac{(b_1^2 + b_2^2)^{3/2} \kappa'}{|\boldsymbol{A} + \boldsymbol{e}' \boldsymbol{l}^\top| (\boldsymbol{a}^3 . \boldsymbol{x})^3 \kappa} \\
\boldsymbol{A} &= [\boldsymbol{l}']_\times \boldsymbol{F} \\
\boldsymbol{F} &\equiv \text{fundamental matrix from view A to second view} \\
\boldsymbol{a}^i &\equiv i^{\text{th}} \text{ column of } \boldsymbol{A}^\top \\
\boldsymbol{b} &= \text{adj}(\boldsymbol{A})^\top \boldsymbol{l} \\
\text{adj}(\boldsymbol{A}) &= [\boldsymbol{a}^2 \times \boldsymbol{a}^3, \boldsymbol{a}^3 \times \boldsymbol{a}^1, \boldsymbol{a}^1 \times \boldsymbol{a}^2].
\end{aligned}$$

The decision to use cubic splines to represent the edges allows the tangents and curvatures to be calculated in a straightforward manner. Tangent vectors can be obtained by differentiating Equation 4.1. Curvature requires the first and second derivatives according to the following equation:

$$\kappa = \frac{|\dot{x}\ddot{y} - \dot{y}\ddot{x}|}{(\dot{x}^2 + \dot{y}^2)^{3/2}} \tag{4.5}$$

As with Schmid and Zisserman, a 15×15 pixel window was used for normalised cross-correlation. Pixels were extracted from a window of this size, centred on x, using linear interpolation. Their locations were mapped into the second view (with H), and intensity values were extracted for comparison. In this way scores were assigned to every intersection point.

Figure 4.4 illustrates these tests. It shows an incorrect match, marked as an outlier and with a very low score, and a correct match that achieves a high score and is close to an epipolar segment. The transformed pixel neighbourhood appears stretched in this case, indicating the homography's sensitivity to errors in curvature and tangent estimation, but is adequate to distinguish between the good and bad match. For cases such as this, with one intersection on the leading edge, and one on the trailing edge, half of the pixels in the neighbourhoods will be relatively bright (corresponding to the background) and half of the pixels will be relatively dark (corresponding to the wing). As long as the orientation of the pixel neighbourhoods relative to the perimeter is fairly well preserved, the patterns of these dark and light pixels should allow the NCC calculation to discriminate between good and bad matches. These patterns can observed in Figure 4.5, which shows the neighbourhoods for this example.

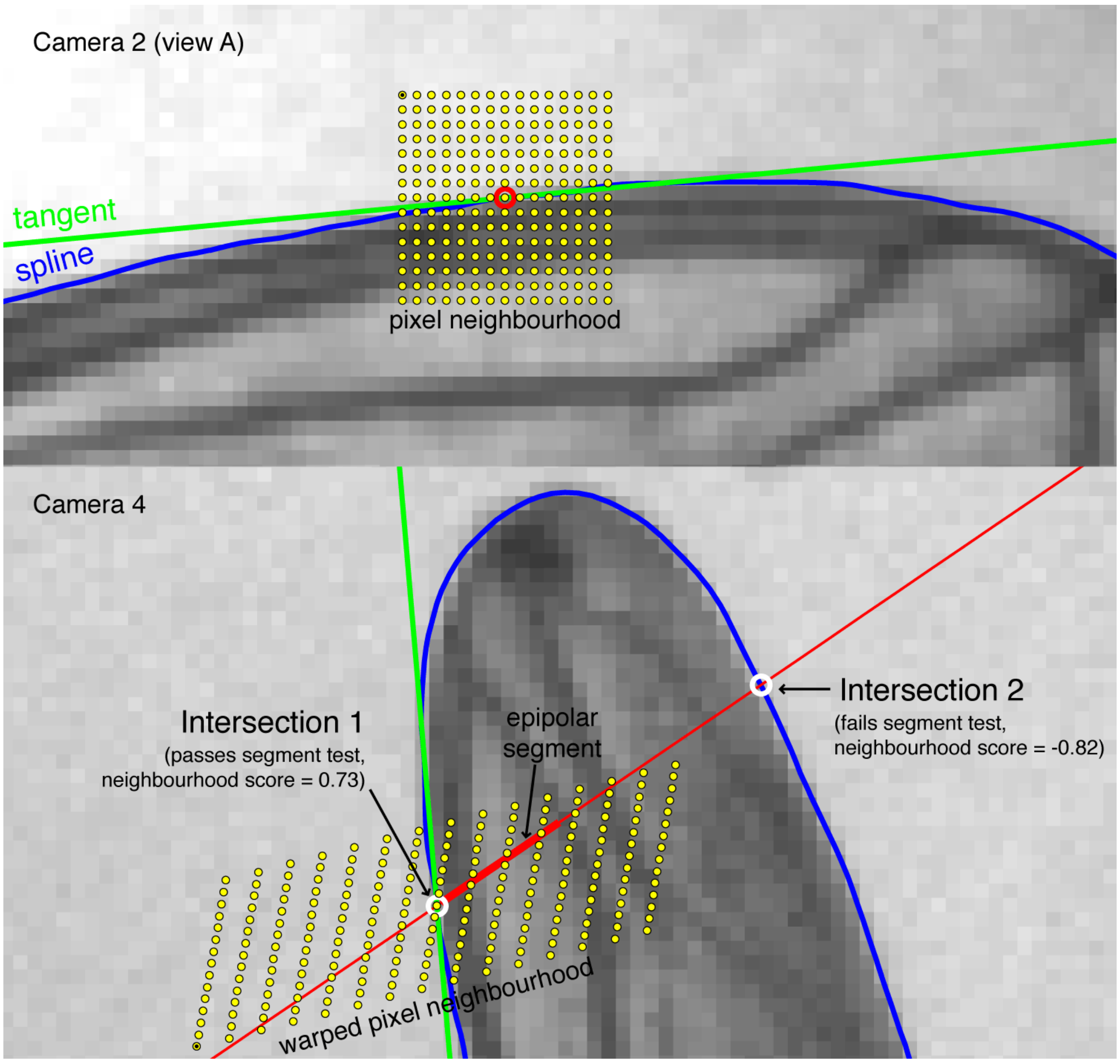

Camera 2 (view A)
tangent
spline
pixel neighbourhood
Camera 4
Intersection 1
(passes segment test, neighbourhood score = 0.73)
epipolar segment
Intersection 2
(fails segment test, neighbourhood score = -0.82)
warped pixel neighbourhood

Figure 4.4: Match candidates (white) for a spline point in view A. The correct correspondence (Intersection 1) is revealed by passing the epipolar segment test and achieving a high similarity score. (15L frame 21)

While a single intersection point paired with the view A spline point determines a 3-d location, a high score and passing the epipolar segment test do not guarantee that the 3-d point is on the wing boundary - the point will be close to the visual hull, but may have a high score by chance. Introducing constraints from a third view can help remove these unwanted points.

To say that a point lies on the perimeters of three silhouettes is equivalent to saying that

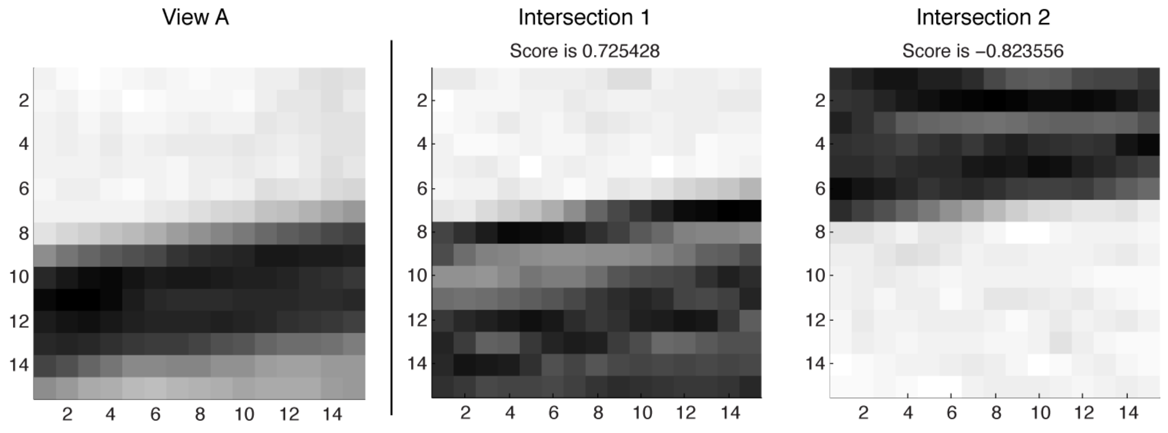


Figure 4.5: Pixel neighbourhoods for similarity measure. (15L frame 21)

it lies at the intersection of the three views' visual cones (see Fig. 1.19 p. 21). For rounded, bulky shapes such as the fly abdomen and the thicker parts of the fly legs, there will be few points that satisfy this condition: if a 3-d point on the abdomen projects to a silhouette perimeter in one view, a slight change in viewpoint would be likely to cause it to project to the silhouette interior. Unoccluded wing boundary points, however, should always project to the silhouette perimeters as the thinness of the wing makes it so unlikely that they'd project to the silhouette interior. This motivated a search for triplets of points: the view A point, and high-scoring epipolar-consistent intersection points from two other views.

This search was performed by examining every pair of intersection points from different views. Consider a four camera scenario for a frame in which view A comes from camera 1, and a spline point in this view has 2, 3 and 4 intersection points in cameras 2, 3 and 4, respectively. This means that there are 6 triplets that could be generated using cameras 2 and 3; 12 triplets using cameras 3 and 4; and 8 triplets using cameras 2 and 4: 26 in total. All triplets that failed to meet the following two conditions were discarded:

1. One of the intersection points must have a similarity score of greater than 0.6. This is the value found empirically by Schmid and Zisserman to be suitable for matching under a wide-range of conditions. Both intersection points were not required to have a high score as this proved too restrictive.
2. The sightline distance between the intersection points must be less than 0.2 mm.

A triplet was chosen among those remaining (if any) by selecting the one containing intersection points with the smallest sightline distance.

Next, the distance between consecutive 3-d points was examined, stepping from the start of the view A spline to the end. Points were grouped into chains using the rule that any two consecutive points separated by less than 1 mm are in the same chain, and separate chains otherwise. Any chains less than 6 points long were discarded. This combination of length requirements was found to be successful at removing the majority of the remaining abdomen and leg points. The points in chains which passed this test constitute the final boundary result.

Figures 4.6-4.8 show example boundaries from set 15 as projections into the original images. The resulting chains typically cover around half of the wing boundary and by and large their projections stick closely to the true locations. Figure 4.6 shows one of the better frames for set 15: all of the points are well-localised; the short, discarded chains were present on the legs; and most of the wing boundaries were covered. Figure 4.7 shows an example of a frame in which very few points were recovered for either wing. This was a very challenging case in which none of the cameras had a clear view - most of the wings are very nearly edge-on, and there is considerable self-occlusion and low contrast. Finally, Figure 4.8 shows a case in which a long chain of poorly-localised points have slipped into the results. This is an artifact of the shape of the visual hull: an incorrect pairing of intersection points happen to give 3-d locations that correspond to an edge of the visual hull. Figure 4.9 shows a section of the visual hull containing this wing area.[1]

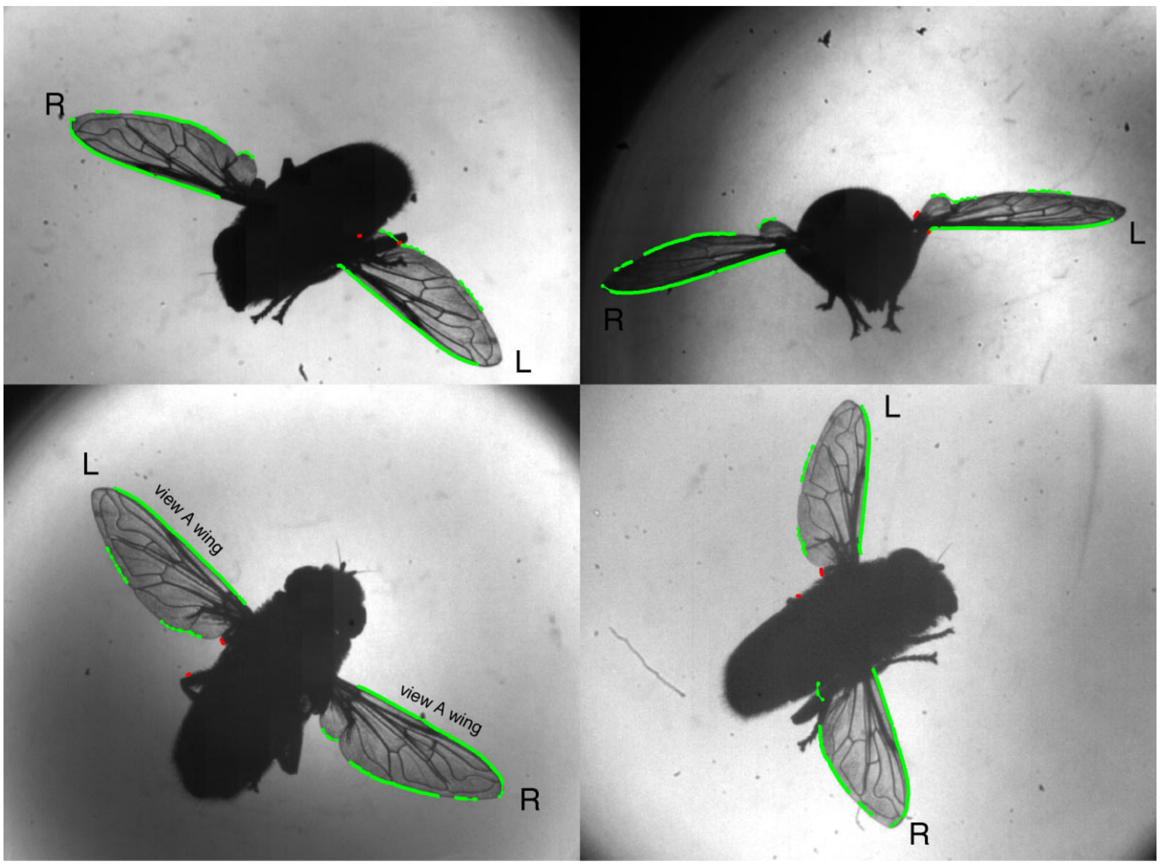


Figure 4.6: Well-extracted boundary (Set 15 frame 114)

[1]This hull was created by first testing a dense grid of 3-d points for consistency with the silhouettes, then calculating the convex hull, which in this case is approximately equivalent to the visual hull.

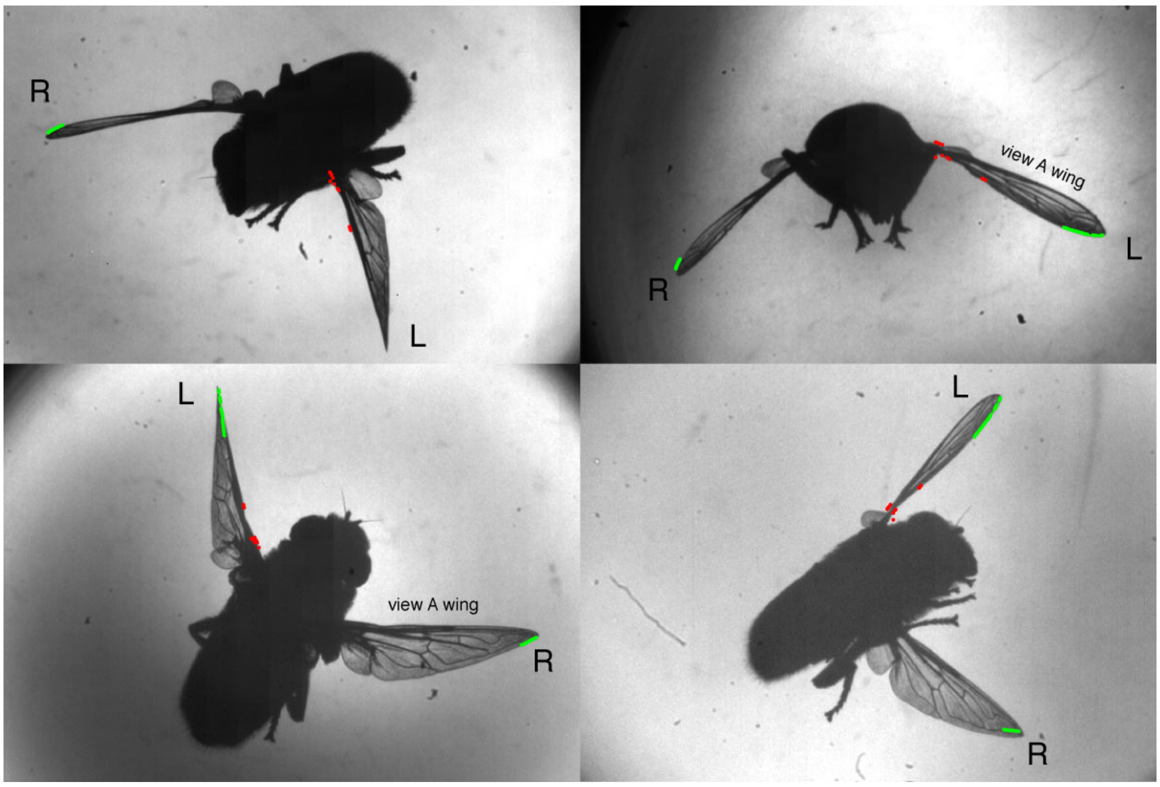


Figure 4.7: Sparsely extracted boundary. (Set 15 frame 66)

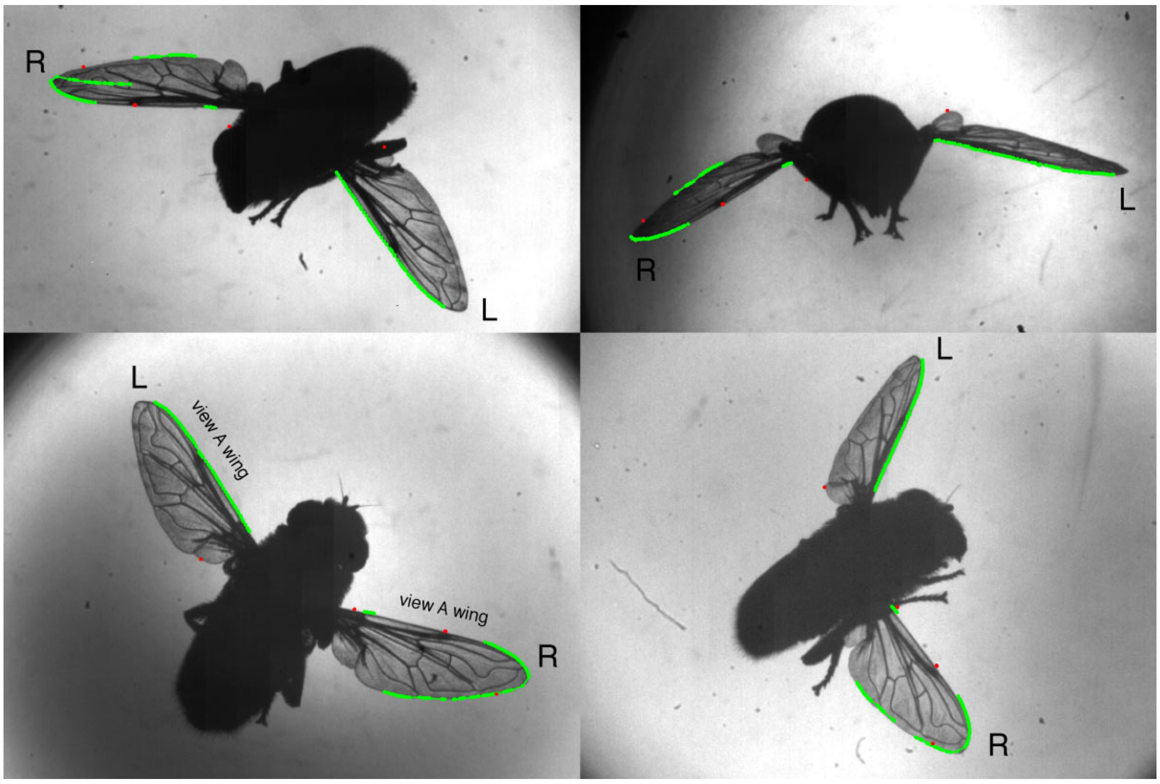


Figure 4.8: Boundary containing artifacts. (Set 15 frame 89)

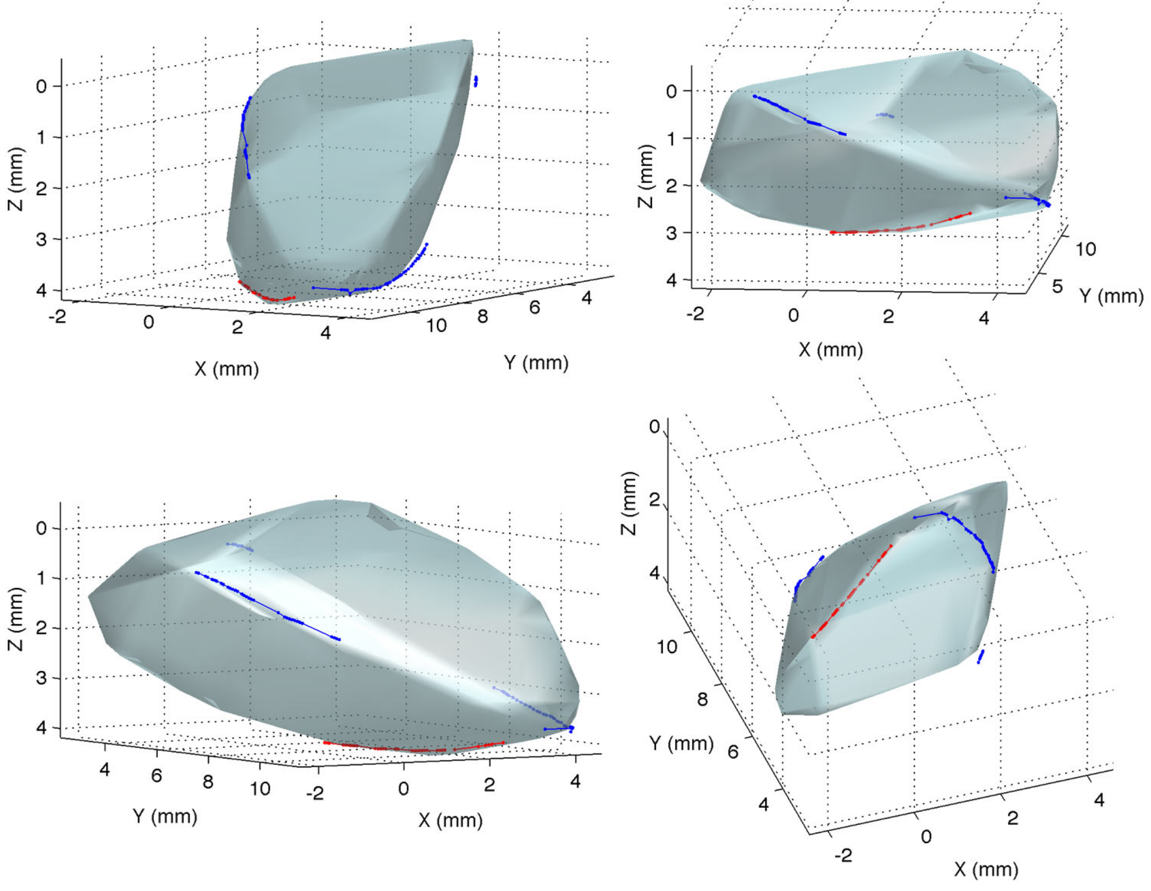


Figure 4.9: A section of the right wing visual hull, showing artifact points in red and well-localised wing boundary points in blue. (Set 15 frame 89)

4.5 Updating the wing surface

With a set of wing boundary points in place, the next step is to combine them with the results from the keypoint tracking algorithm. An important difference between these two types of results is that the keypoint results have known locations within a template image, whereas the wing boundary points can't be directly related to a template or to points in adjacent frames. Combining the results requires that the boundary points for a given frame be mapped to 2-d locations in the wing template. This map was constructed with the aid of the tracking results using the following steps for each frame.

1. A thin plate spline map was constructed for the wing surface using the tracking results and their positions in the general template, with a value of 0.8 for the smoothing parameter. Keypoints 11 and 12 were excluded from this surface fitting as they proved to be too unreliable.
2. This map was used to send each pixel from the perimeter of the general template (extracted from the silhouette) onto the 3-d surface.

3. The distance from each of the points in the boundary chains ('chain points') to each of the template boundary points was calculated, and each chain point was paired with its closest template point, as illustrated in Figure 4.10

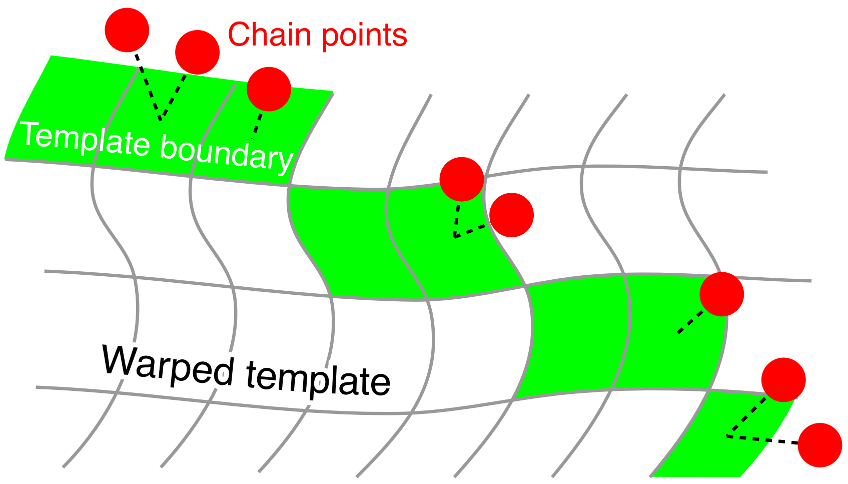


Figure 4.10: Pairing of chain points and closest template boundary point, warped into 3-d space by tracking results.

4. Each of the chain points was paired with its closest template point, and those chain points for which this separation was greater than 0.5 mm were discarded.
5. Where a template point is paired with more than one chain point, the chain point corresponding to the lowest view A spline parameter was kept and the others were discarded. Figure 4.11 shows an example of a final 3-d pairing.

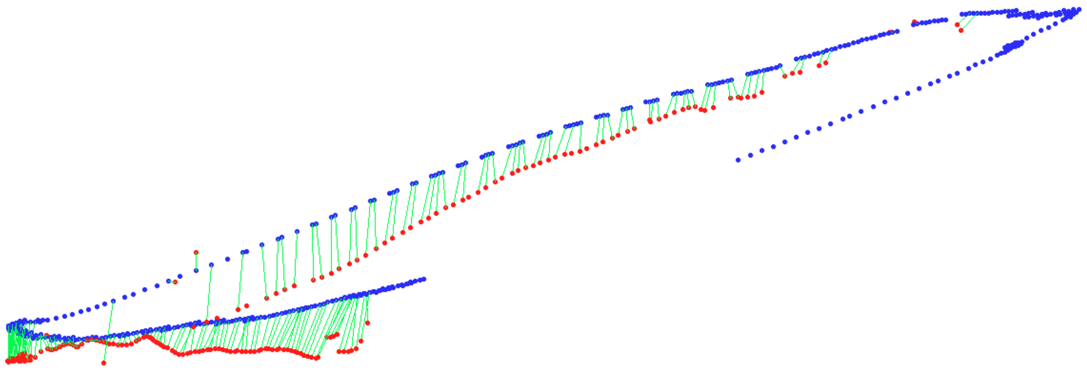


Figure 4.11: Example of final pairing between chain points (red) and wing boundary (blue).

6. The remaining 3-d boundary chain points can therefore be mapped back to the template, using the original 2-d locations of the template boundary points they have been paired with.

With correspondences established between wing boundary points in each frame and the template, new wing surfaces could be estimated that make use of this information. This is the final step in the algorithm, and the resulting wing surfaces are the end result. Smoothing thin plate splines were used to represent the surfaces. A large range of smoothing parameters was tested, examining the resulting wing shapes in 3-d and as 2-d projections, with the aim of striking a good balance between proximity to internal keypoints, proximity to the boundary and surface smoothness. Tests showed that a considerably lower value for smoothing parameter was required than had been used earlier in the algorithm, and a value of 0.001 was selected. Additional tests involved excluding certain internal keypoints. Points 11 and 12 were the least reliably localised of the points, as well as the most frequently occluded (due to being so close to the wing shoulder). It was found that excluding these points from both the boundary chain pairing process and the final wing fitting process resulted in the best wing shapes (as judged by their projections).

Comparing these surfaces to those produced by the internal keypoints alone reveals the influence of the boundary chain points. Figure 4.12 shows 3-d plots of the surfaces, alongside the boundary points that were used in the final fit, and those which were discarded as they are too far from the keypoints-only surface. Figure 4.13 shows the same surfaces as 2-d projections on the wing images. These figures demonstrate several key properties of the boundary inclusion process. The 3-d plot shows that the effect of the boundary points is to pull the edge of the surface towards them. In this example the effect is very noticeable for most of the leading edge and the section of the trailing edge near the wingtip. The effect on most of the wing area, however, is relatively minor: the internal keypoints only move very slightly, and the stretching for the most part happens around them.

The small effect of the boundary on the local area around the internal keypoints is a consequence of the way the boundary points were paired with the the template. The strategy of pairing points with the closest warped boundary pixel implicitly assumes that the keypoint algorithm results (which produced the warp) are reliable. It is only appropriate to use boundary points that are quite close to the warped pixels (0.5 mm was selected as a cut-off), as the chance of an inappropriate pairing increases with distance, and bad pairing for a chain of boundary pixels would cause the surface to be inappropriately squashed or stretched. By only pulling the surface towards these ‘safe’ sections of the boundary, the new surface is an improvement, but as it cannot compensate for large errors in keypoint localisation, it is only a minor one. Figure 4.12 shows how many valid boundary points can be discarded in this process, but it also shows an advantage: by keeping only points close to the initial surface, outlier boundary points, that appear on distant body parts such as the leg, are removed.

4.6 Algorithm Results

Wing surface results for the 81 frames of Set 23R are shown in thumbnail form in Figures 4.14 - 4.17. A complete set of results for all sets is provided in video form in the data DVD that accompanies this thesis.

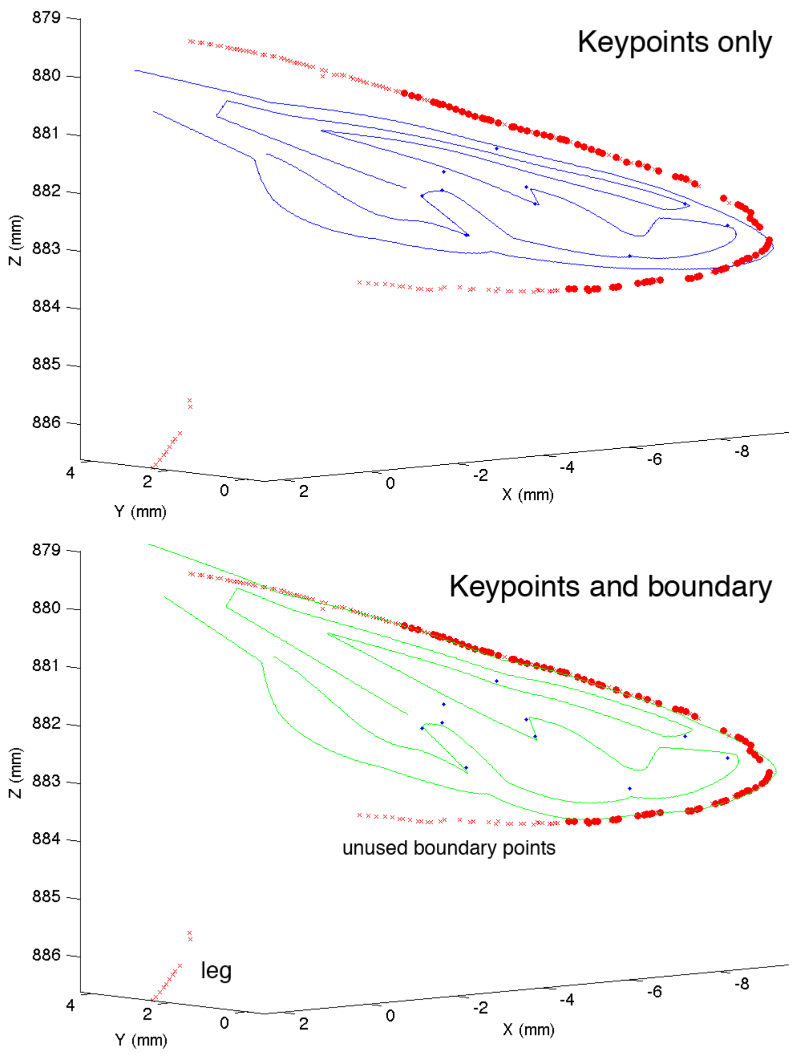


Figure 4.12: Effect of introducing boundary points into wing fit.

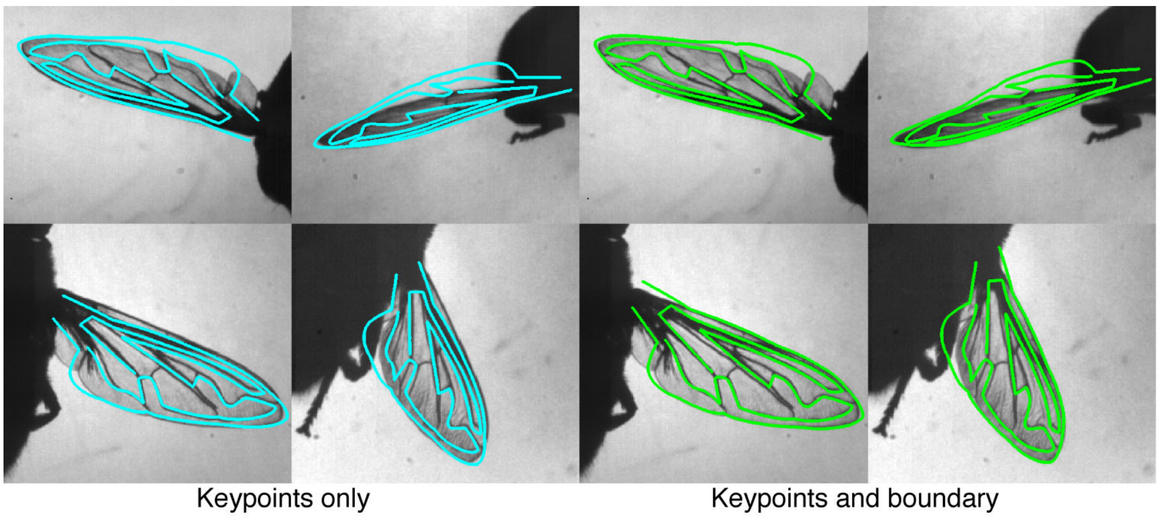


Figure 4.13: Effect of introducing boundary points into wing fit.

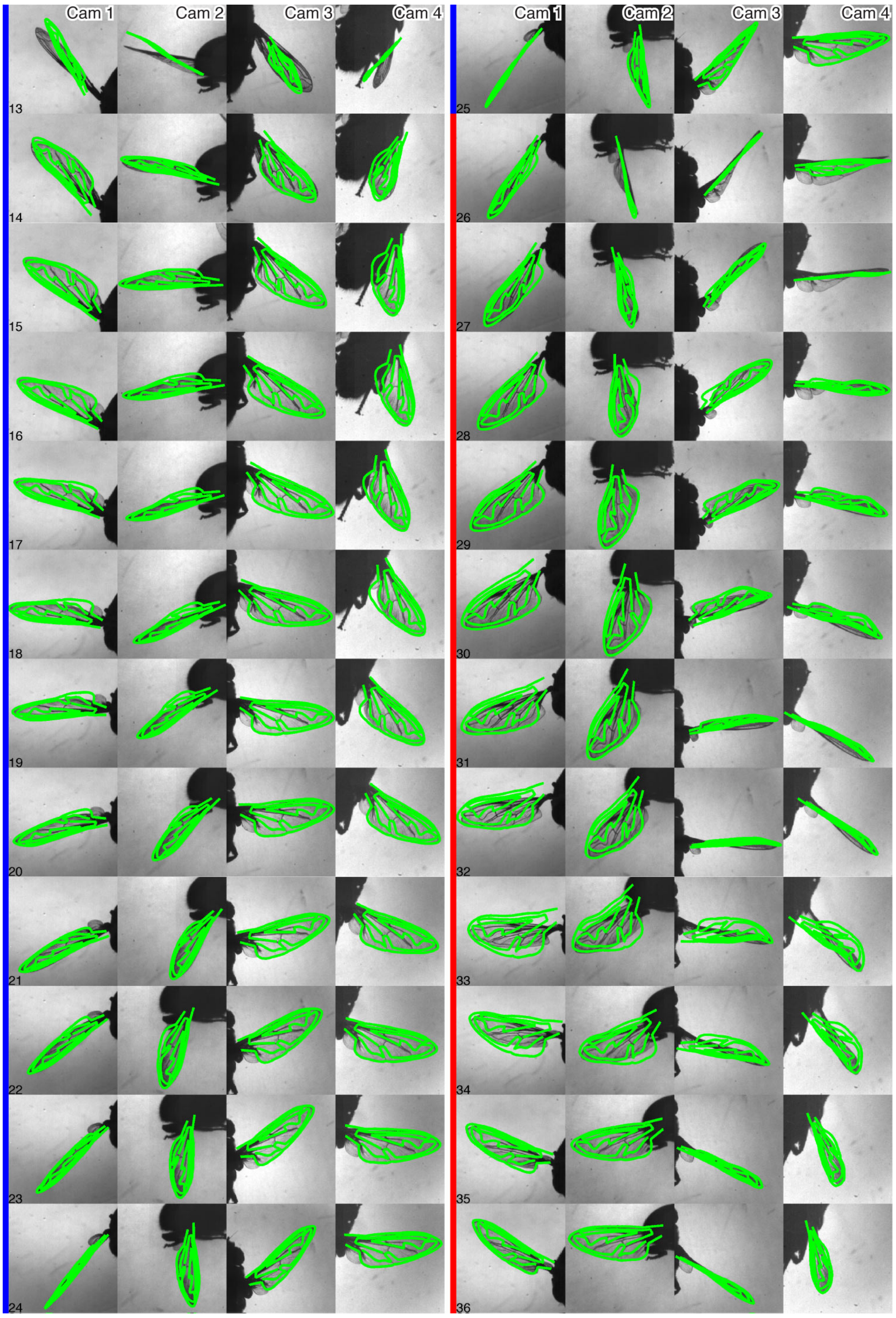


Figure 4.14: Final wing surfaces for Set 23R (1 of 4). Downstrokes and upstrokes are indicated with blue and red, respectively.

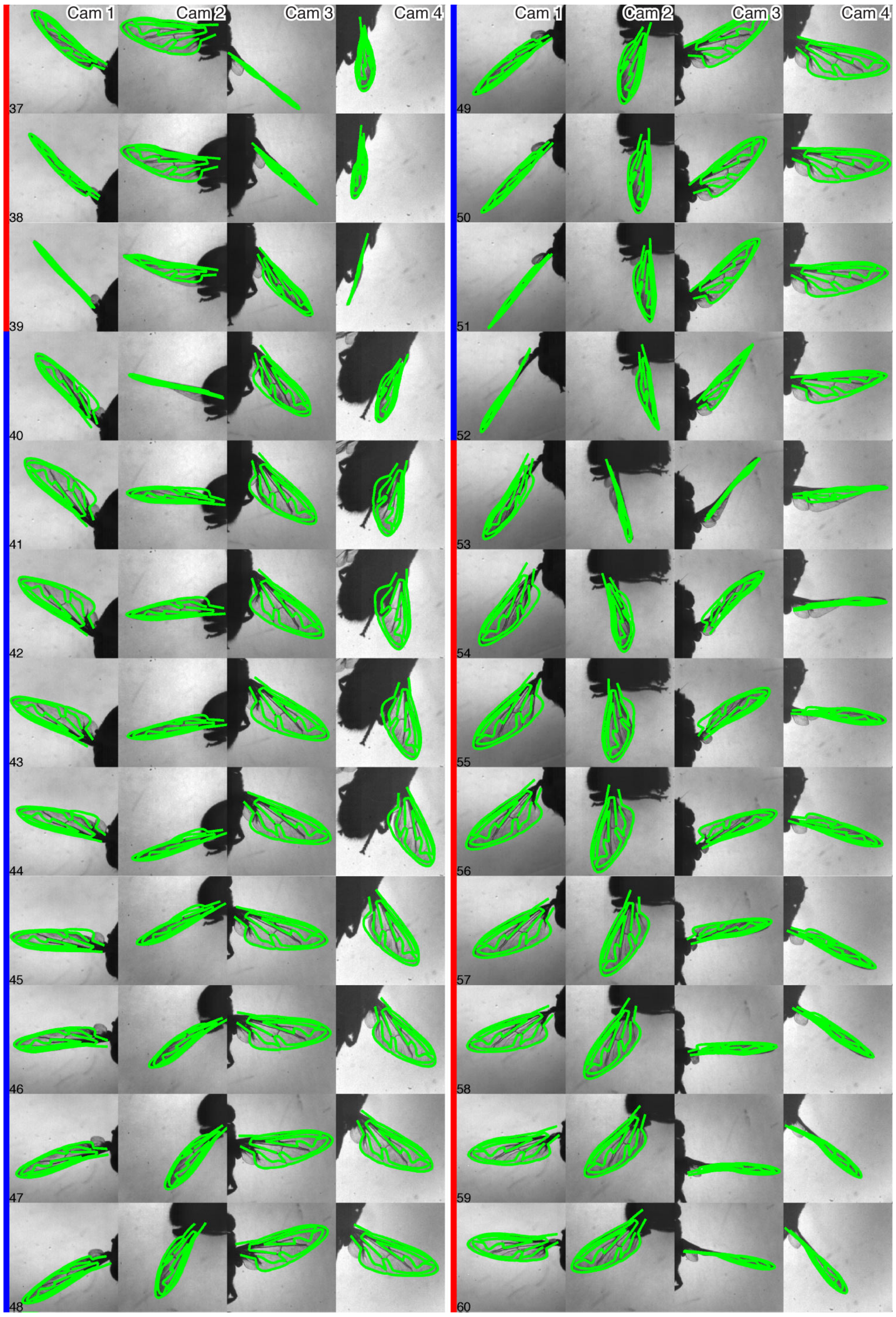


Figure 4.15: Final wing surfaces for Set 23R (2 of 4). Downstrokes and upstrokes are indicated with blue and red, respectively.

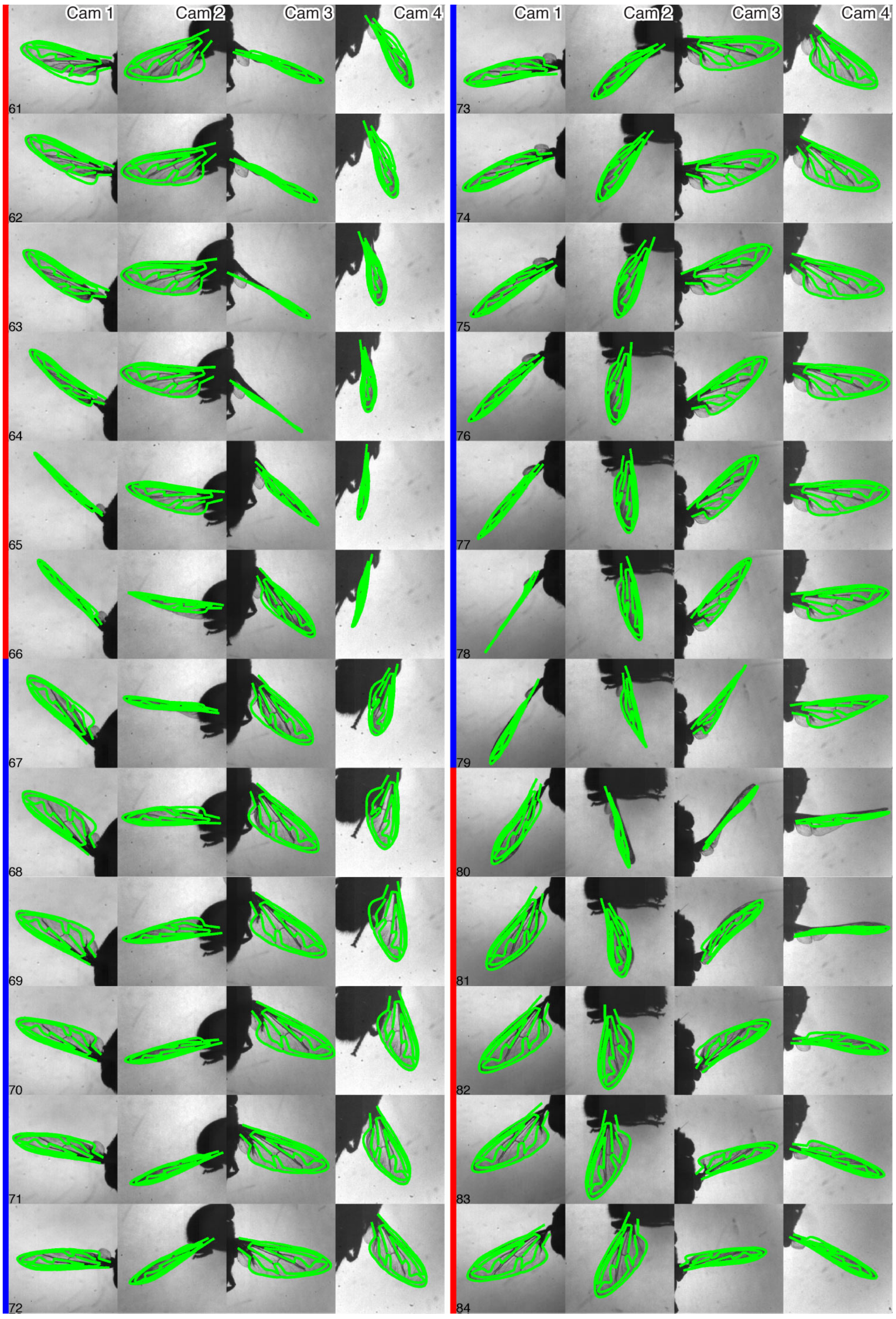


Figure 4.16: Final wing surfaces for Set 23R (3 of 4). Downstrokes and upstrokes are indicated with blue and red, respectively.

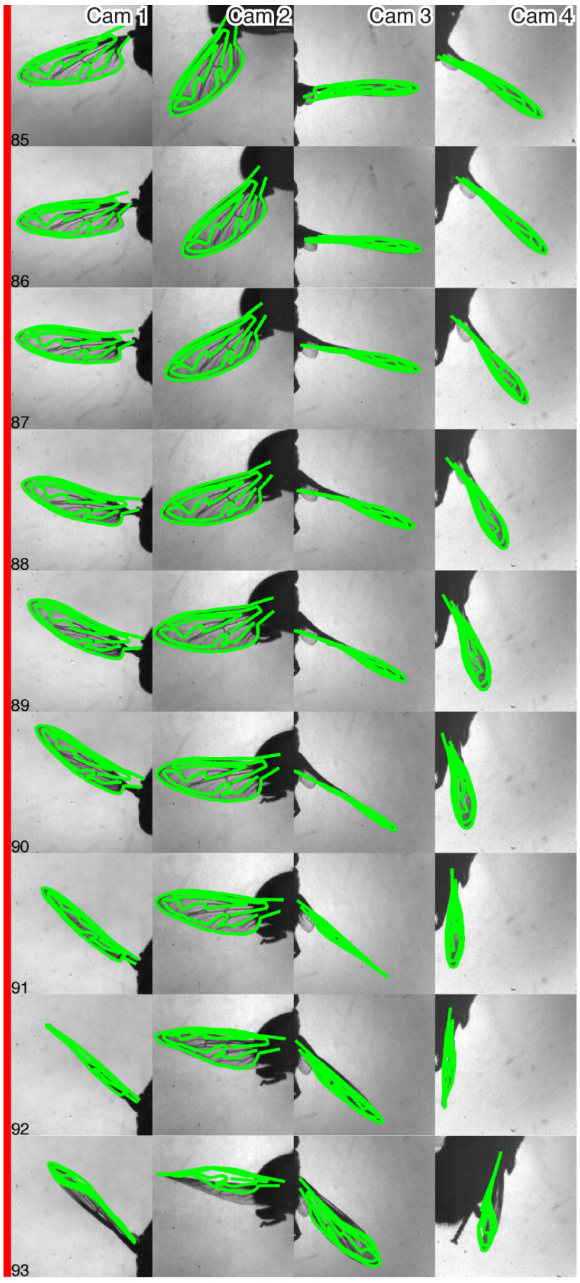


Figure 4.17: Final wing surfaces for Set 23R (4 of 4). Downstrokes and upstrokes are indicated with blue and red, respectively.

4.7 Discussion

Boundary points as quality measure

The focus of this chapter was on extracting and identifying boundary points that are located within a small, fixed distance (0.5 mm) of the surfaces extrapolated by the interior keypoints. These boundary points could be said to 'agree' with the keypoint algorithm results - those that 'disagreed' were discarded in the final stage. Another way of using the boundary points is to examine the extent to which they agree or disagree with the keypoint surfaces. If we assume that the full set of boundary chains extracted in Section 4.4 has only a small minority of outliers, then the full set of (minimum) separations between these points and the keypoint surface boundaries would say something about the accuracy of the keypoints.

If each of the boundary chain points are within a very small distance of the keypoint surface boundaries, the keypoints can be assumed to be reliable. If, however, boundary chain points are very distant, it is much less likely that the keypoints were accurately estimated. The key to this relationship lies in the fact that the keypoint algorithm and boundary extraction are entirely independent of each other.

Figure 4.18 shows boxplots of these separations for the first stroke of Set 23R. The corresponding final wing surfaces can be seen in Figure 4.14.

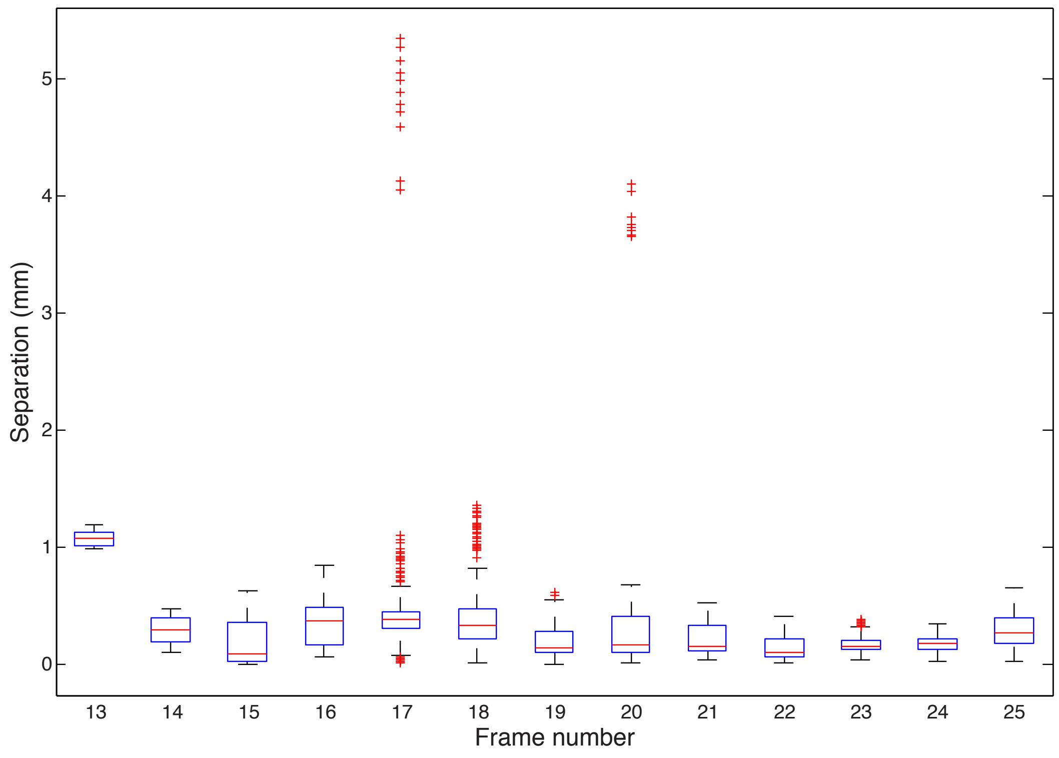


Figure 4.18: Boxplot of separations between extracted boundary points and keypoint algorithm boundary for Set 23R. Plots show the minimum sample, Q1 (25th percentile), median, Q3 (75th percentile), and maximum sample. Outliers are indicated with red crosses.

The first frame, frame 13, provides an example of consistently large separations: each boundary point is over 1 mm from the keypoint surface boundary, and so none was used to update the results. This correctly predicts a poor final wing surface. (Incidentally, it should be noted that frames at the very beginning and very end of the data set have the least support from neighbouring frames in the smoothing processes, so the failure to obtain an accurate wing shape in this case is not very surprising.)

One problem for this approach is that the proportion of outliers is unknown - particularly in light of the fact that coverage of the wing boundary is so variable. Artifacts and leg points which are incorrectly classified as boundary points could therefore distort the metric. For the purposes of verifying results for large datasets, however, this could provide a useful test for frames with poorly estimated wing shapes. Such frames could then be manually reconstructed by clicking on keypoints.

Boundary spline pairing

The method for determining a 3-d point from a set of perimeter spline intersections required a careful choice of parameters for the matching constraints - in particular, minimum score and maximum sightline distance between intersection pairs. The final choice to require a matching score of over 0.6 and a sightline distance of less than 0.2 mm was based on testing over various values, and examining the resulting boundary point sets. Values between 0.05mm and 2mm were tested for the maximum allowed sightline distance and minimum scores between 0.4 and 0.9 were tested. The choice of score threshold was also aided by examining all intersection scores for a given spline pair. Figure 4.19 shows a set of these scores in histogram form. The number of high scoring points spikes above a score of 0.6. The majority of correct matches will be contained in this spike. The smaller spike of very low scoring points is likely to correspond to points on the opposite edge (e.g. the trailing edge if the view A spline point is on the leading edge).

A number of alternative ways of applying these constraints were also investigated. Before attempting to find consistent triplets, high scoring pairs were sought. In this test the highest scoring point for each set of intersections was found, and any point in view A which couldn't be matched to a point with score over 0.6 was excluded. Another attempt created triplets using only the sightline distance constraint: choosing the pair of intersections whose sightline distance is lowest, then removing points for which this distance was over 0.2mm. The final method is an amalgam of these approaches.

Alternative boundary extraction: unordered matching

Before settling on the spline-based matching method inspired by Schmid and Zisserman, extensive testing of boundary extraction was performed based purely on matching individual silhouette perimeter pixels. This approach will be referred to as 'unordered matching' as points are not grouped into ordered chains.

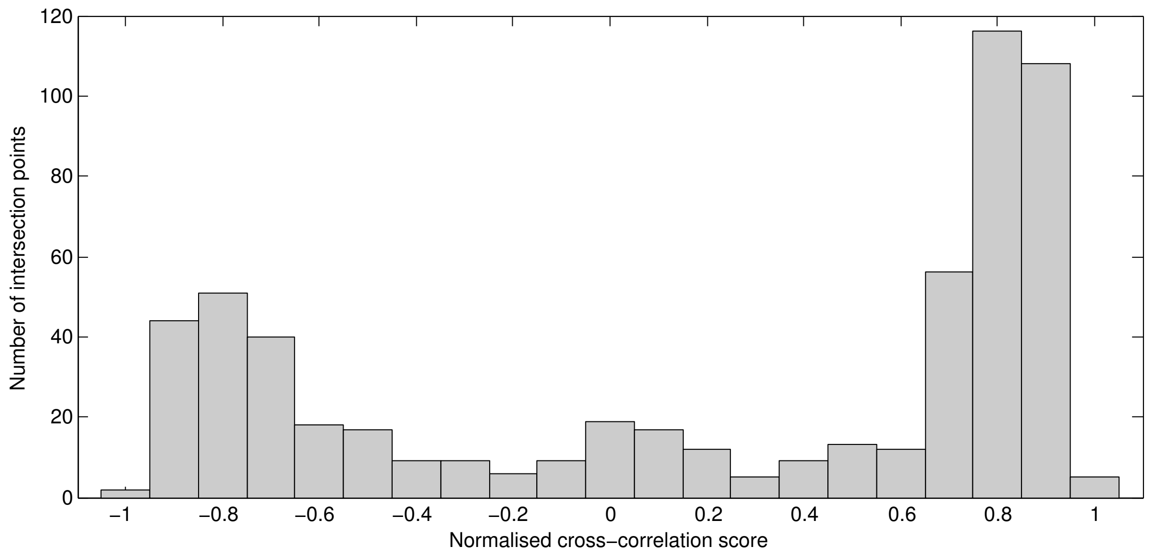


Figure 4.19: Histogram of scores for all intersections in camera 2, from spline in camera 1. (Set 15R frame 16)

This method also starts with the selection of a high area 'view A' which provides the edge pixels ('edgels') to be matched. The other cameras are also ordered, to be dealt with in turn (as view B, view C and view D). Starting with one of these edgels, epipolar lines are projected into the other three views, and rasterised to form 'epipolar strips' of a specified width in pixels. These epipolar strips are effectively boolean image masks, marking pixels as being inside or outside the strip. All edgels in the epipolar strips in view B are identified. Epipolar strips corresponding to these edgels are created in the other views. In view C, the set of edgels within the intersection of the view A and B epipolar strips are identified. As before, corresponding epipolar strips are created in the other views. Finally, in view D, the intersection of epipolar strips from views A, B and C are identified, and new epipolar strips generated again. The edgels in B, C and D that are contained within the strips from the other three views can then be paired with the starting view A edgel, and should specify a 3-d shape fairly close to the visual hull. This process of strip projection is illustrated in Figure 4.20.

A large number of strategies for choosing pairs (or triplets or quadruplets) to intersect in three dimensions were also tested. Briefly, these involved identifying the closest pixel centres to the epipolar lines, or identifying the points that were obtained for all 6 choices of views B, C and D, and examining every combination of 2, 3 or 4 edgels in order to find one which had the lowest reprojection error.

While this process provided a good set of boundary points for many frames, the spline-based method had substantial advantages. Using splines (or another continuous function) to parameterise the silhouette perimeters allows precise calculation of epipolar intersection and calculation of curvature and tangent which can be used in the similarity score calcu-

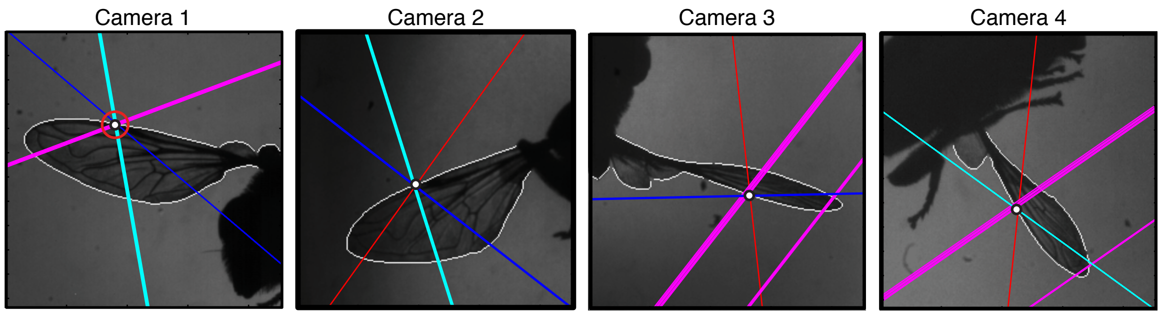


Figure 4.20: Simple epipolar matching. Epipolar strips (width 1 pixel) from cameras 1, 2, 3, 4 shown in red, magenta, cyan and blue, respectively. Final point selection shown as white dot. (Set 15R frame 16)

lation. Figure 4.21 shows a wing for which boundary points have been calculated using both methods. The results from the spline-based method appear much smoother and more tightly packed.

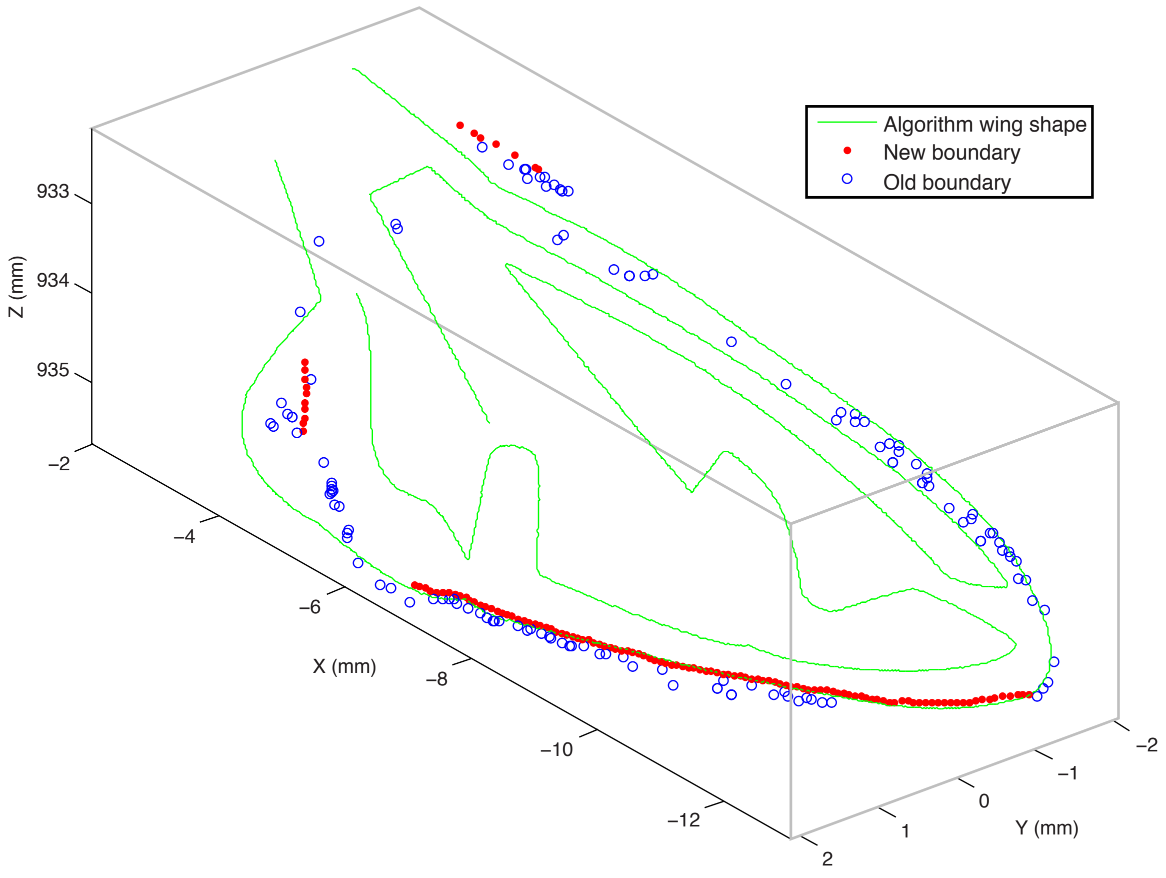


Figure 4.21: Old (blue) and new (red) boundary points, with final wing shape. (Set 15R frame 16)

Boundaries for surface approximation

As the wing is approximately planar for much of the wingbeat (particularly on the downstroke), a plane fit to well-extracted boundary points could provide an approximation of the wing surface. Figure 4.22 shows a frame for which a plane is fitted to the boundary points

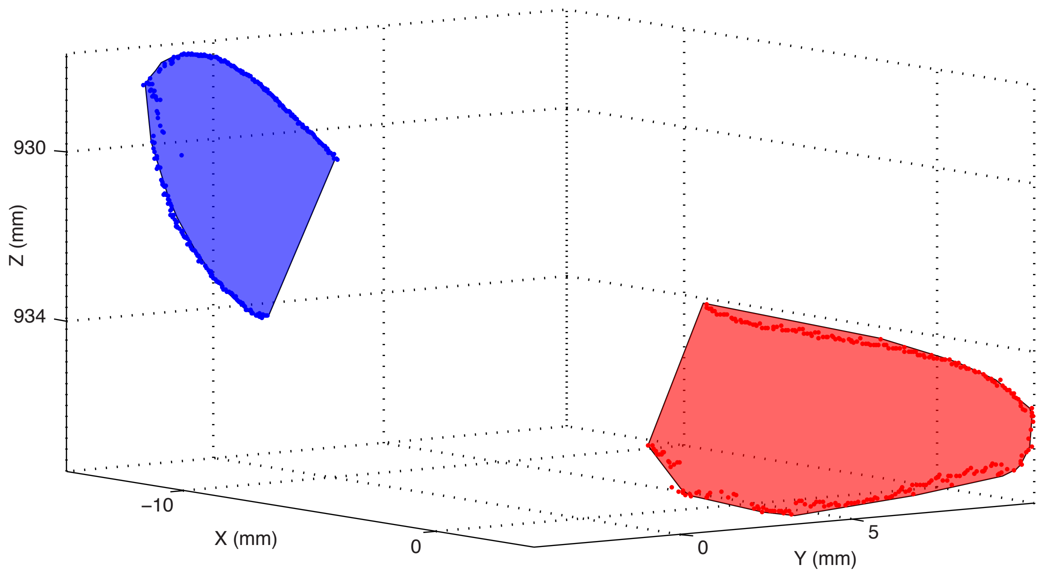


Figure 4.22: Planes fit onto boundary points, extracted using unordered matching method. Set 15R Frame 1.

(using RANSAC for robustness to outliers) and the convex hull of the projections onto the plane is isolated.

Such planes might have a use in identifying the highest area views. An optimal view would have an optical axis parallel to the normal of the wing plane. The absolute value of the dot product of these vectors would therefore give a measure of view quality.

Another potential use would be in providing a homography between views. If keypoints could be identified in one view (for example, using the tuning method of Chapter 3), intersecting their sightlines with the plane would give an estimate of their 3-d location that could then be projected into a second view, aiding a new search for these keypoints. This would provide an alternative to the current method of updating predictions based on view A tuning results.

Chapter 5

Manoeuvres

5.1 Overview

The algorithm described in the last three chapters was principally designed for the measurement of wings during hovering flight. The main assumptions of this algorithm are that adjacent wingbeats should look similar, and that azimuth and elevation of points within a suitable choice of wing axes have extrema near the start and end of wingbeats. These assumptions were made in order to ensure robustness to minor movements of the fly body and gradual changes in wingbeat patterns, but their generality should mean that the resulting algorithm can also be applied to a wide variety of motions.

In this chapter, the algorithm's ability to cope with a complicated manoeuvre is tested. The main deviation is to use a slightly different, though still very simple, method of estimating rigid body motion, tailored to the sequence. The sequence provides a further challenge in that it contains only three views, but this is still enough for the algorithm to work with. The following sections detail the adapted motion model, present results from intermediate stages, and show the final wing surface estimates.

5.2 The data

The manoeuvre set consists of 6 wingbeats, during which the hoverfly appears to pull up from a downward swoop. Its pitch angle increases throughout, while maintaining a forwards motion. Figure 5.1 shows a selection of frames during this manoeuvre, to convey the motion of the body.

As the right wing frequently moves out of the frame in two of the cameras, only the left wing was tracked.

5.3 Rigid body motion estimation

For hovering flight, the fly's rigid body motion was approximated as constant velocity motion along a straight line. The true motion is considerably more complex, with subtle ro-

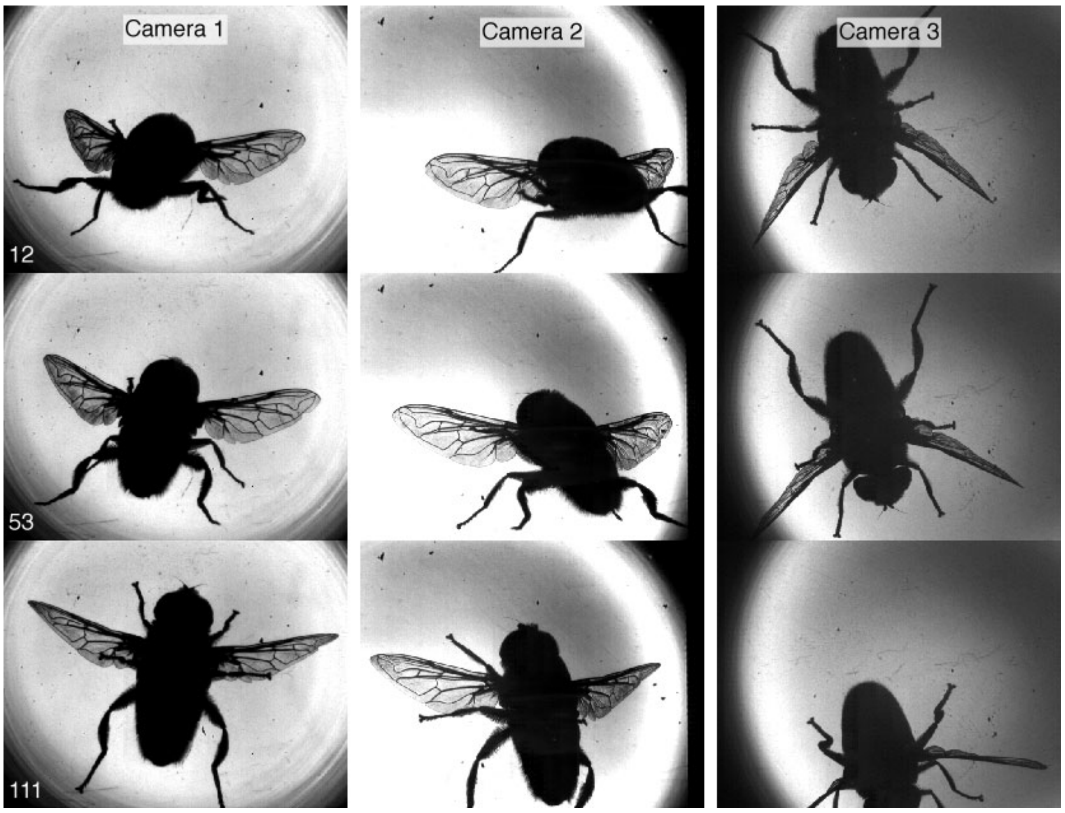


Figure 5.1: Manoeuvre images (Frames 12, 53, 111).

tations and accelerations, but this simplifying assumption keeps manual intervention to a minimum. The user manually identifies start and end locations of the shoulder, and a forwards and outwards direction, assumed to stay constant. For the manoeuvre set, a different motion assumption was used, but with a comparable level of user inter input.

The combined effects of the forwards motion and increasing pitch give the fly the appearance of rotating around a fixed point, some distance above its body. This concept was used as the simplifying assumption. Where previously a fixed linear velocity was assumed, the assumption in this case was of a constant rotational velocity around a fixed point. This velocity was estimated by fitting a three dimensional circle to a clicked set of points on the wing shoulder, clicking in 11 frames, spread across a total of 121. This involved the following steps.

1. Fit a plane to the full set of clicked 3-d points, obtaining a plane equation of the form $ax+by+cz=0$. For this, Peter Kovesi's *fitplane* function [85] was used. A unit vector perpendicular to the plane is given by:

$$\hat{\boldsymbol{n}} = \frac{1}{\sqrt{a^2+b^2+c^2}}[a,b,c]^\top. \tag{5.1}$$

2. Project each point ($\boldsymbol{p}$) onto the plane. Each projection ($\boldsymbol{p'}$) is given by:

$$\boldsymbol{p'} = \boldsymbol{p} - (d + \boldsymbol{p}.\hat{\boldsymbol{n}})\hat{\boldsymbol{n}} \tag{5.2}$$

3. Parameterise the plane as a 2-coordinate system (u, v), using as origin the point $\boldsymbol{p_0} = (-{}^{d}/_{a}, 0, 0)$, and unit basis vectors given by:

$$\begin{aligned}\hat{\boldsymbol{u}} &= \frac{\boldsymbol{p_1} - \boldsymbol{p_0}}{|\boldsymbol{p_1} - \boldsymbol{p_0}|} \\ \hat{\boldsymbol{v}} &= \hat{\boldsymbol{n}} \times \hat{\boldsymbol{u}}\end{aligned} \tag{5.3}$$

where $\boldsymbol{p_1}$ is the shoulder position in the first frame.

4. Calculate u, v coordinates for each projected point using:

$$\begin{aligned}u &= (\boldsymbol{p'} - \boldsymbol{p_0}).\hat{\boldsymbol{u}} \\ v &= \big([1\ 0\ 0](\boldsymbol{p'} - u\hat{\boldsymbol{u}} - \boldsymbol{p_0})\big) / \big([1\ 0\ 0]\hat{\boldsymbol{v}}\big)\end{aligned} \tag{5.4}$$

5. Determine the circle (in u, v coordinates) that passes through the first, middle and last projected point. Use the centre of this circle as the new origin for the the u, v coordinate system, updating the projected point coordinates accordingly.
6. Perform the circle fit in the $u - v$ plane to all projected points as a nonlinear least squares optimisation, using (0,0) as an initial estimate for the circle centre and radius from the 3-point estimate. For this, Matlab's *lsqnonlin* was used.
7. Convert the circle centre back to x, y, z coordinates using:

$$\boldsymbol{c} = \boldsymbol{p_0} + u\hat{\boldsymbol{u}} + v\hat{\boldsymbol{v}}. \tag{5.5}$$

Next, to obtain a value for rotational velocity (ω), the angle between the first and last projected point ($\boldsymbol{p'_A}$ and $\boldsymbol{p'_B}$, respectively) was divided by the number of frames between them ($f_B - f_A$):

$$\omega = \frac{cos^{-1}\big(\frac{1}{R^2}(\boldsymbol{p'_B} - \boldsymbol{c}).\boldsymbol{p'_A} - \boldsymbol{c}\big)}{f_B - f_A} \tag{5.6}$$

p'_A was used as the starting position of the shoulder in the first frame. This, combined with the rotational velocity, determines the approximate position of the shoulder in all subsequent frames.

The assumption that the fly's 'down' direction is always away from the centre of rotation and 'out' direction is always perpendicular to the plane allows for simple construction of the wing coordinate frame. If $\langle(\boldsymbol{p'_A} - \boldsymbol{c}) \times \hat{\boldsymbol{n}}, \boldsymbol{p'_B} - \boldsymbol{p'_A}\rangle > 0$, then $\hat{\boldsymbol{n}}$ can be used as the 'out' vector. Otherwise $-\hat{\boldsymbol{n}}$ is used. To produce a right-handed coordinate system, 'down' crossed with 'out' gives the 'forward' vector. In this way the change of basis matrix can be calculated for each frame. Figure 5.2 illustrates these basis vectors for each of the frames in which the shoulder position is identified.

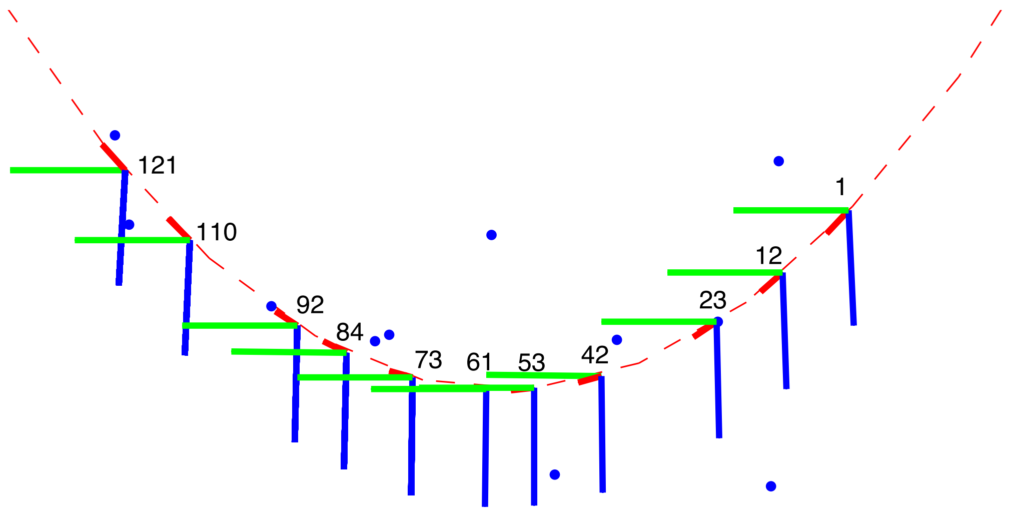


Figure 5.2: New basis vectors: forwards, out, and down shown in red, green, and blue respectively. Circle fit shown as dashed red, and initial clicked points shown as blue dots.

5.4 Intermediate steps

The rest of the algorithm was executed in almost precisely the same way as the other data sets, but with a few of minor modifications:

Reference wingbeat An initial attempt to use the reference wingbeat from Set 15R resulted in poor predictions in early wingbeats and poorly localised keypoints in the tuning stage. This led to increasingly poor predictions in later strokes from which the algorithm was not able to recover. By supplying the algorithm with clicked keypoint locations for the first wingbeat, the predictions for early strokes were much more accurate.

Spherical coordinate range In converting from cartesian to spherical coordinates during the two smoothing processes, it was necessary to solve for theta between -90 and 270 degrees (whereas the values had previously stayed within the range of -180 and 180 degrees). This is a result of the simplistic basis estimation procedure rather than extreme wing positions.

Extrema coordinate amplification Another consequence of inaccurate basis estimation is that the theta and phi extrema are not as widely separated as they would be in the correct basis. In order to ensure a good separation between coordinate values at neighbouring extrema, an additional pitch adjustment was introduced. Consider the theta and phi coordinates of a single stroke, plotted in a 2-dimensional axis. If the points move along a very steep or very shallow curve, the change in one coordinate will be very small, while the other is very large. With a small change in pitch of the coordinate frame (i.e. a rotation around the *y* axis), the extrema can be repositioned such that a vector between them is oriented at 45 degrees to the theta axis, thus creating a large separation between both extrema coordinate values. This is illustrated in Figure 5.3.

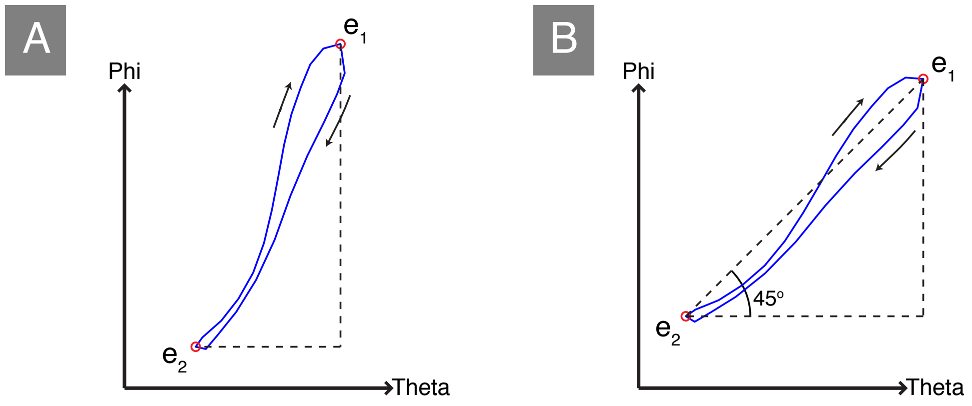


Figure 5.3: Extrema (e_1 and e_2) before and after pitch adjustment. (A) Very large phi separation, low theta separation; (B) large phi and theta separation.

A constant pitch adjustment was used for each frame, based on the angle between the first two extrema of keypoint 1. The transformation consisted of a positive rotation around the y-axis of α degrees, where:

$$\alpha = \tan^{-1}\left(\frac{\phi_2 - \phi_1}{\theta_2 - \theta_1}\right) - 45. \tag{5.7}$$

Here the first and second extrema are given by (θ_1, ϕ_1) and (θ_2, ϕ_2), respectively.

The effect of this rotation is illustrated in Figure 5.4. Notice the much greater similarity between corresponding strokes once the rotation has been applied. Without it, the predictions are wildly inaccurate, and can cause the tuning process to fail.

Wing boxes As the wings occupy a slightly larger area in each view than in previous sets, the size of the manually selected bounding box was increased to accommodate them (from 227 to 229 pixels).

In all other respects the algorithm is the same, down to individual parameter specification.

5.5 Algorithm Results

Wing surface results for the 118 frames of this set are shown in Figures 5.5 - 5.8. As with the hovering sets, these results are also contained in the accompanying data DVD.

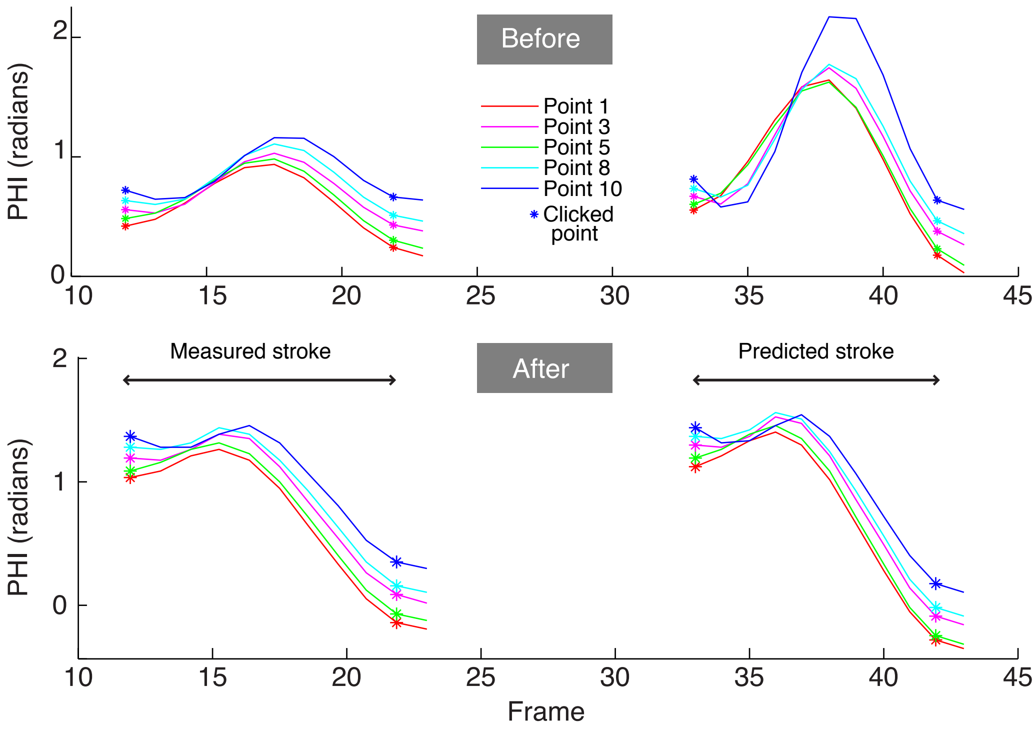


Figure 5.4: Measured and predicted phi values before and after pitch adjustment. (Strokes 2 and 4)

For reference, the extrema and 'clicked' frames are as follows:

Extrema	1	12	23	33	43	53	62	72	82	92	101	111	119
Clicked frames	1	12	23	33	42	53	61	73	84	92	103	110	121

5.6 Discussion

There are several lessons to take away from the results. The first is the level of independence of the upstroke and downstroke in terms of reconstruction accuracy. The patterns of good and bad wing shape recovery are well divided between downstroke and upstroke. The set begins with a downstroke (1-12) that is well recovered, as are the following upstroke (13-23) and downstroke (24-33). The second upstroke (34-43) goes off track very quickly, with tuning unable to correctly localise keypoints. From there, each subsequent upstroke (54-62, 73-82, 93-101, 112-119) is very inaccurate, while almost all frames of the remaining downstrokes (44-53, 63-72, 83-92, 102-111) have remarkably good projections. The downstrokes are able to retain their accuracy as a consequence of the smoothing and prediction strategy: each set of downstroke predictions is primarily based on the previous downstroke, and poorly tuned points in neighbouring strokes have little impact on these predictions as they are assigned low smoothing weights.

The poor upstroke measurements can be traced back to a result from the the smoothing process for upstroke 1, illustrated in Figure 5.10. Keypoint 10 in frame 16 was well-localised in the tuning process, but in the process of converting it to spherical coordinates,

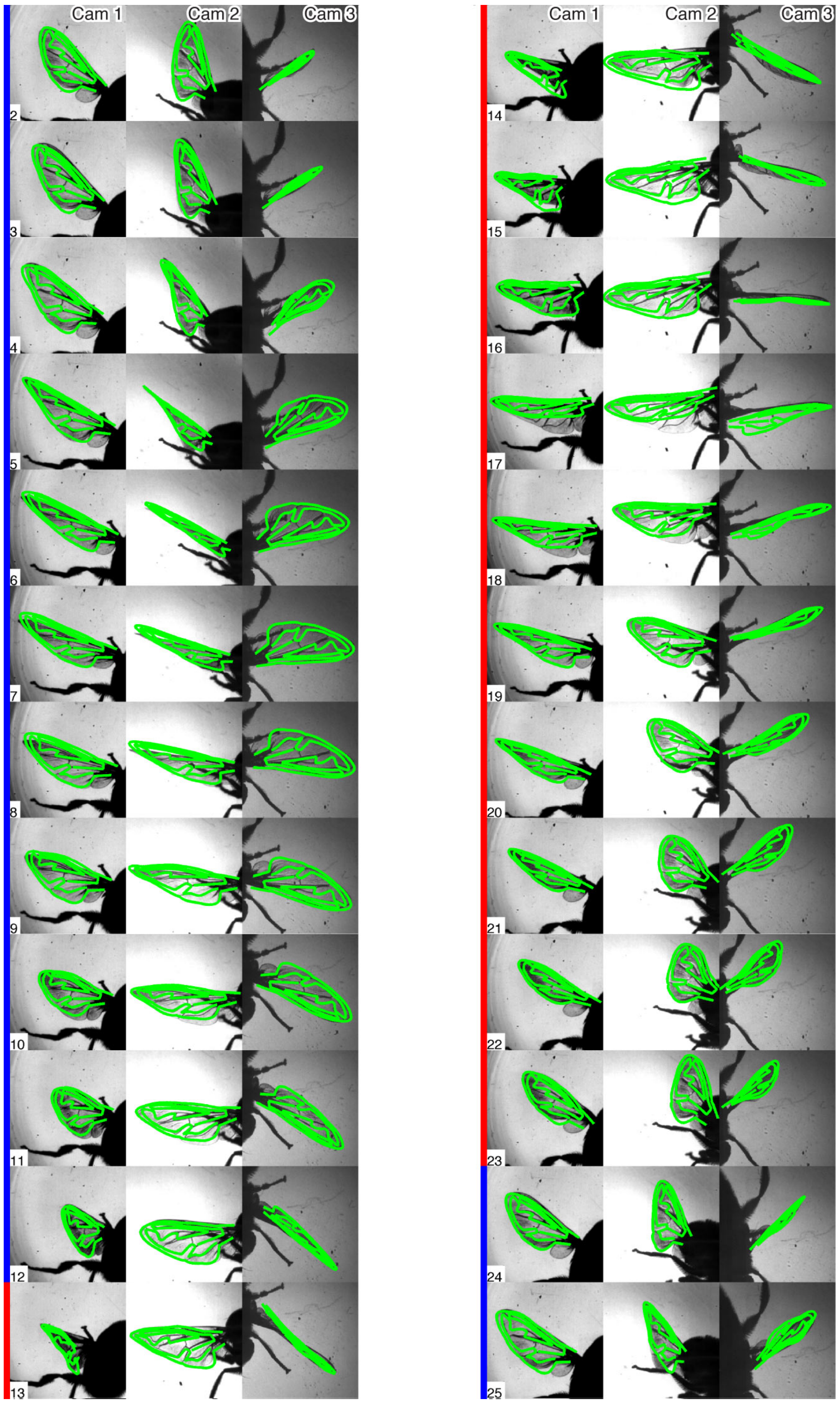


Figure 5.5: Final wing surfaces for manoeuvre set (1 of 5). Downstrokes and upstrokes are indicated with blue and red, respectively.

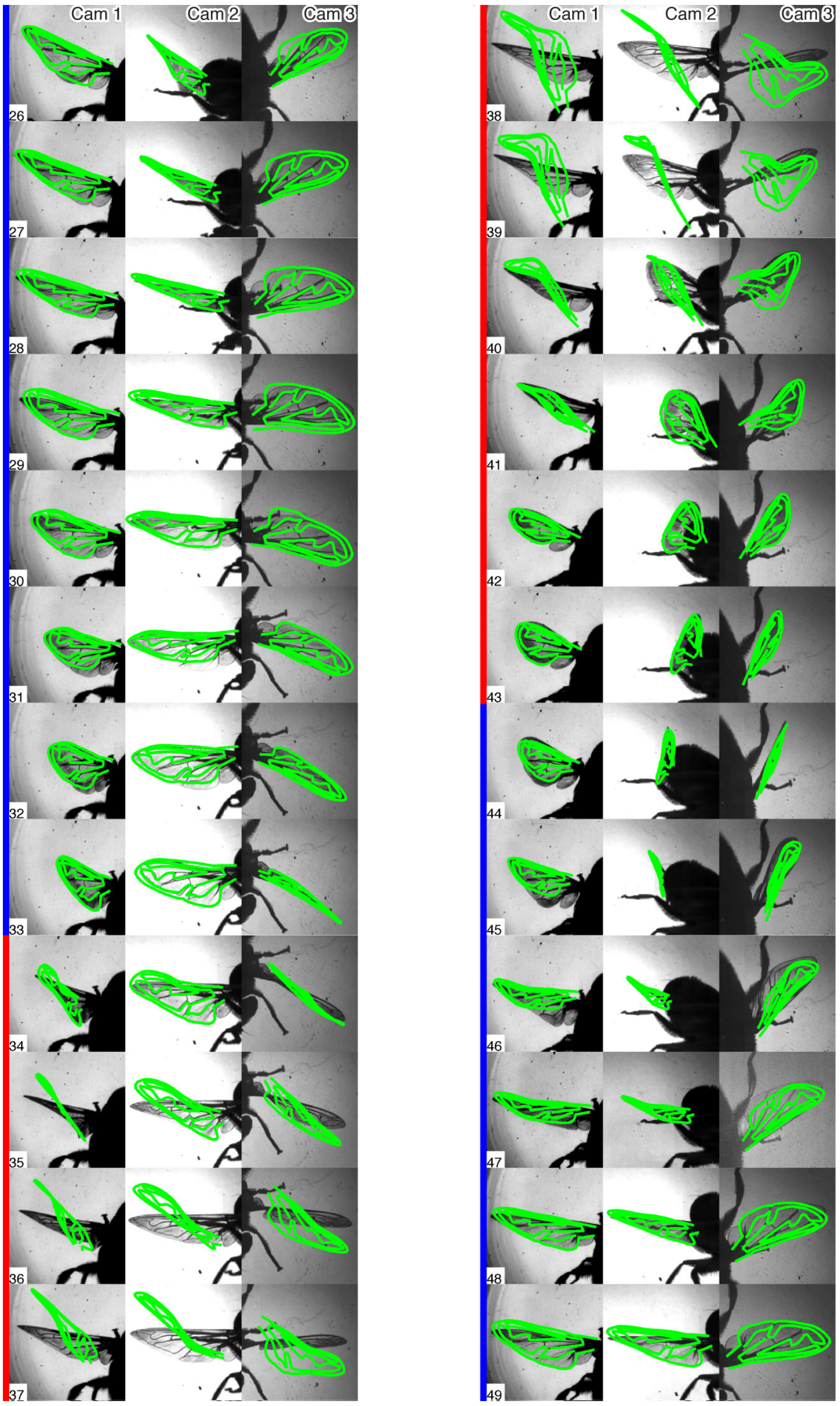


Figure 5.6: Final wing surfaces for manoeuvre set (2 of 5). Downstrokes and upstrokes are indicated with blue and red, respectively.

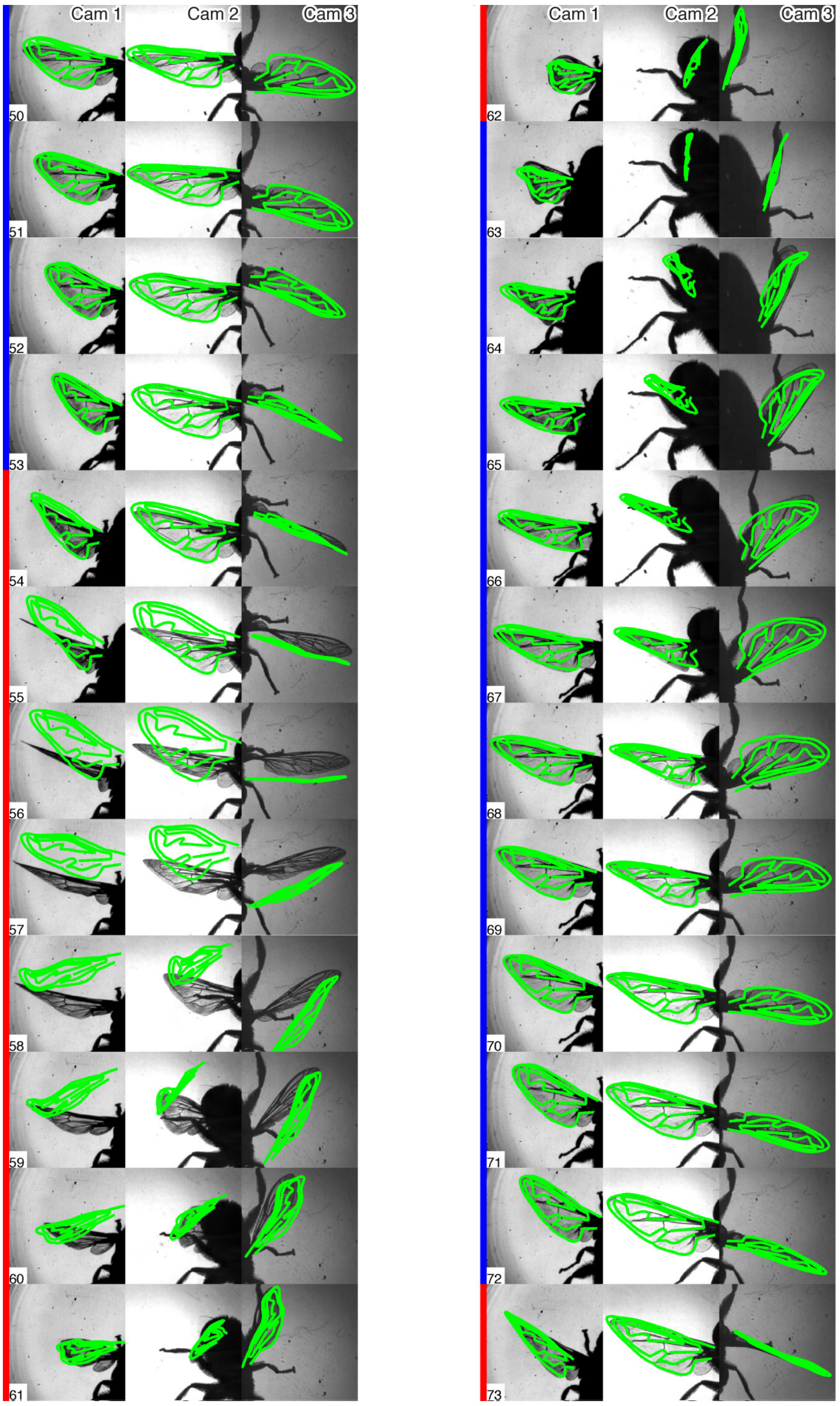


Figure 5.7: Final wing surfaces for manoeuvre set (3 of 5). Downstrokes and upstrokes are indicated with blue and red, respectively.

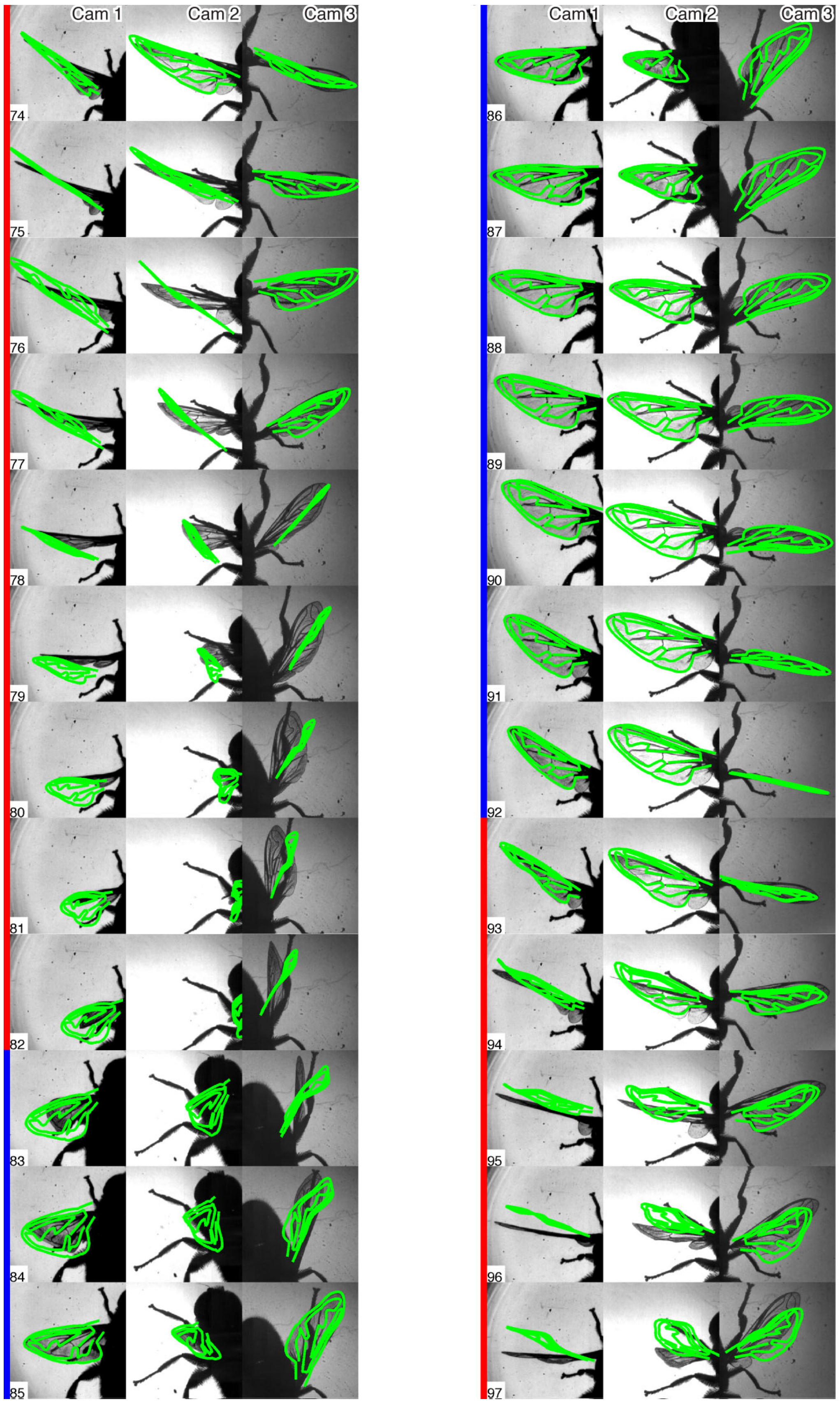
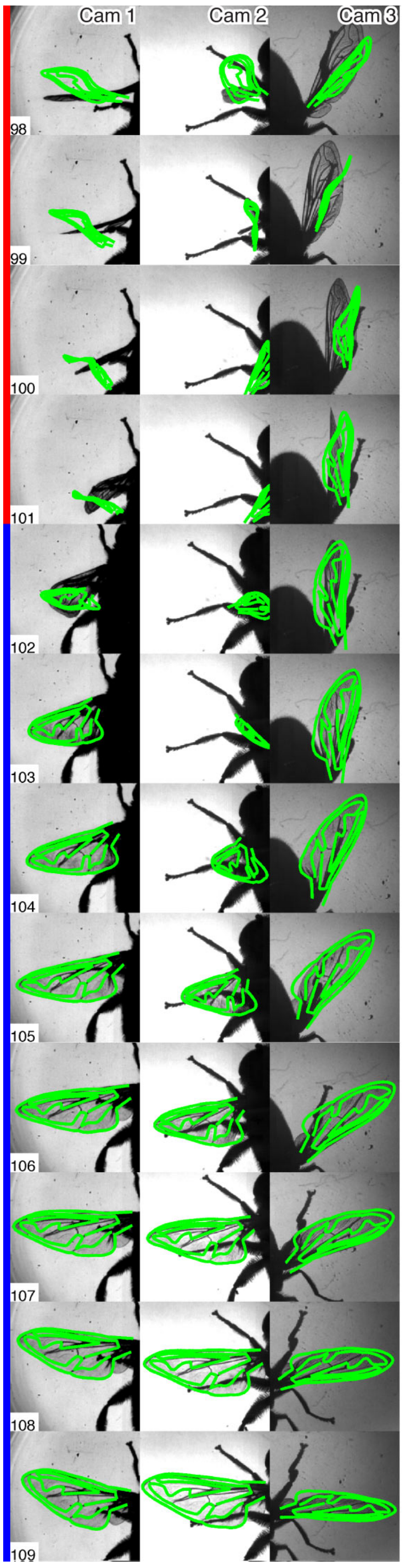


Figure 5.8: Final wing surfaces for manoeuvre set (4 of 5). Downstrokes and upstrokes are indicated with blue and red, respectively.



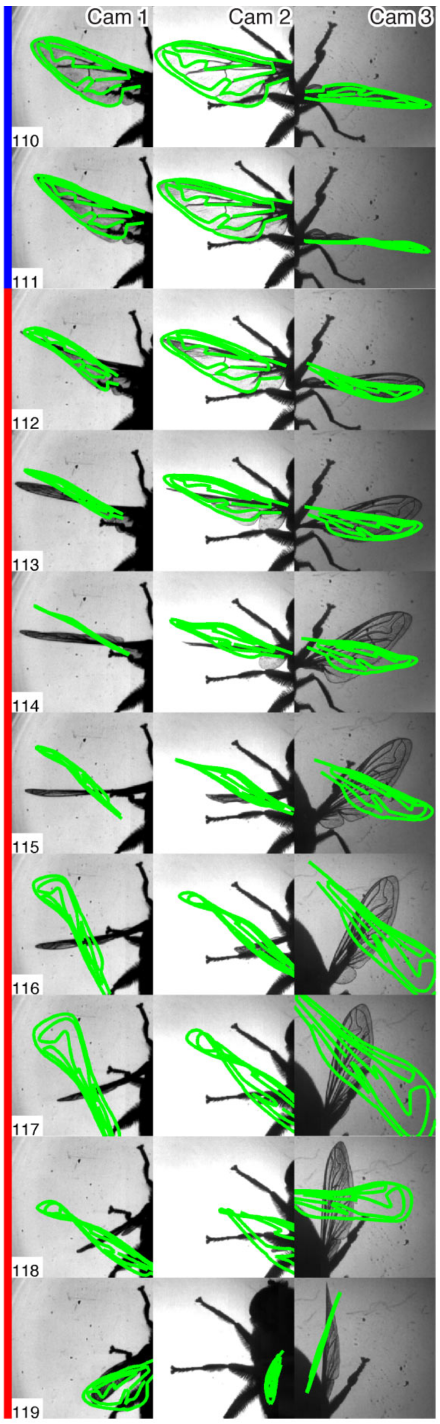


Figure 5.9: Final wing surfaces for manoeuvre set (5 of 5). Downstrokes and upstrokes are indicated with blue and red, respectively.

it was mislabelled as having a very high theta value of just under 270 degrees, rather than the more suitable low value of just below -90 (the low 'cutoff' value for the conversion). As an inlier with high weight it was able to distort the results of the smoothing process and therefore the prediction process for the next upstroke.

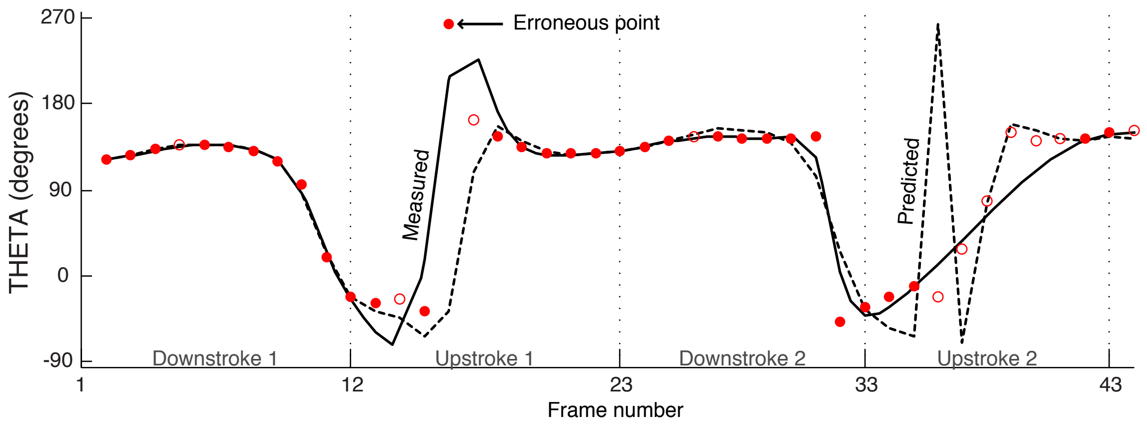


Figure 5.10: Measured (i.e. smoothed) and predicted theta values for keypoint 10. Tuned points are shown in red, with empty circles for outliers. The marked 'erroneous' tuned point has a large impact on the predictions for upstroke 2.

As theta is the angle away from the positive (forwards) x axis in the x-y plane, its values would ordinarily be between 0 and 180 degrees. The problem therefore stems from the choice of basis and shows the algorithm can be tripped up by inadequately accurate basis estimation. As manoeuvres will by definition involve complex motions this suggests that a higher level of manual supervision is required to initialise the algorithm, which could take the form of inspecting basis vectors and at an evenly spaced set of frames, correcting them if necessary and interpolating between them.

In the final stroke of the set, the upstroke in frames 112-119, the poor projections are partly the inheritance from previous stroke failures, but they were also inevitable from the very low wing areas and very high degree of occlusion. From 114 onwards there is only one view (camera 3) in which keypoints could be recognised, and even in this view there is significant occlusion from a leg. In the final frames from camera 2 the wing is entirely occluded by the body. This underscores the importance of good camera positioning. It also demonstrates that a set of views which yield high wing areas in early frames will not necessarily maintain their quality for later frames. One should therefore attempt to work with as many views as possible in order to increase the proportion of usable frames.

Chapter 6

Validation

In the previous chapters wing surface results were presented as two-dimensional projections, allowing them to be qualitatively assessed by how well projected wing veins and the wing boundary line up with the original images. In this chapter the accuracy of the algorithm is quantified. An accuracy metric is proposed which compares the results against a set of manually identified points (which will serve as a 'ground truth'), and additional analysis quantifies the effects of errors from calibration and errors from the manual identification process.

6.1 Calibration error

The calibration error can be assessed as the 'reprojection error' of the image points used to fit the calibration parameters. The image points were identified as the centroids of the dots in the calibration grid. Corresponding points can be intersected based on the final model parameters to create a 3-d point which can then be projected back into the images. The separation, in pixels, between an original image point and its reprojected point is the reprojection error. By collecting this error for every reprojected point for every calibration dot in each camera in every calibration frame we have a measure of the error introduced by imperfect calibration. Any 3-d points estimated from these cameras will be affected by this error.

Figure 6.1 shows this error for one of the hovering sets (Set 23), drawn from 11,939 points. The others are similar. From this figure, one can draw the following conclusions:

- The reprojection error varies by camera. In camera 4 the maximum observed error is half a pixel less than for camera 1.
- Less than 25% of the points project to within 0.1 pixels of their original locations.
- The largest reprojection error is 1.35 pixels. If the separation between a manually identified (i.e. 'clicked') image point and the projection of a 3-d point measured by the algorithm is less than 1.35 pixels, they are equivalent as far as the calibration is concerned.

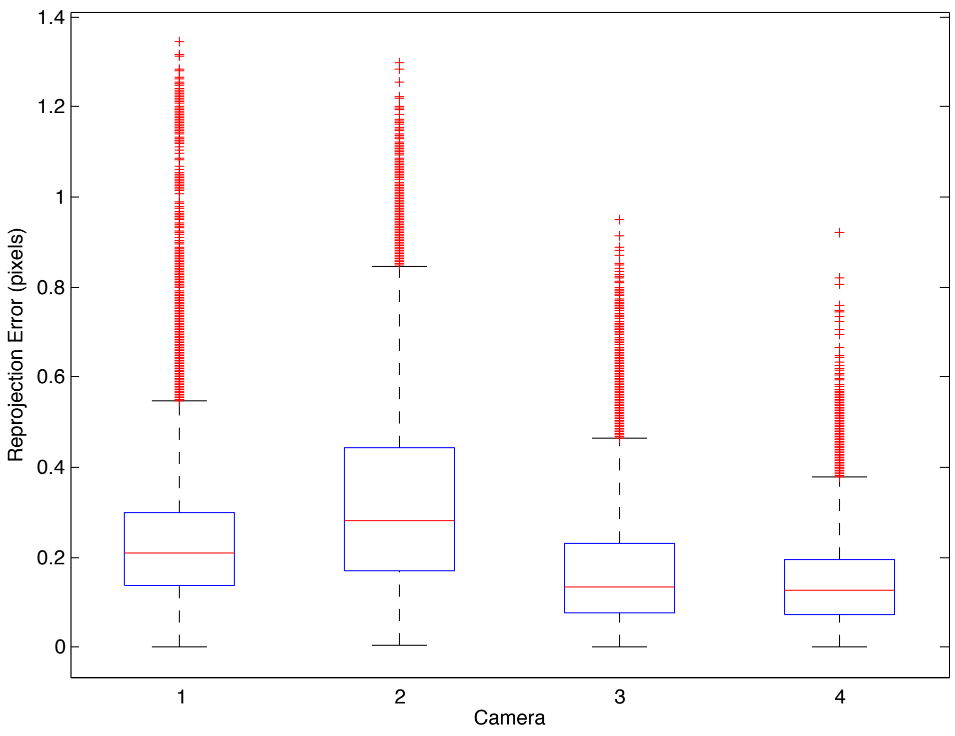


Figure 6.1: Reprojection error barplot for set 23.

6.2 Ground truth

For ‘ground truth’ measurements, each keypoint was clicked in each frame of the final wingbeat of the hovering data sets, noting their image locations and the 3-d locations of their intersections. Only those points that were clearly visible in a given view were clicked. As there were frames for which certain keypoints were visible in less than two views, these keypoints could not be used to make a 3-d measurement in these frames. Figure 6.2 shows these gaps in the data for the clicked points of set 23L. In this set only keypoint 5 was assigned a 3-d point in all frames.

The identification of the 2-d image points introduces a new source of error, as the features cannot be clicked with perfect accuracy. As the features are defined by vein intersection points, the widths of the intersecting veins make this location subjective. This is further complicated by the image resolution, as subpixel localisation requires judgement about how to interpolate between discreet intensities. Another factor is that differences in the way the vein intersections project into the images - both due to projective distortion and the fact that vein thickness will cause them to protrude slightly from the surface - mean that slightly different points might be selected depending on viewpoint. All of these factors, combined with the error inherent in the calibration itself, mean that the term ‘ground truth’ applies only very loosely to the clicked points.

In order to assign a value to this error, it was assumed (based on the experience of

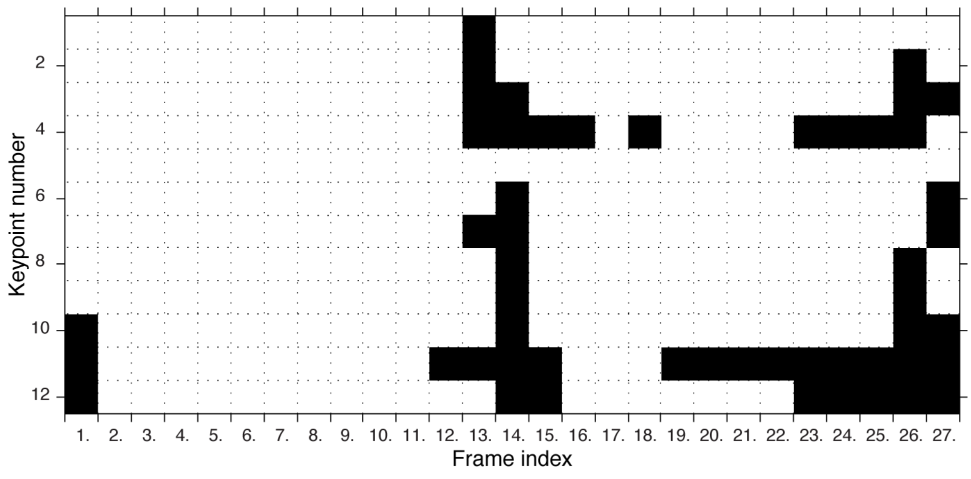


Figure 6.2: Reprojection error barplot for set 23.

clicking the points) that clicked measurements had Gaussian noise, with mean error 0 and a standard deviation of 0.5. This would mean that 99% of the points identified are within 1.3 pixels of the 'true' point, and 75% within 0.3 pixels. (Roughly 1 in 300 would be separated by 2 pixels.)

This assumption allows the estimation of 3-d error that can be attributed to imperfect clicking. By sampling from this distribution and adding the values as displacements to the clicked points, the effect of this uncertainty was approximated. In this way, 1000 alternative locations were created for each clicked image point that were then used to generate 3-d locations. This is illustrated in Figure 6.3, which shows a cloud of 3-d locations obtained for an individual keypoint.

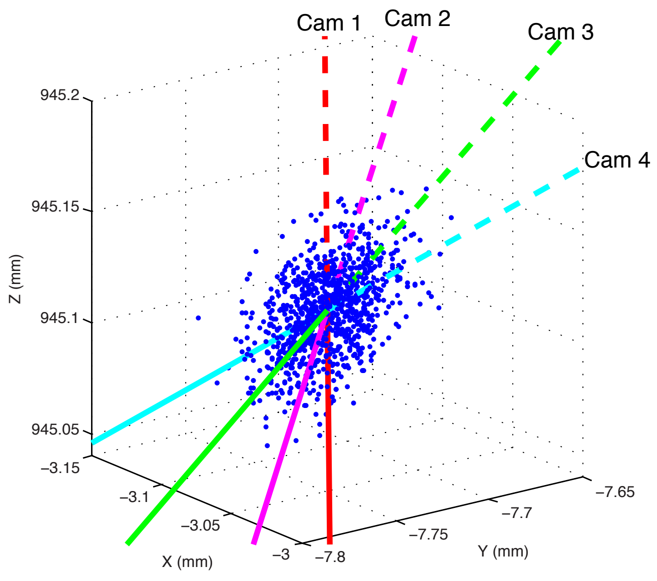


Figure 6.3: Cloud of possible 3-d keypoint locations, based on samples from approximated clicking error.

The distance between each of these 3-d points and their corresponding point from the original clicked locations were calculated for a keypoint in every frame in the final wing-beat. Figure 6.4 shows these separations for keypoint 1 for set 23R. The maximum observed error over the 27000 samples (1000 samples per frame) was 0.16mm. This means that the subjectivity in keypoint image locations contributes up to 0.16mm of uncertainty in the 3-d measurement.

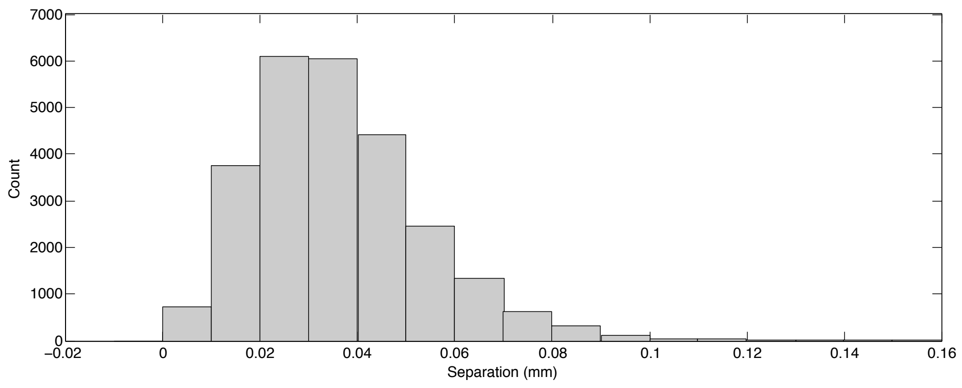


Figure 6.4: Histogram of 3-d localisation differences induced by clicking errors for keypoint 1. (Set 23R)

6.3 Algorithm accuracy

As mentioned above, the overall accuracy of the algorithm was measured by comparing the final keypoint location estimates to clicked keypoint locations. This comparison is therefore affected by a combination of the clicking error and calibration error, which can mask the effects of inaccuracy in measurement caused by the algorithm itself. Both the error in two dimensions (in pixels) and in three dimensions (in millimetres) are considered.

The two dimensional error was calculated by projecting the final keypoints, extracted from the TPS surface, back into the original images, and measuring their distance from the clicked locations (where available) in pixels. Figure 6.5 shows these 'reprojection errors' split by camera for Set 23L.

Some observations:

- The furthest outlier is over 25 pixels from the clicked point in camera 1, but the corresponding separation in camera 3 and 4 is less than 15 pixels (the most distant outlier in these views). This demonstrates that pixel error is highly dependent on viewpoint. (The camera 1 outlier could of course have a 0 pixel reprojection error in a view for which the clicked point (in the 3-d camera plane), camera centre, algorithm 3-d point and 'true' 3-d point are colinear.)
- The median pixel separation is approximately 1.7 pixels in each view. So 50% of the algorithm keypoint projections are within 1.7 pixels of the clicked points (which is within the range of clicking errors).
- 25% are less than 1 pixel from the clicked points, and 75% of the separations are

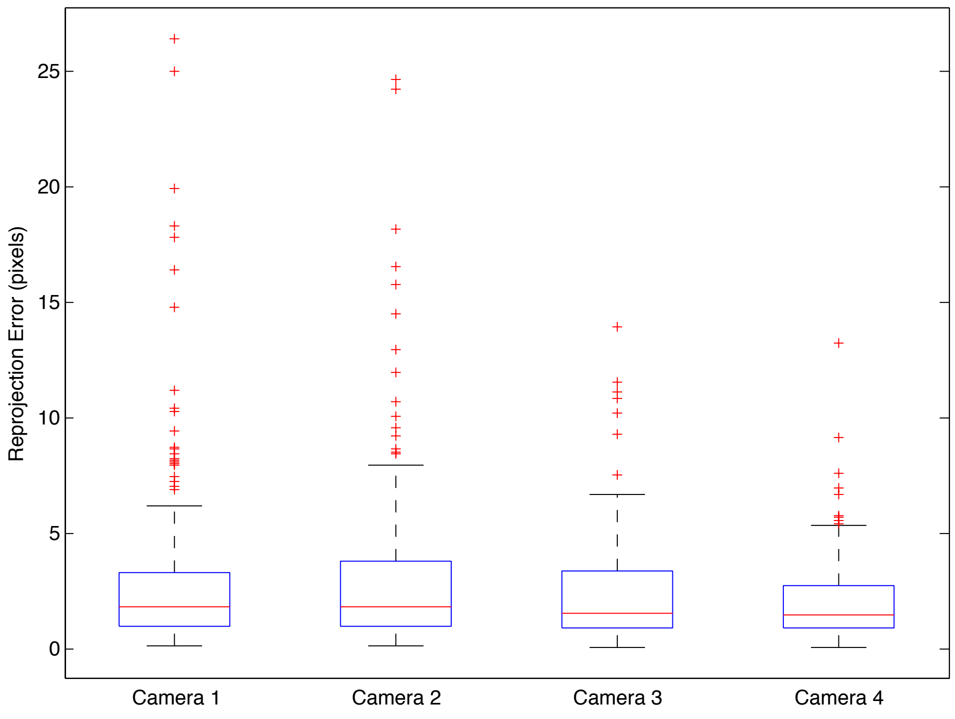


Figure 6.5: Pixel separations between clicked points and projected algorithm points (Set 23L)

within 3.8 pixels.

Three dimensional errors are shown in Figure 6.6, which shows boxplots of millimetre separations for each of the hovering sets, and one for all sets combined.

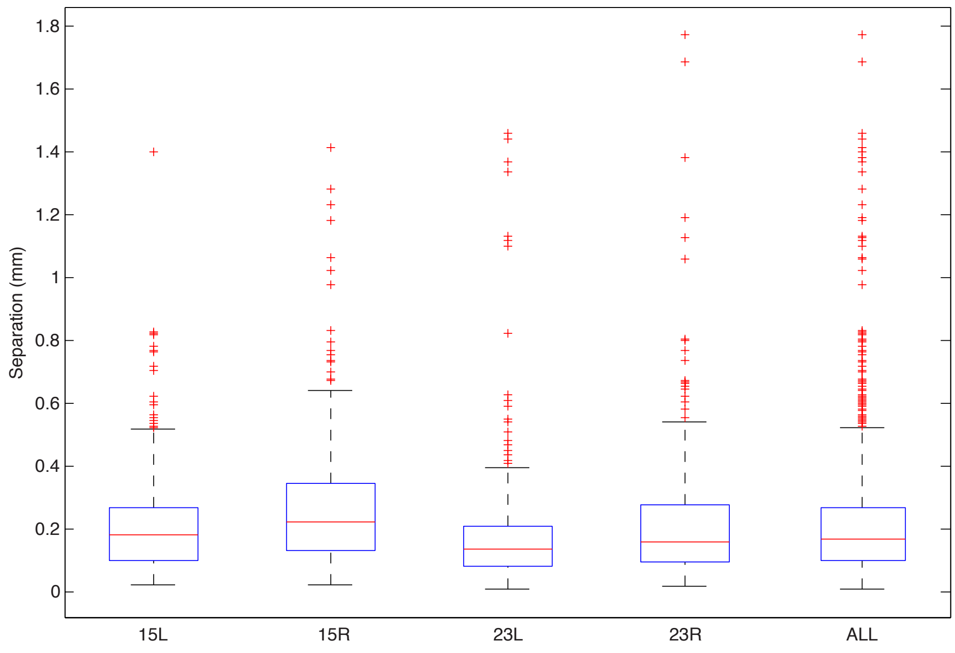


Figure 6.6: Reprojection error barplot for sets 15 and 23.

From the combined set of 3-d separations we observe that:

- The furthest outlier is 1.8mm from the clicked point, equivalent to roughly one fifth of the length of the wing.
- The median separation is 0.17mm. As the uncertainty in clicking leads to up to 0.16mm of 3-d localisation error, this means that about 50% of the data points are found to within an error equivalent to clicking uncertainty.
- 25% of the keypoints are separated by less than 0.01mm from the clicked 3-d points, and 75% are within 0.27mm.

Process accuracy

The separate processes within the algorithm can also be judged based on their ability to estimate keypoint locations, revealing the relative contributions of each process to the overall accuracy. Six algorithm processes were evaluated: prediction, tuning, initial smoothing, wing fitting, second smoothing and then the final result. As the prediction process only generates 3-d locations for 5 keypoints, a TPS surface was fitted to these points (with smoothing parameter 0.8) to obtain locations for additional keypoints.

The 3-d separation between the 12 keypoints and their corresponding clicked locations in each frame of the final wingbeat for each of the hovering sets (15L, 15R, 23L and 23R) was calculated. Figure 6.7 shows boxplots for the combined results.

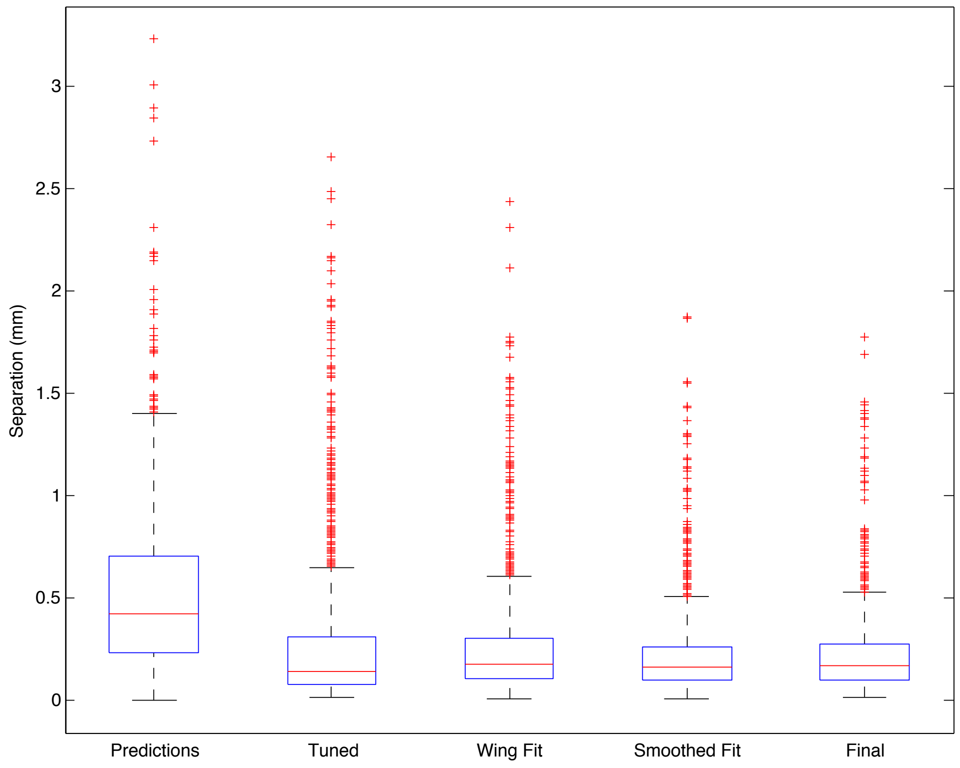


Figure 6.7: Error across all processes.

The figure shows that the separation for the largest outlier drops with each consecutive process, so in this sense, the accuracy is improved at each stage. The median error, however, is lowest for the tuning process. This illuminates a key property of the algorithm, which is that at its core it is a stereo image registration process that produces very accurate keypoint estimates under certain conditions and poor estimates in others (with relative frequency indicated by the boxplot). The later stages are then attempts to identify and remove the poor estimates or smooth over all estimates in a way that increases the average accuracy while decreasing the accuracy of the well-tuned points.

Chapter 7

Conclusions

7.1 Summary

This thesis introduces an algorithm for the automated reconstruction of 3-d wing surfaces over an arbitrary number of wingbeats of hovering flight, requiring limited manual initialisation. Its development satisfied each the objectives listed in Chapter 1, repeated in Figure 7.1.

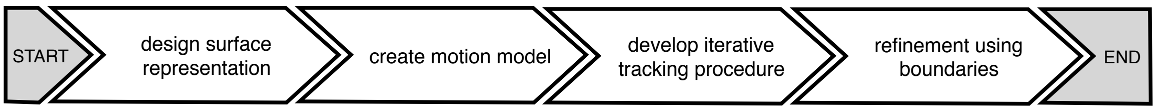

START
design surface representation
create motion model
develop iterative tracking procedure
refinement using boundaries
END

Figure 7.1: Recap of tracking algorithm objectives.

To represent the shape of the wing surface (objective 1), thin-plate spline maps were used for each frame. This representation allows a continuous, parameterised description of the wing as a 2-d manifold, implicitly defined using a number of tracked 3-d keypoints and a reference image. In this continuous form, an arbitrary number of points can be sampled from the wing, while the use of splines ensures that any deformation shape can be captured.

To predict the locations of the keypoints in each new frame (objective 2), a novel motion model was developed, called the 'normalised stroke model'. This model utilises the tendency for surface points to behave as if moving on the surface of the sphere and the similarity of consecutive strokes of the same type (i.e. upstrokes and downstrokes). It extracts patterns in spherical coordinates from a sparse set (5 points) of recently measured image locations and uses them to interpolate a suitable stroke shape between user-supplied locations at the start and end of the stroke.

The predicted points are then fed into the 'tuning' process (of objective 3), which refines their locations in a stereo image registration framework, and can generate 3-d estimates for all points on the wing surface. The later stages identify outliers from this process, smooth over the results, and fit a wing shape to the results using the assumption of wing extensibility.

Finally a set of wing boundary points, reconstructed in a fully automated process, are used to improve the surface (completing objective 4). This process uses the space curve matching technique developed by Schmid and Zisserman [33] in combination with a novel strategy to incorporate the boundaries into the wing shape estimates.

This algorithm is unique among purely image-based flapping flight studies in the level of automation it achieves. Wing measurements would ordinarily require over 1000 manually identified keypoint locations per wingbeat (the number used here to create a 'ground truth' estimate for a single wing during a wingbeat). This is the case for the recent study by [86] which uses the same data, but relies entirely on smoothing over manually identified keypoints and discrete boundary locations. In contrast, this algorithm requires about 40 of these localisations per wingbeat (5 points for each stroke in 4 views), after an initialisation requiring 16 points to determine a moving wing coordinate frame (4 clicks each for the head and 'tail' for the forwards direction, and 4 clicks each for the start and end shoulder position) and wing bounding boxes for each frame (which do not need to stick tightly to the wing).

The algorithm was tested on two hovering data sets with two wings in each set, creating a total of 384 surface estimates. It was also tested on a challenging manoeuvre set, producing a further 118 surface estimates.

The algorithm can be evaluated qualitatively, by projecting the reconstructed wing surfaces into the original views, or quantitatively, by comparison with manually identified points. Complete sets of image projections are included in the data DVD accompanying this thesis. They show that the algorithm performs very well in those frames where there are multiple views with high wing area, and that the smoothing processes are reasonably successful in interpolating the wing shapes in low visibility frames (where manually identifying keypoints may be difficult or impossible). As a consequence of the camera locations, coupled with the fact the wing is at its most cambered, the extrema frames have almost uniformly low visibility, so the algorithm benefits from manual point selection in these frames. The area of the wing near the shoulder was most problematic for the algorithm, largely due to the high level of occlusion in this area, so the projections here are least reliable.

Tests on the manoeuvre set showed the importance of estimating a good moving coordinate frame: if it is not estimated with suitable accuracy, the prediction model can fail in a way that prevents successful tuning, and leads to an accumulation of errors in subsequent strokes. But the manoeuvre set also demonstrated the ability for the algorithm to maintain successful wing reconstruction in half of the wingbeat - the downstrokes - even while the other half fails. The accuracy of the downstroke reconstructions in this challenging data set shows that the algorithm can handle complex motions.

Quantitative evaluation of the algorithm showed that it achieves sub-millimetre accuracy. The median separation between algorithm points and manually identified points for the last wingbeat of the hovering sets was 0.17mm - very close to the range of errors attributable to manual identification (up to 0.14mm). 75% of the results were less than 0.6mm, and 100% less than 1.8mm. The wing itself is 10mm in length. In terms of pixel error, 75% are within 3.8 pixels of manually clicked locations. For comparison, the calibration can produce a reprojection error of up to 1.4 pixels and 'clicking error' can add the same amount again.

The algorithm has been tailored to the case of hoverfly flight but it may have potential for more general applications. To make use of the motion model, a data set must involve motion that is approximately periodic and rotates around a common point. The tuning process relies not only on the assumption that the object to be tracked is textured (for registration), but it must also be a thin surface with an identical pattern on both sides to allow viewpoints on either side of the object to be registered to the same template. The wing fit procedure penalises stretching of the surface, but does not prohibit it, so the object should have limited extensibility. The boundary extraction process also depends on the object being very thin and uses the assumption that it has a curved rim. Combined, the broadness of these assumptions indicate that the algorithm could be used for any number of flapping flight scenarios.

7.2 Future Work

Analysis of the results and unexplored avenues from the literature suggest the following future extensions to this project.

Visual hull Figure 7.2 shows a visual hull for the silhouettes of a frame in Set 15.[1] The visual hull was implicitly used in the 'epipolar segment' distance test for outliers, but there was no explicit final test to verify that the reconstructed surfaces lie within the visual hull. There may be scope for improvement involving a search for the smallest deformation of the surface that would place it entirely within the visual hull. A triangulated mesh representation of the surface may make the investigation of candidate transformations easier, while allowing one to frame constraints such as inextensibility and limits on bending in terms of relationships between vertices, edges and faces.

A complete set of visual hulls could also be useful for obtaining a better estimate of rigid body motion, e.g. the motion of the centre of mass of the hull, and for identifying views that are likely to suffer most from occlusion by the body.

Vein tracing There may be potential to make more direct use of wing veins. Blood vessel

[1]This hull was produced by identifying a bounding volume for the fly, then testing points for silhouette consistency within an even grid of 10 points per millimetre (totalling 11.25M points). The outer shell of consistent points was extracted using Matlab's *bwperim* function, then the open-source software Meshlab was used to extract surface normals (via poisson disk sampling), and to reconstruct a triangulated surface (via poisson surface reconstruction).

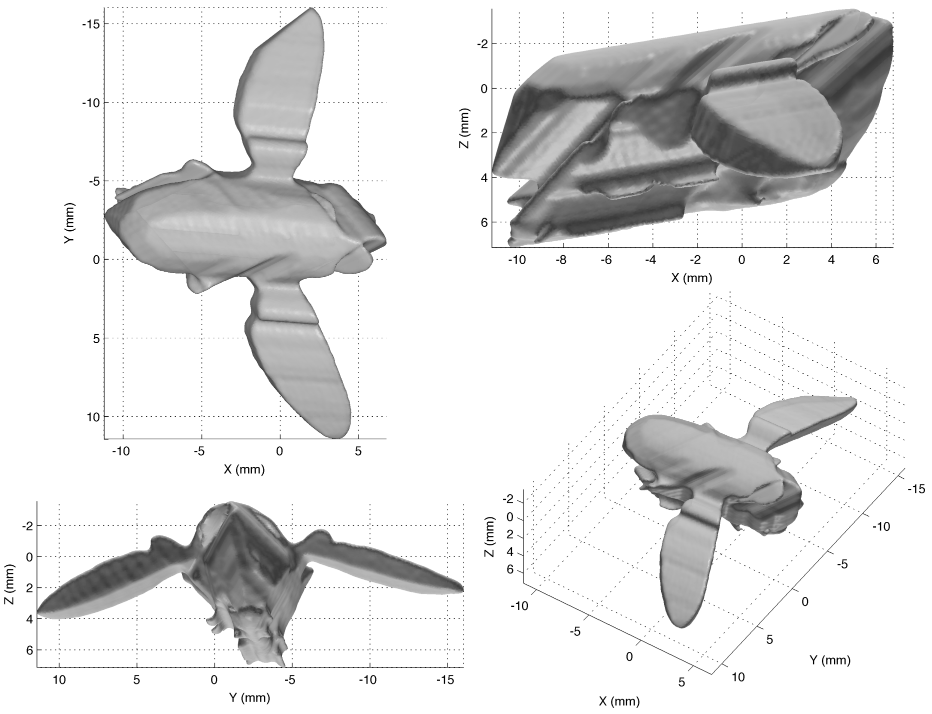


Figure 7.2: Visual hull. (Set 15 Frame 89)

tracing techniques from digital retinal image analysis (described in Section 1.2.1) could be used to extract a full set of wing veins. In an approach similar to that used in the object recognition algorithm by Mikolajczyk et al. [15] (discussed in Section 1.2.1), points along these veins could be sampled and matched to a similarly sampled wing template using robust point matching method. If successful, this could provide a large number of correspondences to assist in constructing the wing surface.

Temporal coherence for silhouette extraction The algorithm does not make use of the similarity between silhouettes in adjacent frames - each is extracted independently. A method such as Active Contours might aid the segmentation process, by searching for the fly boundary in the vicinity of previous frame's boundary. This might solve the problem of background clutter becoming attached to the silhouettes, or the more damaging (albeit rare) problem of low contrast wing regions failing to be segmented with the rest of the fly.

Between-wing symmetry The algorithm reconstructs left and right wings independently. As the wing motions will be close to mirror images of each other for hovering flight, there may be scope for bringing in this symmetry - a reconstruction for one wing could aid reconstruction in the other.

Characteristic motion patterns By running this algorithm on many different data sets and investigating patterns in the results, it may be possible to develop a physical model to represent typical wing surface motion using a small number of parameters. This could allow wing deformation to be more accurately predicted and remove the need for manual measurements at extrema.

Appendix A

Calibration

For the fixed camera setup used in this project, camera calibration required a set of 3-d points and their corresponding projections to be identified throughout the measurement volume. Corresponding image points were obtained with the aid of a precisely-machined planar 'calibration target' containing a regular grid of dots. Using a simple image processing 'blob detection' algorithm, the centres of the dots were identified in each image. The alignment of these dots, combined with the manual identification of an 'origin' dot, allowed correspondences between dots to be established. Figure A.1 shows an example frame from the calibration of Set 15. Each set used approximately 100 of these frames, with a different target position and orientation in each frame - the result of waving the target through the volume. The complete set of corresponding dot centres in all frames (6733 points for Set 15) were used as measurement points.

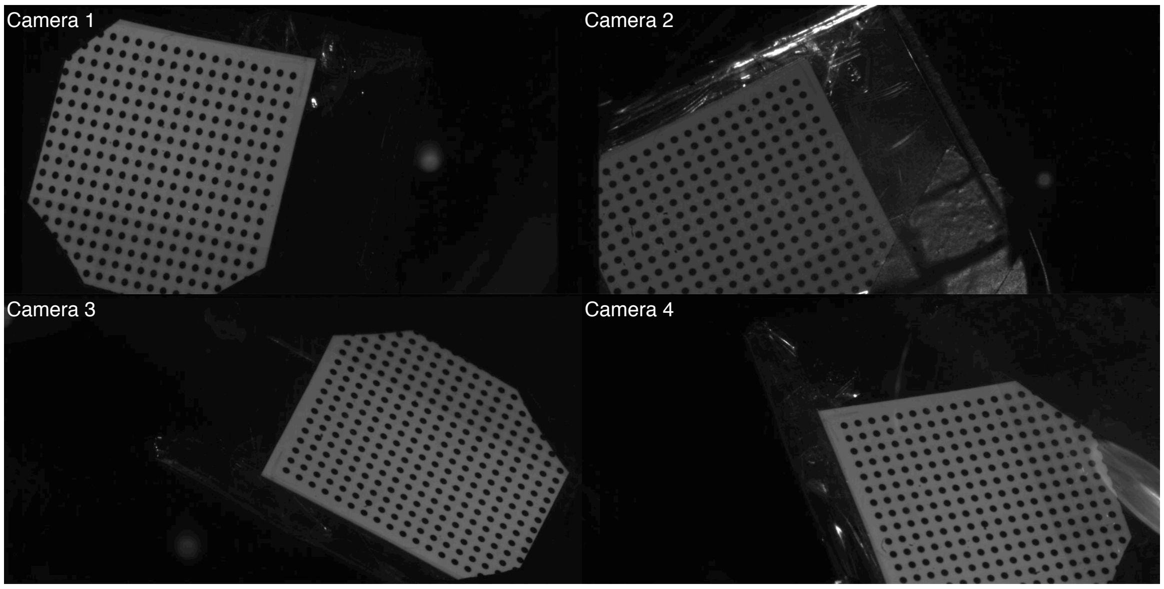


Figure A.1: Example calibration images from four cameras. Dot spacing is 1.5mm. (Set 15)

Next a least squares fit was performed on the measurement points to obtain estimates of intrinsic and extrinsic camera parameters. This fitting process minimises the 'reprojection errors' of measurement points (the euclidean distances between the 2-d initial measurements and the reprojections of 'intersection' points) by the Levenberg-Marquadt algorithm.

Intrinsic parameters fitted for included focal length and spherical lens distortion. Best results were obtained when principal point, image skew, and pixel aspect ratio were not included in the fit: the centre of each image was used as 'principal point'; the images were assumed to have zero skew; and pixels were assumed to be perfectly square. The extrinsic parameters were the camera centres and optical axis orientations. These were defined relative to Camera 1, whose centre was taken as global origin, and viewing axes taken as global X-Y-Z axes. As a final calibration step, a 'bundle adjustment' was performed to refine the parameter values. Example parameters (from Set 15) are shown in Table A.1.

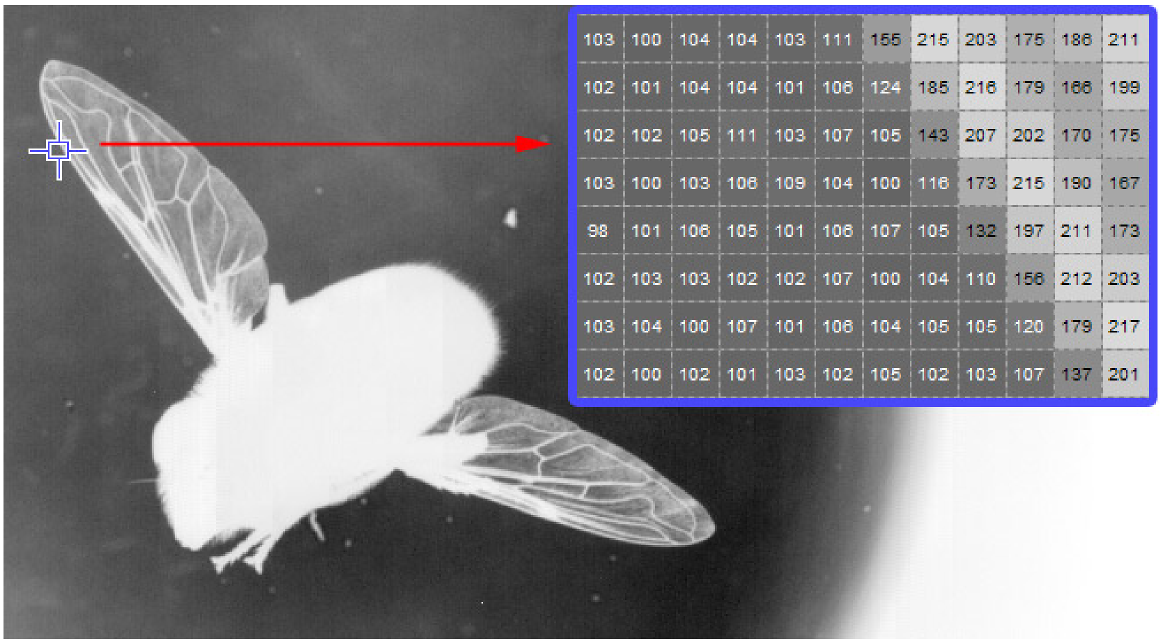
Variable	Camera 1	Camera 2	Camera 3	Camera 4
Focal length (mm)	16278.3	15791.8	16560.9	16497.2
Distortion (spherical)	1.2	0.9	0.6	1.1
Rotation (rad)	(0, 0, 0)	(-0.4, -0.6, -0.6)	(-0.2, -1.2, -2.8)	(-0.5, -1.1, -2.0)
Translation (mm)	(0, 0, 0)	(344.6, -483.4, 176.1)	(-71.3, -678.0, 326.0)	(85.2 -778.6 432.4)

Table A.1: Calibration parameters (rounded to 2 d.p.) for set 15. Camera rotations and translations are given relative to camera 1, used as global origin.

Appendix B

Background Subtraction

In ideal hoverfly tracking data, every pixel in the background would have the same intensity (or in the case of colour images, the same RGB values). In the case of dark objects of interest (here called ‘foreground’) against a bright background, this would make the identification of background pixels a matter of simple thresholding: all pixels of a certain brightness could be separated as background. In the case of the actual hoverfly data, the situation is more complicated: pixel values vary across the background (as illustrated in Fig. B.1) due to non-uniformity of the laser light reaching the cameras and the presence of background features such as dust. As the dust particles are imaged as dark blobs, these would be classified as foreground by a thresholding segmentation procedure. Conversely the pixels on wing membranes are often very bright, so these may be incorrectly classified as background.



103	100	104	104	103	111	155	215	203	175	186	211
102	101	104	104	101	106	124	185	216	179	166	199
102	102	105	111	103	107	105	143	207	202	170	175
103	100	103	106	109	104	100	116	173	215	190	167
98	101	106	105	101	106	107	105	132	197	211	173
102	103	103	102	102	107	100	104	110	156	212	203
103	104	100	107	101	106	104	105	105	120	179	217
102	100	102	101	103	102	105	102	103	107	137	201

Figure B.1: Intensity values in a pixel window around a wing edge. (Inverted image)

An issue for segmentation is that the background becomes very dark towards the edges of each image. In those frames where the wing strays into the dark regions there is very little difference in pixel intensities between foreground and background. This is illustrated in Figure B.2, which shows (inverted) pixel intensities as depth in a surface plot. The wing can be identified through the high ridges of its veins, but this is more difficult as the intensity differences drop towards the edge of the image. Successful segmentation by thresholding is equivalent to finding an intensity 'depth' which is above background pixels but below wing pixels.

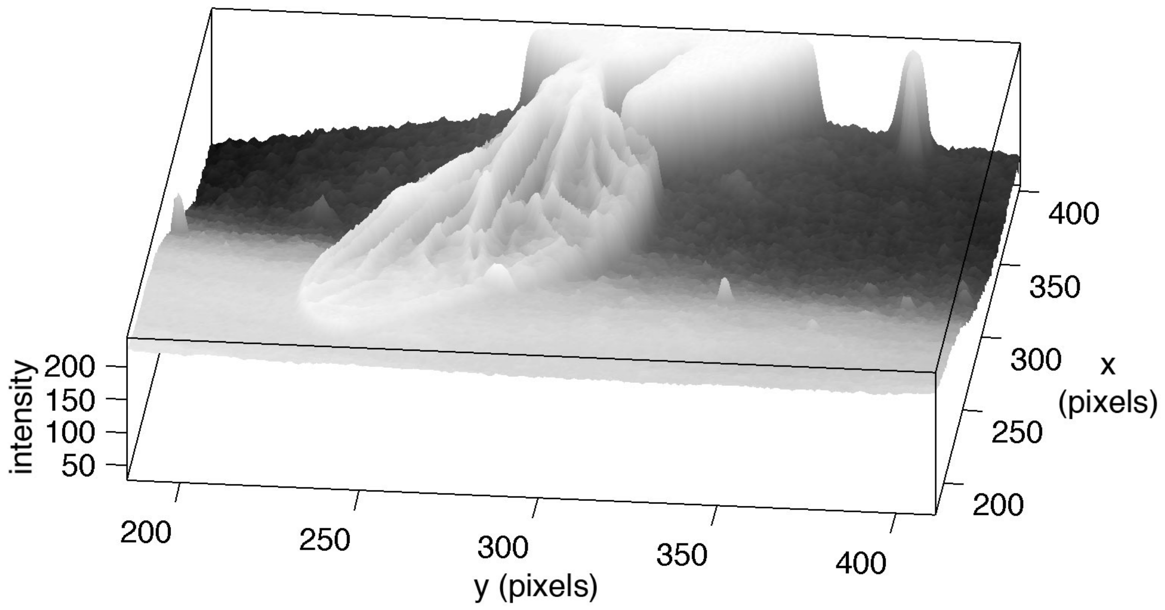


Figure B.2: (Inverted) intensity values as surface plot, showing wing veins as ridges elevated above background. (Frame 23, camera 2, right wing)

The process of removing background intensity patterns (such as the dark dust particles) to create a more uniform background intensity is called 'background subtraction'. As a first step, the illumination differences in the images, resulting from the pulse-to-pulse intensity variation, were quantified. This was accomplished by identifying pixel windows which belong to the background in every image of the sequence. The minimum intensity value achieved by every pixel in the image sequence was calculated, creating a 'minimum' image. Dark pixels in this image correspond to locations that were occupied by the fly at some point during the sequence (or to dark background features). Bright regions were manually selected for use as pixel neighbourhoods, as shown in Figure B.3.

The background intensity for each image was quantified as the mean pixel value for this neighbourhood. A graph of the mean pixel values for the full image sequence in all views is shown in Figure B.4.

A 'maximum' image was also constructed by calculating maximum pixel values through the image sequence. This approximates an image of the background without the fly present, due to the drift of the fly body during the image sequence. (Had an image been taken from each camera before the fly was put in the apparatus, this would not have been necessary.)

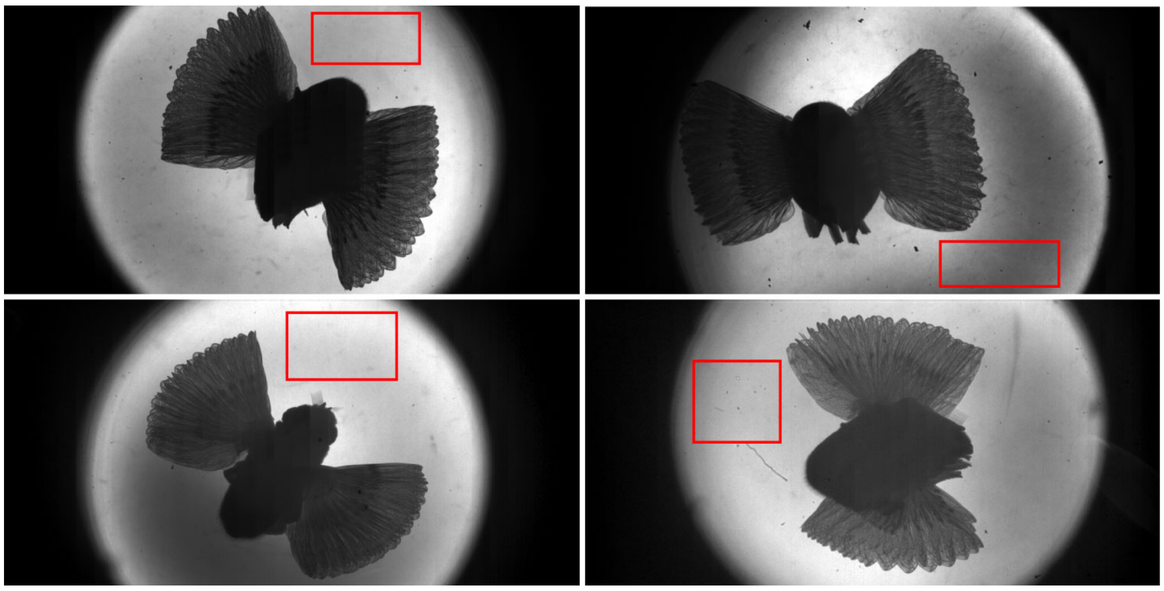


Figure B.3: ‘Minimum’ images, showing manually selected background pixels.

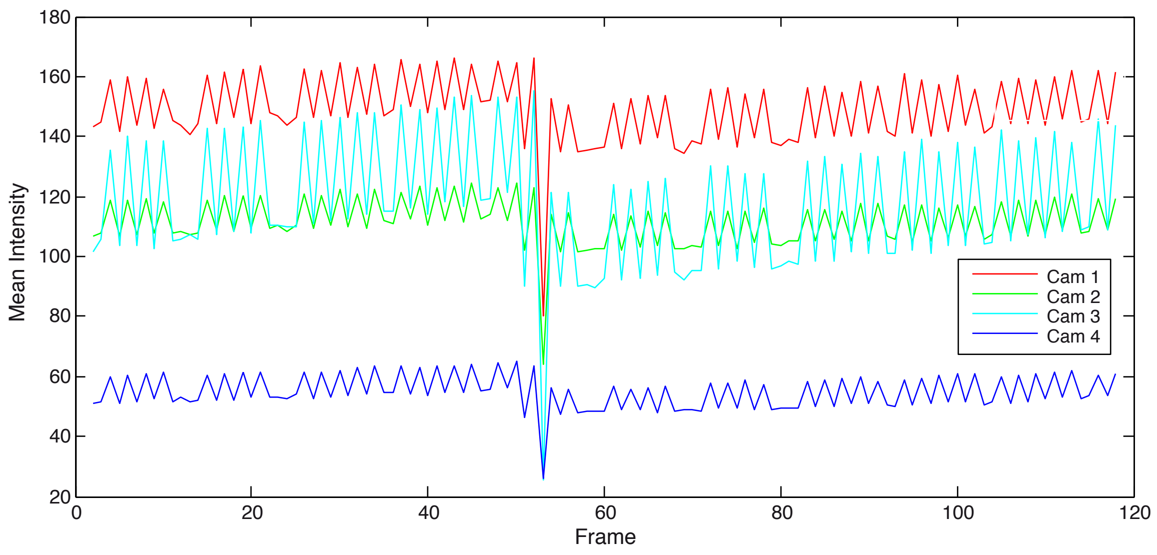


Figure B.4: Background intensity variation over frames of 4 cameras.

In the regions of the image which the fly never fully vacates, the pixels remain dark. These regions are larger in some views than others. Figure B.5 shows these 'maximum' images.

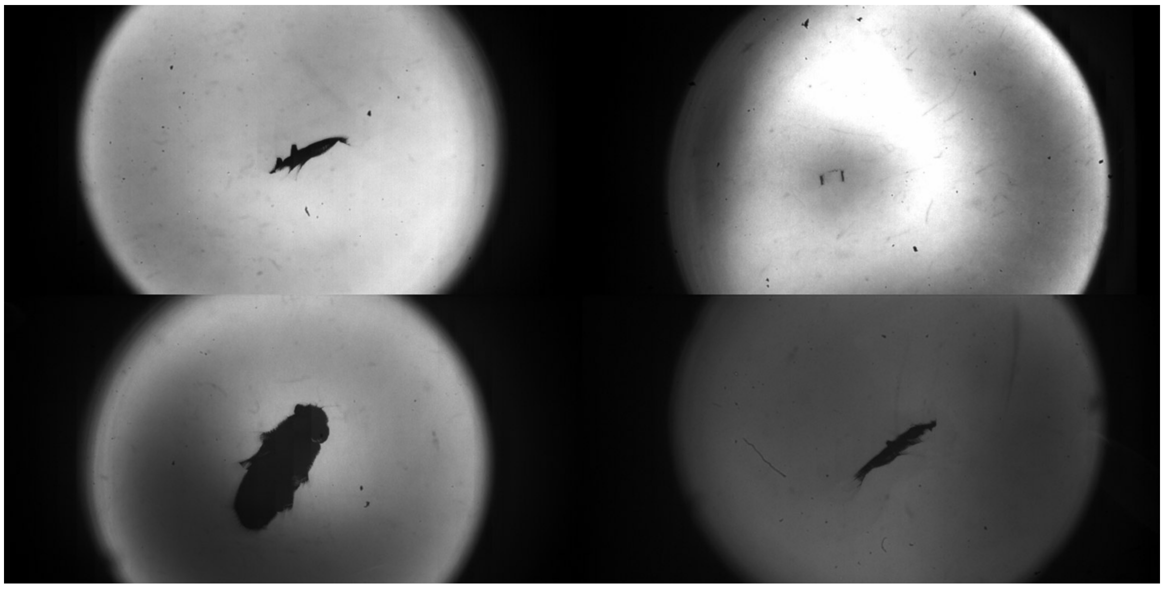


Figure B.5: 'Maximum' images from maximum intensity value at each pixel over image sequence.

Background subtraction then involved subtracting a scaled version of the current image from the maximum image. The scaling factor used was the ratio of the mean brightness of the background window current image to the mean brightness of the same window in the maximum image. This results in a new image with an almost uniformly dark background, with most fly pixels standing out as bright regions. Inverting this image gives a reasonable approximation of a 'background subtracted' image, as shown in Figure B.6.

The presence of the 'holes' cut by through the fly by the dark 'fly' pixels in the maximum image limit the usefulness of this background subtracted image. When these images were used to extract silhouettes, using the method in Section 2.2, less background clutter appeared in the edge images, but for a number of frames (particularly for camera 3) they broke up edges on the fly boundary. A solution might have been to fill the holes by applying a manually-selected mask of dark pixels, but this would produce an additional loss of automation.

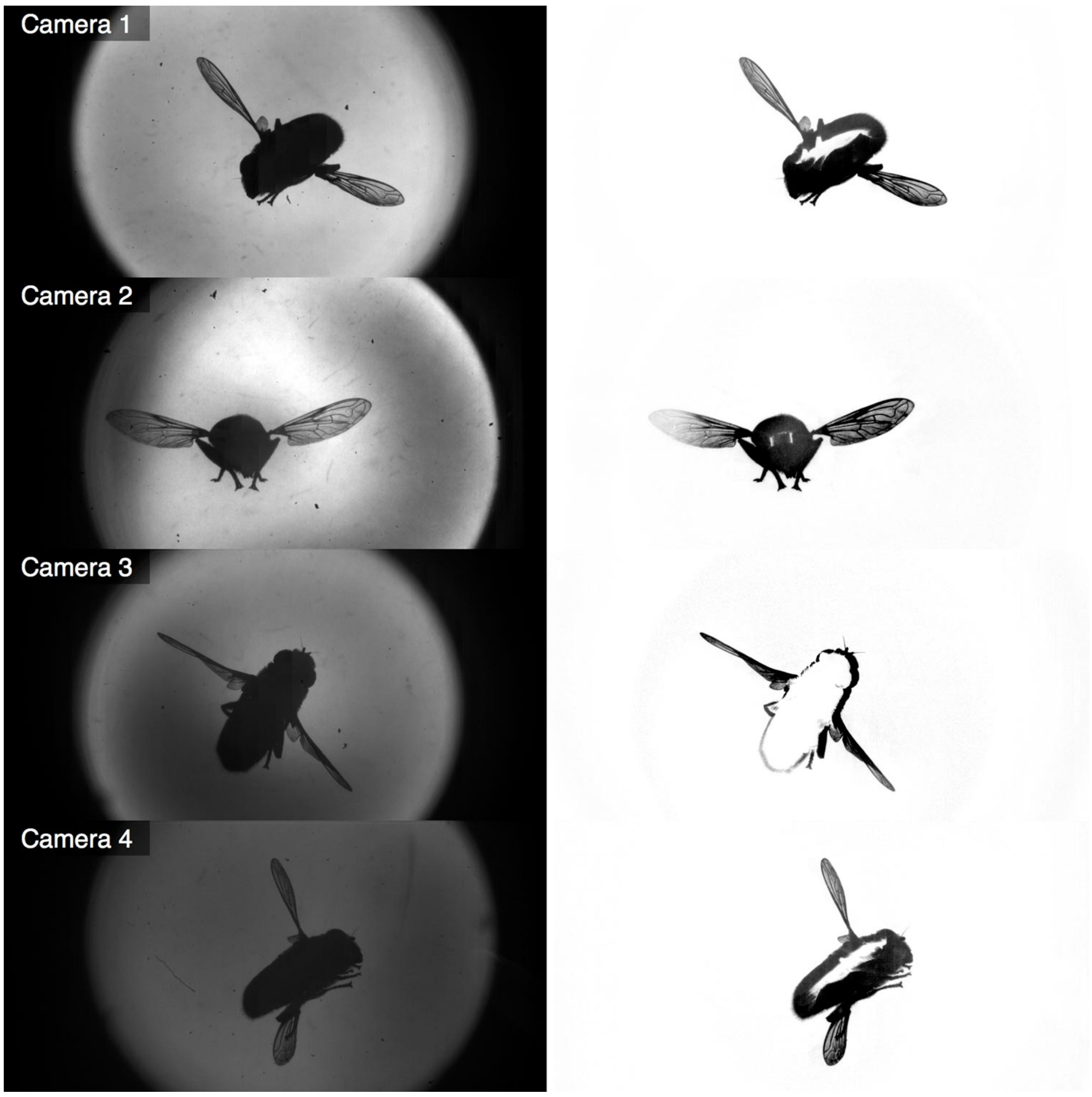


Figure B.6: Example results from scaled subtraction of ‘maximum’ image. (Frame 22)

References

[1] I. D. Wallace, N. J. Lawson, A. R. Harvey, J. D. C. Jones, and A. J. Moore. High-speed photogrammetry system for measuring the kinematics of insect wings. *Applied optics*, 45(17):4165–73, 2006. ISSN 0003-6935. URL http://www.ncbi.nlm.nih.gov/pubmed/16761060.

[2] S. N. Fry, R. Sayaman, and M. H. Dickinson. The aerodynamics of hovering flight in Drosophila. *Journal of Experimental Biology*, 208(12):2303–2318, 2005. doi: 10.1242/jeb.01612.

[3] C. P. Ellington. The Aerodynamics of Hovering Insect Flight. III. Kinematics. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 305(1122): 41–78, Feb. 1984. ISSN 0962-8436. doi: 10.1098/rstb.1984.0051. URL http://rstb.royalsocietypublishing.org/cgi/doi/10.1098/rstb.1984.0051.

[4] H. Wang. Measuring wing kinematics, flight trajectory and body attitude during forward flight and turning maneuvers in dragonflies. *Journal of Experimental Biology*, 206(4):745–757, Feb. 2003. ISSN 00220949. doi: 10.1242/jeb.00183. URL http://jeb.biologists.org/cgi/doi/10.1242/jeb.00183.

[5] R. J. Wootton. Functional morphology of insect wings. *Annual review of entomology*, 37(1):113–40, 1992.

[6] A. R. Ennos. The importance of torsion in the design of insect wings. *The Journal of Experimental Biology*, 140(1):137–160, Nov. 1988. URL http://jeb.biologists.org/cgi/content/abstract/140/1/137.

[7] R. J. Wootton. Support and deformability in insect wings. *Journal of Zoology*, 193 (4):447–468, Aug. 1981. ISSN 09528369. doi: 10.1111/j.1469-7998.1981.tb01497.x. URL http://doi.wiley.com/10.1111/j.1469-7998.1981.tb01497.x.

[8] C. P. Ellington. The novel aerodynamics of insect flight: applications to micro-air vehicles. *Journal of Experimental Biology*, 202(23):3439–3448, 1999.

[9] T. Weis-fogh. Quick estimates of flight fitness in hovering animals, including novel mechanisms for lift production. *Journal of Experimental Biology*, 59(1):169–230, 1973.

[10] R. B. Srygley and A. L. R. Thomas. Unconventional lift-generating mechanisms in free-flying butterflies. *Nature*, 420(6916):660–664, Dec. 2002. ISSN 0028-0836. doi: 10.1038/nature01223. URL http://www.nature.com/doifinder/10.1038/nature01223.

[11] V. Ferrari, T. Tuytelaars, and L. Van Gool. Simultaneous Object Recognition and Segmentation from Single or Multiple Model Views. *International Journal of Computer Vision*, 67(2):159–188, Jan. 2006. ISSN 0920-5691. doi: 10.1007/s11263-005-3964-7. URL http://www.springerlink.com/index/10.1007/s11263-005-3964-7.

[12] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A Comparison of Affine Region Detectors. *International Journal of Computer Vision*, 65(1-2):43–72, Oct. 2005. ISSN 0920-5691. doi: 10.1007/s11263-005-3848-x. URL http://www.springerlink.com/index/10.1007/s11263-005-3848-x.

[13] K. Mikolajczyk and C. Schmid. Scale & Affine Invariant Interest Point Detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.

[14] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, editors, *Computer Vision - ECCV 2002*, volume 2350 of *Lecture Notes In Computer Science*, pages 128–142. Springer Berlin / Heidelberg, 2002. doi: 10.1007/3-540-47969-4_9.

[15] K. Mikolajczyk, A. Zisserman, and C. Schmid. Shape recognition with edge-based features. *Proceedings of the British Machine Vision Conference*, pages 79.1–79.10, 2003. doi: 10.5244/C.17.79. URL http://www.bmva.org/bmvc/2003/papers/paper-155.html.

[16] T. Tuytelaars and L. Van Gool. Matching Widely Separated Views Based on Affine Invariant Regions. *International Journal of Computer Vision*, 59(1):61–85, 2004.

[17] G. Carneiro and A. D. Jepson. Pruning local feature correspondences using shape context. In *Proceedings of the International Conference on Pattern Recognition*, volume 3, pages 16–19. IEEE, 2004. ISBN 0-7695-2128-2. doi: 10.1109/ICPR.2004.1334458. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1334458.

[18] H. Chui and A. Rangarajan. A new point matching algorithm for non-rigid registration. *Computer Vision and Image Understanding*, 89(2-3):114–141, Feb. 2003. ISSN 10773142. doi: 10.1016/S1077-3142(03)00009-2. URL http://linkinghub.elsevier.com/retrieve/pii/S1077314203000092.

[19] V. Ferrari, F. Jurie, and C. Schmid. Accurate Object Detection with Deformable Shape Models Learnt from Images. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, June 2007. ISBN 1-4244-1179-3. doi: 10.1109/CVPR.2007.383043. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4270068.

[20] T. F. Cootes, C. J. Taylor, D. H. Cooper, J. Graham, and A. Lanitis. Active Shape Models-Their Training and Application. *Computer Vision and Image Understanding*, 61(1):38–59, 1995.

[21] C. Graetzel, S. N. Fry, and B. Nelson. A 6000 Hz Computer Vision System for Real-Time Wing Beat Analysis of Drosophila. *The First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 278–283, 2006. doi: 10.1109/BIOROB.2006.1639099. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1639099.

[22] D. Houle, J. Mezey, P. Galpern, and A. Carter. Automated measurement of Drosophila wings. *BMC Evolutionary Biology*, 3(1):1–13, 2003. doi: 10.1186/1471-2148-3-25. URL http://www.biomedcentral.com/1471-2148/3/25.

[23] H. F. Jelinek, C. Depardieu, C. Lucas, D. J. Cornforth, W. Huang, and M. J. Cree. Towards Vessel Characterisation in the Vicinity of the Optic Disc in Digital Retinal Images. In McCane, editor, *Proceedings of the Image and Vision Computing Conference*, pages 2–7, 2005.

[24] A. Hoover, V. Kouznetsova, and M. Goldbaum. Locating Blood Vessels in Retinal Images by Piecewise Threshold Probing of a Matched Filter Response. *IEEE Transactions on Medical Imaging*, 19(3):203–210, 2000.

[25] M. Foracchia, E. Grisan, and A. Ruggeri. Detection of Optic Disc in Retinal Images by Means of a Geometrical Model of Vessel Structure. *IEEE Transactions on Medical Imaging*, 23(10):1189–1195, Oct. 2004. ISSN 0278-0062. doi: 10.1109/TMI.2004.829331. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1339426.

[26] S. Chaudhuri, S. Chatterjee, N. Katz, M. Nelson, and M. Goldbaum. Detection of blood vessels in retinal images using two-dimensional matched filters. *IEEE*

transactions on medical imaging, 8(3):263–9, Jan. 1989. ISSN 0278-0062. doi: 10.1109/42.34715. URL http://www.ncbi.nlm.nih.gov/pubmed/18230524.

[27] M. J. Cree, D. J. Cornforth, and H. F. Jelinek. Vessel Segmentation and Tracking Using a Two-Dimensional Model. *IVC New Zealand*, pages 345–350, 2005.

[28] J. Houssineau, S. Ivekovic, and D. E. Clark. Disparity Space: A Parameterisation for Bayesian Triangulation from Multiple Cameras. In *15th International Conference on Information Fusion (FUSION)*, number 2, Singapore, 2012.

[29] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition, 2004. URL http://www.worldcat.org/title/multiple-view-geometry-in-computer-vision/oclc/52783638.

[30] U. Dhond and J. Aggarwal. Structure from stereo-a review. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(6):1489–1510, 1989. ISSN 00189472. doi: 10.1109/21.44067. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=44067.

[31] H. H. Baker and T. O. Binford. Depth from Edge and Intensity Based Stereo. In *Proceedings of the 7th international joint conference on Artificial intelligence*, volume 2, pages 631–636, San Francisco, 1981. Morgan Kaufmann Publishers Inc. URL http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA126745.

[32] Y. Ohta and T. Kanade. Stereo by Intra- and Inter-Scanline Search Using Dynamic Programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(2): 139–154, Mar. 1985. ISSN 0162-8828. doi: 10.1109/TPAMI.1985.4767639. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4767639.

[33] C. Schmid and A. Zisserman. The Geometry and Matching of Lines and Curves Over Multiple Views. *International Journal of Computer Vision*, 40(3):199–233, 2000. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.7.8857.

[34] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 519–528. Ieee, 2006. ISBN 0-7695-2597-0. doi: 10.1109/CVPR.2006.19. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1640800.

[35] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. Multi-view stereo evaluation web page. URL http://vision.middlebury.edu/mview/.

[36] G. Vogiatzis, C. Hernández Esteban, P. H. S. Torr, and R. Cipolla. Multiview stereo via volumetric Graph-Cuts and occlusion robust photo-consistency. *IEEE transactions on pattern analysis and machine intelligence*, 29(12):2241–6, Dec. 2007. ISSN 0162-8828. doi: 10.1109/TPAMI.2007.70712. URL http://www.ncbi.nlm.nih.gov/pubmed/17934232.

[37] O. Faugeras and R. Keriven. Variational principles, surface evolution, PDE's, level set methods, and the stereo problem. *IEEE Transactions on Image Processing*, 7(3):336–44, Jan. 1998. ISSN 1057-7149. doi: 10.1109/83.661183. URL http://www.ncbi.nlm.nih.gov/pubmed/18276253.

[38] J.-P. Pons, R. Keriven, and O. Faugeras. Modelling Dynamic Scenes by Registering Multi-View Image Sequences. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 822–827. IEEE, 2005. ISBN 0-7695-2372-2. doi: 10.1109/CVPR.2005.227. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1467528.

[39] C. Hernández Esteban and F. Schmitt. Silhouette and stereo fusion for 3D object modeling. *Computer Vision and Image Understanding*, 96(3):367–392, Dec. 2004. ISSN 10773142. doi: 10.1016/j.cviu.2004.03.016. URL http://linkinghub.elsevier.com/retrieve/pii/S1077314204000542.

[40] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence*, 32(8):1362–76, Aug. 2010. ISSN 1939-3539. doi: 10.1109/TPAMI.2009.161. URL http://www.ncbi.nlm.nih.gov/pubmed/20558871.

[41] C. R. Dyer. Volumetric Scene Reconstruction from Multiple Views. In L. S. Davis, editor, *Foundations of Image Understanding*, number 1, pages 469–489. Springer US, Boston, 2001. ISBN 978-1-4615-1529-6. doi: 10.1007/978-1-4615-1529-6_16. URL http://dx.doi.org/10.1007/978-1-4615-1529-6_16.

[42] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, 1994. ISSN 01628828. doi: 10.1109/34.273735. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=273735.

[43] S. Lazebnik, Y. Furukawa, and J. Ponce. Projective Visual Hulls. *International Journal of Computer Vision*, 74(2):137–165, Dec. 2007. ISSN 0920-5691. doi: 10.1007/s11263-006-0008-x. URL http://www.springerlink.com/index/10.1007/s11263-006-0008-x.

[44] B. G. Baumgart. *Geometric Modeling for Computer Vision*. PhD thesis, Stanford University, 1974. URL http://www.baumgart.org/Bruce_Baumgart_PhD.pdf.

[45] R. Szeliski. Rapid octree construction from image sequences. *CVGIP Image Understanding*, 58:23–32, 1993. URL http://research.microsoft.com/pubs/75650/Szeliski-CVGIPIU93.pdf.

[46] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques - SIGGRAPH*, volume 21, pages 163–169, New York, New York, USA, 1987. ACM Press. ISBN 0897912276. doi: 10.1145/37401.37422. URL http://portal.acm.org/citation.cfm?doid=37401.37422.

[47] J.-S. Franco and E. Boyer. Efficient polyhedral modeling from silhouettes. *IEEE transactions on pattern analysis and machine intelligence*, 31(3):414–27, Mar. 2009. ISSN 0162-8828. doi: 10.1109/TPAMI.2008.104. URL http://www.ncbi.nlm.nih.gov/pubmed/19147872.

[48] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. Image-Based Visual Hulls. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 369–374, 2000. ISBN 1581132085.

[49] E. Boyer and J.-S. Franco. A hybrid approach for computing visual hulls of complex objects. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 695–701. IEEE Comput. Soc, 2003. ISBN 0-7695-1900-8. doi: 10.1109/CVPR.2003.1211421. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1211421.

[50] A. Blake. Invited Letter Visual tracking : a very short research roadmap. *Electronics Letters*, 42(5):254–256, 2006. doi: 10.1049/el.

[51] A. Blake and M. Isard. *Active Contours: The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion*. Springer-Verlag London Ltd., 1998. ISBN 9783540762171.

[52] G. Welch and G. Bishop. An Introduction to the Kalman Filter, 2004.

[53] A. Gouze, C. De Roover, B. Macq, A. Herbulot, E. Debreuve, and M. Barlaud. Watershed-driven active contours for moving object segmentation. In *Proceedings of the IEEE International Conference on Image Processing*, volume 2, page 818. IEEE, 2005. ISBN 0-7803-9134-9. doi: 10.1109/ICIP.2005.1530181. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1530181.

[54] T. F. Cootes. Active Shape Models. URL http://www.isbe.man.ac.uk/~bim/Models/asms.html.

[55] T. F. Cootes, G. Edwards, and C. J. Taylor. Comparing Active Shape Models with Active Appearance Models. In *Proceedings of the British Machine Vision Conference*, pages 173–182, 1999.

[56] P. Felzenszwalb. Representation and detection of deformable shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(2):208–220, Feb. 2005. ISSN 0162-8828. doi: 10.1109/TPAMI.2005.35. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1374867.

[57] J. Pilet. *Augmented Reality for Non-Rigid Surfaces*. PhD thesis, 2008.

[58] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A Database and Evaluation Methodology for Optical Flow. *International Journal of Computer Vision*, 92(1):1–31, Nov. 2010. ISSN 0920-5691. doi: 10.1007/s11263-010-0390-2. URL http://www.springerlink.com/index/10.1007/s11263-010-0390-2http://www.springerlink.com/content/p516733117226378/.

[59] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. T. Freeman. SIFT Flow : Dense Correspondence across Different Scenes. In *Proceedings of the European Conference on Computer Vision*, volume 3, pages 28–42, 2008.

[60] S. Baker and I. Matthews. Lucas-Kanade 20 Years On: A Unifying Framework. *International Journal of Computer Vision*, 56(3):221–255, Feb. 2004. ISSN 0920-5691. doi: 10.1023/B:VISI.0000011205.11775.fd. URL http://www.springerlink.com/openurl.asp?id=doi:10.1023/B:VISI.0000011205.11775.fd.

[61] B. Glocker, N. Paragios, N. Komodakis, G. Tziritas, and N. Navab. Optical flow estimation with uncertainties through dynamic MRFs. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, June 2008. ISBN 978-1-4244-2242-5. doi: 10.1109/CVPR.2008.4587562. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4587562.

[62] A. Bartoli. Image Registration for Nonrigid Surfaces, 2009. URL http://users.isr.ist.utl.pt/~adb/tutorial/2009-tutorial/.

[63] V. Gay-Bellile, A. Bartoli, and P. Sayd. Direct Estimation of Non-rigid Registrations with Image-Based Self-Occlusion Reasoning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):87–104, Jan. 2010. ISSN 0162-8828. doi: 10.1109/TPAMI.2008.265. URL

http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4668346http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4668346.

[64] A. Bartoli and A. Zisserman. Direct Estimation of Non-Rigid Registrations. In *Proceedings of the British Machine Vision Conference*, volume 2, pages 899–908, 2004.

[65] F. Brunet, V. Gay-Bellile, A. Bartoli, N. Navab, and R. Malgouyres. Feature-Driven Direct Non-Rigid Image Registration. *International Journal of Computer Vision*, 93 (1):33–52, Dec. 2010. ISSN 0920-5691. doi: 10.1007/s11263-010-0407-x. URL http://www.springerlink.com/index/10.1007/s11263-010-0407-x.

[66] A. Hilsmann and P. Eisert. Tracking deformable surfaces with optical flow in the presence of self occlusion in monocular image sequences. *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–6, June 2008. doi: 10.1109/CVPRW.2008.4563081. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4563081.

[67] J. Pilet, V. Lepetit, and P. Fua. Real-time nonrigid surface detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 822–828. IEEE, 2005. ISBN 0-7695-2372-2. doi: 10.1109/CVPR.2005.293. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1467352.

[68] L. Masson, M. Dhome, and F. Jurie. Tracking 3D Objects using Flexible Models. In *Proceedings of the British Machine Vision Conference*, 2005.

[69] M. Krinidis, N. Nikolaidis, and I. Pitas. 2-D Feature-Point Selection and Tracking Using 3-D Physics-Based Deformable Surfaces. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(7):876–888, July 2007. ISSN 1051-8215. doi: 10.1109/TCSVT.2007.897463. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4265630.

[70] J. Zhu, M. R. Lyu, and T. S. Huang. A Fast 2D Shape Recovery Approach by Fusing Features and Appearance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(7):1210–1224, July 2009. ISSN 0162-8828. doi: 10.1109/TPAMI.2008.151. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4540099.

[71] M. Salzmann, R. Hartley, and P. Fua. Convex optimization for deformable surface 3-d tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1–8. IEEE, 2007. ISBN 9781424416318. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4409031.

[72] M. Salzmann, J. Pilet, S. Ilic, and P. Fua. Surface deformation models for nonrigid 3D shape recovery. *IEEE transactions on pattern analysis and machine intelligence*, 29(8):1481–7, Aug. 2007. ISSN 0162-8828. doi: 10.1109/TPAMI.2007.1080. URL http://www.ncbi.nlm.nih.gov/pubmed/17568151.

[73] V. Gay-Bellile, M. Perriollat, A. Bartoli, and P. Sayd. Image Registration by Combining Thin-Plate Splines with a 3D Morphable Model. In *Proceedings of the IEEE International Conference on Image Processing*, pages 1069–1072. IEEE, 2006. ISBN 1-4244-0480-0. doi: 10.1109/ICIP.2006.312740. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4106718.

[74] A. Varol, M. Salzmann, E. Tola, and P. Fua. Template-free monocular reconstruction of deformable surfaces. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1811–1818. IEEE, Sept. 2009. ISBN 978-1-4244-4420-5. doi: 10.1109/ICCV.2009.5459403. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5459403.

[75] M. Perriollat, R. Hartley, and A. Bartoli. Monocular Template-based Reconstruction of Inextensible Surfaces. *International Journal of Computer Vision*, 95(2):124–137, May 2010. ISSN 0920-5691. doi: 10.1007/s11263-010-0352-8. URL http://www.springerlink.com/index/10.1007/s11263-010-0352-8.

[76] T. Basha, Y. Moses, and N. Kiryati. Multi-view scene flow estimation: A view centered variational approach. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1506–1513. IEEE, 2010. doi: 10.1109/CVPR.2010.5539791. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5539791.

[77] J. Courchay, J.-P. Pons, P. Monasse, and R. Keriven. Dense and Accurate Spatio-temporal Multi-view Stereovision. In *Computer Vision – ACCV 2009*, Lecture Notes in Computer Science, pages 11–22. Springer, 2010. URL http://www.springerlink.com/index/486150768R794L6L.pdf.

[78] Y. Furukawa and J. Ponce. Dense 3D Motion Capture from Synchronized Video Streams. In R. Ronfard and G. Taubin, editors, *Image and Geometry Processing for 3-D Cinematography*, volume 5 of *Geometry and Computing*, pages 193–211. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-12391-7. doi: 10.1007/978-3-642-12392-4_9. URL http://www.springerlink.com/index/10.1007/978-3-642-12392-4.

[79] E. Aganj, J.-P. Pons, F. Segonne, and R. Keriven. Spatio-Temporal Shape from Silhouette using Four-Dimensional Delaunay Meshing. In *Proceedings of the IEEE Interna-*

tional Conference on Computer Vision. IEEE, 2007. ISBN 978-1-4244-1630-1. doi: 10.1109/ICCV.2007.4409016. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4409016.

[80] K. Varanasi, A. Zaharescu, E. Boyer, and R. Horaud. Temporal Surface Tracking using Mesh Evolution. In *Proceedings of the European Conference on Computer Vision*, volume 2, pages 43–55, 2008.

[81] J. Neumann and Y. Aloimonos. Spatio-temporal stereo using multi-resolution subdivision surfaces. *International Journal of Computer Vision*, 47:181–193, 2002. doi: 10.1109/SMBV.2001.988768. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=988768.

[82] R. W. Zbikowski, R. J. Bomphrey, J. Hoen, A. L. R. Thomas, S. M. Walker, and G. K. Taylor. Study of Experimental Aspects of Dynamic Modelling and Control System Analysis of Hovering Flight of Eristalis Hoverflies, 2006.

[83] P. H. C. Eilers. The Smooth Complex Logarithm and Quasi-Periodic Models. In T. Kneib and G. Tutz, editors, *Statistical Modelling and Regression Structures*, pages 1–17. Physica-Verlag HD, Heidelberg, 2010. ISBN 978-3-7908-2413-1. doi: 10.1007/978-3-7908-2413-1_1. URL http://www.springerlink.com/index/10.1007/978-3-7908-2413-1.

[84] E. W. Weisstein. Line-Line Distance. URL http://mathworld.wolfram.com/Line-LineDistance.html.

[85] P. Kovesi. MATLAB and Octave Functions for Computer Vision and Image Processing. URL http://www.csse.uwa.edu.au/~pk/research/matlabfns/.

[86] S. M. Walker, A. L. R. Thomas, and G. K. Taylor. Photogrammetric reconstruction of high-resolution surface topographies and deformable wing kinematics of tethered locusts and free-flying hoverflies. *Journal of the Royal Society, Interface*, 6(33):351–66, Apr. 2009. ISSN 1742-5689. doi: 10.1098/rsif.2008.0245. URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2658659&tool=pmcentrez&rendertype=abstract.