CV flight tracker: Automatic motion analysis with applications to hummingbird tracking

Contents

1 Introduction

Large-scale video data collection has the potential to advance numerous scientific studies, but manually analyzing this data is extremely laborious. Recent advancements in computer vision can automate such analysis tasks in a variety of different scientific studies. Here, we introduce an open-source package to help analyze videos of hummingbirds, a relatively new application area for computer vision.

Existing computer vision work on birds exists, although data sets and studies are generally limited to classification tasks [1, 2, 3]. However, many motion-analysis tasks can be automated with computer vision [4]. One recent study goes beyond such classification tasks and uses computer vision to track hummingbirds in videos [5]. Specifically, computer vision is used to track (1) the position of a bird over time and (2) the orientation of a bird's body over time.

Here, we automate two tasks related to analysis of animal flight in videos (wing tracking and meniscus tracking) in a simple-to-use python package with a friendly user interface. All code and documentation is available at https://github.com/csinva/hummingbird_tracking. The code is in python and is built on top of OpenCV [6]. It is well-documented and simple to use, with no coding experience necessary. The following two sections detail the wing-tracking method and the meniscus-tracking method, respectively.

2 Wing tracking

In this section, the task is to automatically find the angle between a bird's wings for each frame of a video, as shown in Fig 1. The task is difficult for a few reasons. First, the wing pronates, changing shape between frames. This makes simple point tracking and template-matching methods ineffective. Moreover, data-driven approaches are likely to fail as there is relatively little labelled data and the method must generalize to multiple different types of hummingbirds. Thus, here we use an approach

Preprint. Work in progress.

based on the Hough-line transform, which readily generalizes to birds of different poses, shapes, and sizes.

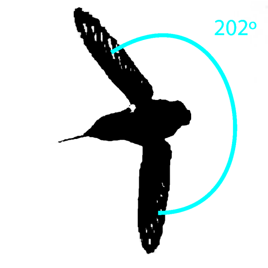

Figure 1: Desired angle shown as cyan curve.

2.1 Method pipeline

The method here takes advantage of the fact that the wings are the segments in the video that change the most frame to frame. Thus, the wings are identified as regions of interest and the background is subtracted. To remove the background, a mixture of gradients approach is used to separate the foreground from the background by analyzing motion between the frames [7]. Fig 2 helps to illustrate this process. The frames in the top row show the original video frames while the frames on the bottom row show the result after doing the background subtraction (where black is the background and gray identifies the regions of interest.

Next, we use a Hough line transform to find line segments in the resulting images [8]. These are shown as the short, dark blue/red segments in the bottom row of images. Now, we cluster these line segments into clusters, one for each wing. Clustering is performed via K-means using the lowest endpoint of each line segment. This can lead to some possible errors when the wings are very close to one another.

Now that each line segment belongs to either the top wing or the bottom wing, we calculate the angle of each line. We then average the angles of the lines for each wing (weighted by the length of the line segments), and use this to calculate the final angle. Calculation of the direction of this angle can be imprecise, as the direction each wing is facing is unknown. Thus angles from previous frames are used to ensure that the current angle is consistent with the information from previous frames.

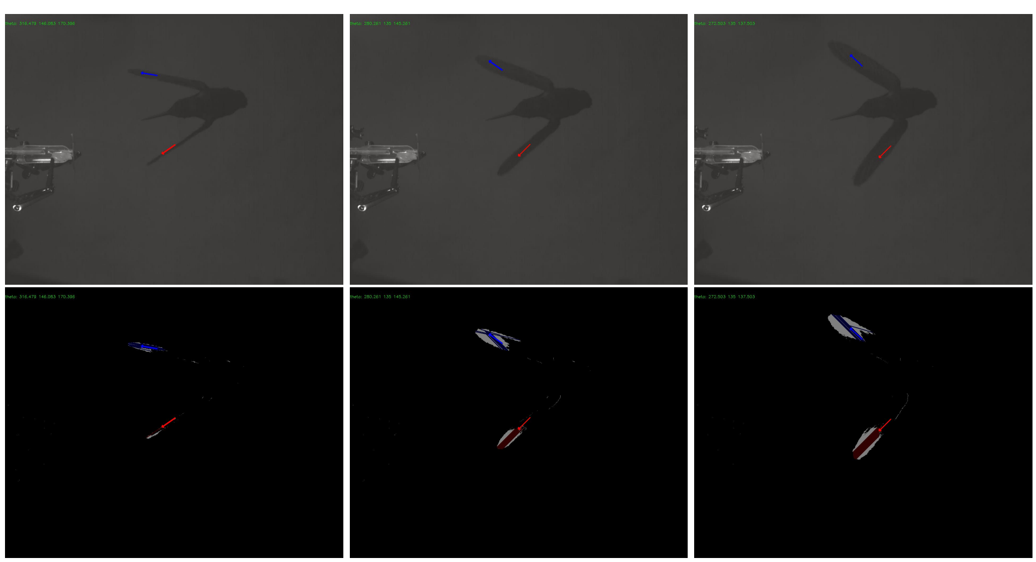

Figure 2: A vector representing each wing's direction is found. Top: vectors overlaid on original frames. Bottom: vectors overlaid on denoised motion between frames along with representative line segments. The final angle is easily calculated from the red and blue arrows.

2.2 Postprocessing

Once angles are extracted from the raw video frames, some postprocessing is done to remove noise from the signal and automatically identify when wingbeats occur. First, raw angle values are smoothed with the Savitsky-Golay filter [9], which removes high-frequency noise from the data. Since wingbeats occur only when wings are fully extended forward or backward, we can safely disregard wingbeats around 180 degrees. Fig 3 shows the results of the postprocessing: The original extracted angles are shown as red points and the smoothed fit is shown as a blue line. Wingbeats (times when the wing is fully extended forwards or backwards) are calculated by finding the local minima and maxima of the smoothed signal. Fig 3 shows that the extracted timing of the wing beats (x-coordinates of the black crosses) match very well with expert-labelled wingbeats (blue triangles).

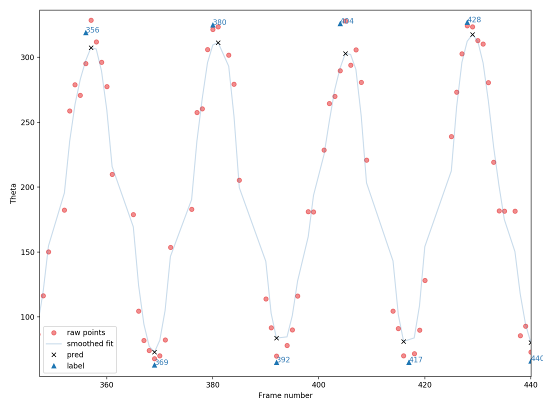

Figure 3: Angle tracked over time with extracted wingbeats and expert-labelled wingbeats.

2.3 Assumptions and limitations

This method makes some assumptions based on the specific dataset at hand. Minor tweaks can fix some of these assumptions to adapt this method to other datasets. The major assumptions are:

- the bird is approximately facing left (if this is not the case, the video should be rotated beforehand)
- the bird's wings are mostly within the frame
- there are no major movements in the video besides the bird
- the wing beat lasts at least a couple frames (i.e. the camera samples faster than the bird beats its wings)

The method makes some rare mistakes. Fig 4 shows the two common types of errors. First, the beginning frames of the video are often not quite correct. Second, there are a few rare errors when there is motion in the video besides the bird. Luckily these mistakes are very noticeable and easily correctable by a human.

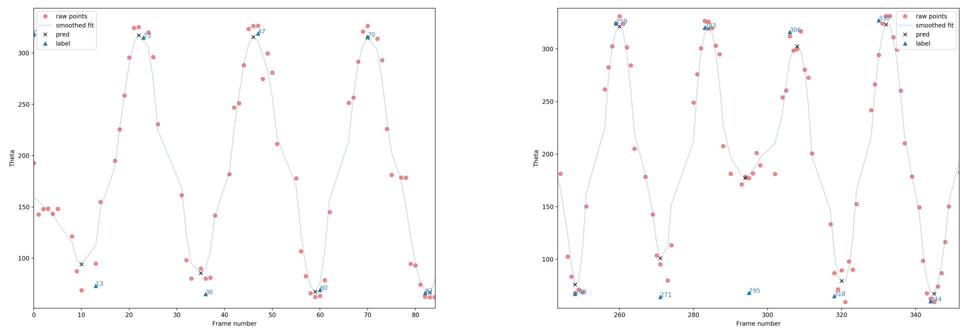

Figure 4: Errors with wing identification. Left shows that there are errors with finding wingbeats for the first few frames of the video. Right show that there are occasionally errors, although they are quite rare and are easily identifiable by a human.

3 Meniscus tracking

The task in this section is to automatically track a receding meniscus in a video. This task is somewhat difficult as the subtle change in the meniscus makes it difficult to track over time. Moreover, different videos illuminate the meniscus differently. In the end, a relatively straightforward method is used which tracks the brightness of pixels in the tube over time and allows a user to specify where the meniscus is, similar to a Kymograph. Though it is not fully automated, this procedure allows one to robustly annotate thousands of frames in seconds.

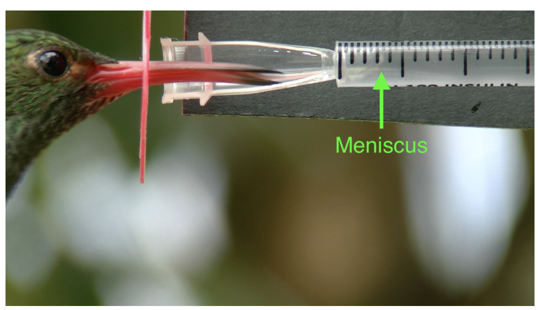

Figure 5: Example frame of a video with a meniscus to be tracked.

3.1 Method pipeline

First, the tube is extracted. This is done by showing an image to the user and having them click the four corners. The tube is then extracted and justified to a rectangle via an affine transformation.

Figure 6: Extracted tube from user clicks.

Next, we use the fact that when as the meniscus moves, it creates lasting, localized motion within the tube. Thus, we subtract the stationary background via a mixture model as we did in Sec 2 and calculate the amount of motion along the tube. Then, since the meniscus is perpendicular to the walls of the tube, we sum the amount of motion along the axis perpendicular to the walls of the tube.

Figure 7 shows the resulting tracked motion over time. Each horizontal band shows the total amount of motion at a point along the tube in Fig 6 for a single frame. The red line is drawn by the user. This is done by selecting points that track the meniscus's position, which generally follows the region of most motion. Linear interpolation is used to fill in the path of the meniscus between user clicks.

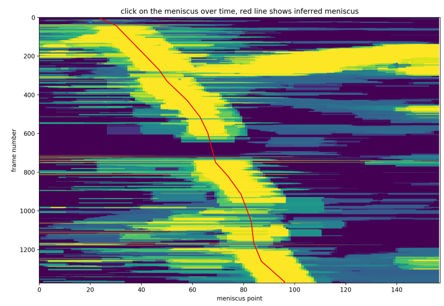

Figure 7: Extracted meniscus

3.2 Assumptions and limitations

This method makes a few key assumptions. First, the method requires that the tube is viewed from the side. Additionally, the meniscus must change occasionally and these changes must be detectable. If the meniscus is stationary for the entire video or obstructed it can not be identified.

References

[1] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[2] Tomoaki Saito, Asako Kanezaki, and Tatsuya Harada. Ibc127: Video dataset for fine-grained bird classification. In *Multimedia and Expo (ICME), 2016 IEEE International Conference on*, pages 1–6. IEEE, 2016.

[3] Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, and Serge Belongie. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 595–604, 2015.

[4] Greg Welch and Eric Foxlin. Motion tracking: No silver bullet, but a respectable arsenal. *IEEE Computer graphics and Applications*, 22(6):24–38, 2002.

[5] Roslyn Dakin, Paolo S Segre, Andrew D Straw, and Douglas L Altshuler. Morphology, muscle capacity, skill, and maneuvering ability in hummingbirds. *Science*, 359(6376):653–657, 2018.

[6] Gary Bradski and Adrian Kaehler. Opencv. *Dr. Dobb's journal of software tools*, 3, 2000.

[7] Pakorn KaewTraKulPong and Richard Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection. In *Video-based surveillance systems*, pages 135–144. Springer, 2002.

[8] Richard O Duda and Peter E Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[9] Abraham Savitzky and Marcel JE Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8):1627–1639, 1964.