Bats Video Segmentation

Biological Signal Analysis – Final Project
Goni Naamani, 200032787
24.2.19

The main goal of the code is to detect and segment individual bats and analyze their movement dynamics during the course of the time-lapse movie. The outputs are a movie in which individual bats are circled in red and collected data about each bat's movement frame by frame.
While writing the code I came across the next challenges:

Bats fly fast

In some frames the bats are blurred due to their movement. I took it into account while writing an algorithm that will be able to detect the bat even if blurred.

Bats look similar to each other

It is difficult to identify individual bats, so I needed to find another way to be able to follow the same individuals from frame to frame. I did it by assuming that each bat will be in the closest location to itself in consecutive frames. More explanation about this in the "DistanceCalcAllBats" and "FindDistance" functions part in the code explanation.

Bats tend to stay close together

The algorithm needed to be able to distinguish between grouped bats that are very close to each other without missing out isolated bats. At first I tried to detected the bats using the function "segment" (it still in the code, but not in use) which worked only with erosion and dilation filters. It did surprisingly well in detecting individual bats, but not so much for grouped bats. To be able to detect individual in a bat clusters I used Watershed implementation, that is able to find the borders between touching objects. I wasn't able to find the distance transform, needed to initialize the sure foreground area, in a good enough manner, so I used erosion and dilation to find the sure foreground and the sure background. I found there is a trade-off between detect all the bats in a close touching cluster and the ability to detect isolated bats. If I eroded enough to detect individuals in a cluster, I usually missed the isolated ones. If I eroded less, than the isolated bats were easily found, but bats cluster would be detected as one or two bats only. In the end I choose something in between that works well in most of the frames that I tried, but probably won't work as well in a frame with a very clustered bats.

The background of the frames is very noisy

A mesh ceiling makes it hard to distinguish between the bats and the background. With the use of morphological filters, I was able to remove most of the noise. Big wood beams on the ceiling were not able to be removed this way, so I segmented them in the first frame and then subtracted them from the other frames. It did remove the noise, but in case bats are located in the wood beams area in the frame, the algorithm is much less sensitive to detect them, and in some cases misses them.
More about it in the "background" and "watershed" functions explanations.
It's probably possible to get better results in a video more zoomed in, which makes the bats more apparent on the background and therefore easier to remove the noise.

There was some changing in the light between some of the frames

Light changes between frames made it so that an algorithm that might work well on one frame won't be as good in another. I tried to find something that works for most of the frames.

Bats go in and out of the frames in the video

With no individual bat identification, it's probably impossible to tell if a bat that entered the frame is the same bat who left the frame before. I decided to index the bats in each frame so that bats got numbers from 1 to 20. If a bat left in the consecutive frame, it "lost" its number, and a new bat that entered the frame might get this number (and therefore be identified as the same bat). Another

solution could be to decide that each bat that left a frame is out, and each bat entering the frame is a new bat, but than the algorithm could be handling hundreds of bats in just a few frames, and I wanted to prevent such a thing to keep it fairly simple.

About the code:

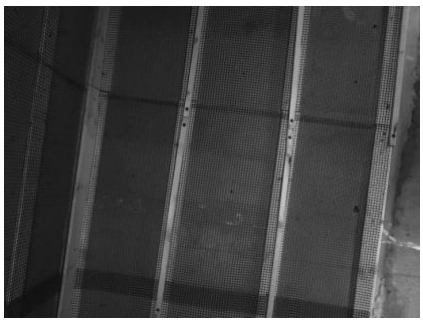
The main code imports functions from "Goni_project_func.py" and contain the function "readVideo".

The function "readVideo" imports and reads the video frame by frame, detects bats and circles them in each frame and saves it into a new video. It also finds the bats locations in each frame and save it into the "BatsInFrames" dictionary. "readVideo" calls the functions "background", "watershed", "find_contours" and "timeStamp".

"readVideo" gets the video to read and the number of frames to read from it. If 0 is given as a number of frames, it reads the whole video. It defines the first frame as background, assuming there are no bats in this frame. The user can choose to define a different background picture.

The function "background" takes the background picture and converts the BGR image into a gray image and then to binary image. Using the morphological filters erosion and dilation on the background image it remains the image with a binary mask of the wood beams on the ceiling. This mask is going to be used later in the function "watershed".

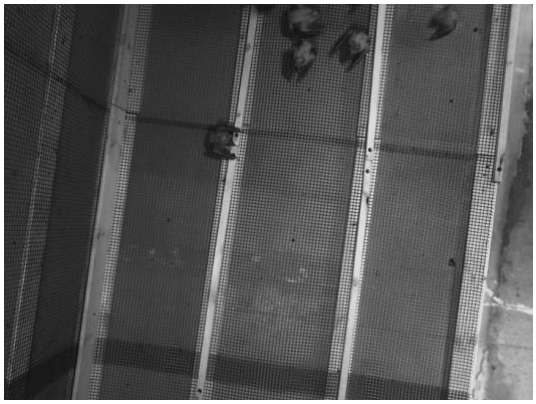
Original image- first frame



"background" output- first frame

The function "watershed" gets the output background mask from the function "background" and a frame.

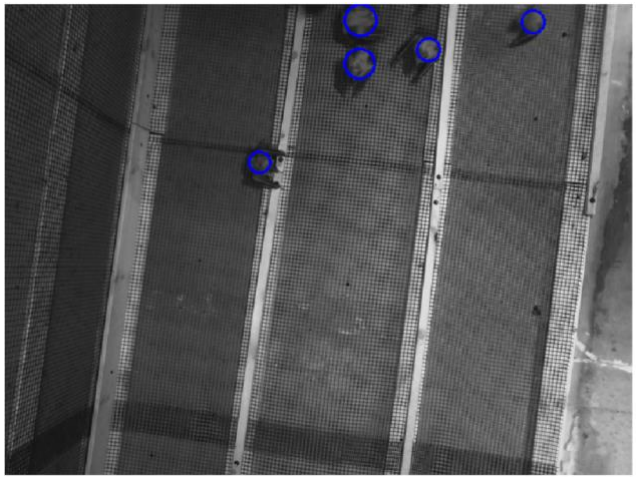
First it converts the BGR frame into a gray image. After binary threshold the function removes most of the noise with the morphological filters erosion and dilation. Big wood beams on the ceiling are not able to be removed this way, so the function uses the background image from before, by subtracting it from the processed frame. After most of the noise had been removed, it remains with the sure foreground mask. It initializes the sure background after opening and dilation. Than It initializes the unknown regions by subtracting them from each other. Than it uses connected component labeling and applying watershed function to the markers (Open cv functions). The function returns the markers.



Original image- frame 2052

"watershed" markers output- frame 2052

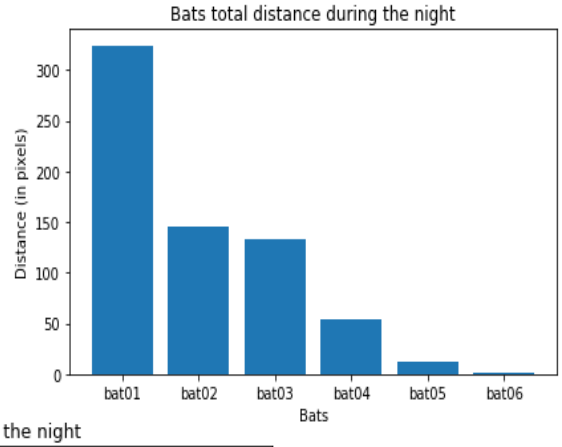
"find_ contours" output- frame 2052



The function "find_ contours" takes the markers, calculate each marker radius and center point. If the radius is in the right size (too small or too big probably means that it is noise) it draws a circle around it and append the center point values into the "centers" list. The function returns the "centers" list and the marked frame. Each center point is saved as a value of a different bat (indexed with numbers) in the "bats_loc" dictionary. The "bats_loc" dictionary with all the bats and their locations is saved as a value of the frame in the "BatsInFrames" dictionary. The function "readVideo" returns the "BatsInFrames" dictionary. In this stage, the bat number in the "BatsInFrames" dictionary could be random, which means that "bat01" in frame95 is not necessarily the same bat as "bat01" in frame 96. To solve this, I assume that each bat will be in the closest location to himself in the next frame.

The function "DistanceCalcAllBats" gets the "BatsInFrames" dictionary, calculates the distance between each bat to each bat in the following frame and returns the results in the "bat_dict" dictionary.

Than **the function "FindDistance"** get the "bat_dict" dictionary. For each bat it checks what was the minimal distance between this bat to any other bat in the following frame. If this minimal distance is smaller than the threshold than it's assumed to be the same bat and the distance is saved to the "batsM_dict" dictionary with the index name of the bat in the earlier frame. For example, if the distance between "bat03" in frame 94 to "bat01" in frame 95 was the minimal one (and smaller than the threshold). It will be saved to bat index "bat03" in frame 95. The result is the "batsM_dict" dictionary that contains the distance that each bat did in all consecutive frames.



The function "calcDistance" get the "batsM_dict" dictionary, calculates the distance that each bat did during the night and **the function "plotDistance"** plots all bats total distance. **The functions "calcMeanMovement" and "plotMeanMovement"** get the "batsM_dict" dictionary, calculate and plot mean movement for all the bats.

The function "plotMovement" gets the "batsM_dict" dictionary and plots for each bat the level of movement in each frame.