Report for End of Year on Progress on Fish Tracking Software

Ari Spraggins

2020-11-19

Abstract

One of the major problem faced by using the locational data of fish in analysis is a tendency for the software to transpose the identities of the two fish whenever they cross. The goal of this program is to analyse that data and to remove that error from it. To do so we are comparing a metric that is unique to the fishes before and after the crossing, in this case a histogram of their brightness, and comparing them.

1 Introduction

2 Methods

The basis of this work is going to be the videos of fish swimming in a tank. One of the things that quickly becomes apparent is that the fish are present as a series of dark pixels against a white background, which we can then feed to a different software (in this case trilab-tracker) to generate a list of pixels that compose the fishes for tracking. Once we have this data for the locations of the fishes, we can determine the areas in which the software detects only one fish, which is the overlapping regions in which errors can occur, which is the areas that we want to concentrate on. Once we have these overlapping regions, we can begin performing operations on the fish during them, firstly the naive approach of tracking the distance between the fish to determine if there was a swap, by comparing the on and off axis distances of the two possible positions of the fish.

```
def swapStatus(pos,i):
    '''
    Detect swaps between consecutive frames based on proximity.

    Input:
        pos:Postionts. Array with shape (Nframes,Nfish,Ndimensions),
        i: Frame index. Int.

    Output:
        Int. 0 if no swaps, 1 if swapped, 2 if overlapping.
    '''
    nFish=pos.shape[1] #Number of fish
    distanceMatrix=[np.linalg.norm(pos[i+1][0]-pos[i][0]),
                    np.linalg.norm(pos[i+1][1]-pos[i][1]),
                    np.linalg.norm(pos[i+1][0]-pos[i][1]),
                    np.linalg.norm(pos[i+1][1]-pos[i][0])]
    swapCriteron=(distanceMatrix[0]+distanceMatrix[1])-(distanceMatrix
    [2]+distanceMatrix[3])
    if abs(swapCriteron)<1e-10:
        return 2 #Overlapping
    elif swapCriteron>0:
        return 1 #Swapped
    elif swapCriteron<0:
        return 0 #Normal
    else:
        return -1
```

Once we have this measure of swapping between the beginning and end of the overlap range, we can both perform an initial unswapping and check any future unswapping attempts against this baseline.

3 Results

4 Discussion

5 Conclusion