Jellies in the ocean: Are they truly drifters?

Sarah Black, Florida Atlantic University

Mentors: Brett Hobson, Jordan Stanway, Alana Sherman, Kakani Katija

Summer 2015

Keywords: JellyCounter, JellyBehavior, Orientation

ABSTRACT

This paper validates a previous code JellyCounter that visually tracks jellyfish and identifies them within a recorded video. Validation is necessary to prove that the objects JellyCounter is identifying are full body jellyfish, not just pieces of them. MATLAB was used to validate JellyCounter and to create a new jellyfish tracking code called JellyBehavior. This code also determines of the orientation of the jellyfish in the videos. Orientation of jellyfish is necessary because there is not much knowledge on how jellyfish travel through the ocean and form large blooms with respect to ocean currents. It is thought they orient themselves someway with respect to ocean currents, but first the orientation of the jellyfish has to be determined. The end result of designing a code that determines orientation was fulfilled.

1. INTRODUCTION

This paper validates an algorithm that autonomously identifies jellyfish from footage taken by a long-range autonomous underwater vehicle (LRAUV) and also determines the orientation of the jellyfish identified. This code can be used on all the LRAUV videos to determine the orientation of all the jellyfish identified and then compare those values with local ocean currents and see how they travel through Monterey Bay. This project was done at the Monterey Bay Aquarium Research Institute (MBARI).

1.1 Objective

Two of jellyfish's main goals are to find food and to reproduce, however, how they travel to do has not been readily studied. Jellyfish are known to be weakly swimming organisms, whose movements are directly influence by ocean currents. (Fosette et all, 2015) Because jellyfish are weak swimmers the impacts from ocean currents are more severe; they are more prone to strandings and reduced bloom behavior. A jellyfish bloom is when large numbers of jellyfish appear suddenly. When jellyfish form these large aggregations it provides them with increased protection, reproduction becomes more efficient, and finding food is not as challenging. However, how can these animals be expected to survive if they are not actively swimming on some specific path to get to a desired location; they need to detect the effects of current on their movements and respond to it.

In the Bay of Biscay, France, researchers found that jellyfish are current oriented swimmers. (Fosette et al., 2105) By manually recording the orientation of jellyfish, researchers found that these jellyfish orient themselves against the current to travel. This was done by having two scientists with hand held compasses manually record the direction of the jellyfish that would pass within three meters of their boat. (Fosette et al., 2105) They discovered at ebb tide jellyfish would swim countercurrent, at flow tide they went with or against the current, and at slack tide they were in all different directions. They ran a virtual experiment jellyfish exhibiting passive behavior and jellyfish exhibiting active behavior at ebb, flow, and slack tides. They found that active jellyfish were more likely to have good bloom behavior and their risk of stranding reduced. (Fosette et al., 2105) This doesn't just pertain to jellyfish in France; another study in

Japan has been done to look at how jellyfish blooms are formed. (Hayami et all., 2007) Instead of manually recording the direction of jellyfish they saw, they used aerial photography to track these jellyfish patches. They found that jellies in the Hokezu Bay, Japan were also found to swim countercurrent in order to aggregate. (Hayami et all., 2007)

Overall if it is determined how jellyfish all over the world orient themselves with respect to ocean currents, important information can be gathered about their bloom formation. Scientists are starting to look at the socio-economic impacts of jellyfish blooms. (Berline et all., 2013) Many times blooms are a threat to tourists and the tourism economy; many beaches have already deployed jellyfish nets to counteract the blooms that appear in these heavily people populated areas. In order to find more information about how jellies orient themselves with respect to ocean currents, a manual approach is not going to be sufficient. With the use of videos taken by LRAUV's, jellyfish movements can be recorded and an autonomous method can be derived to determine the orientation of the jellies in the video.

2. MATERIALS AND METHODS

2.1 Validation of JellyCounter

A previous code called JellyCounter was developed to identify jellyfish in videos taken by the LRAUV, but was never validated. All it did was identify where the objects were with a bounding box as seen in Figure 1. In order to validate JellyCounter a manual approach was taken to identify the number of jellies in the video. The video was read into MATLAB using the VideoReader function and set to read through the video frame by frame. As the code stepped through each frame a manual input of how many jellyfish were seen in the frame was put into the code. The ginput function was then utilized to plot the head and tail locations of each jellyfish identified in the frame. As the points were collected the angle between the head and tail locations with respect to the x axis was calculated by taking the inverse tangent of the two points. The points and corresponding angles were stored in a structure array. The bounding box data from JellyCounter and the manual points were then plotted simultaneously on the videos to compare and validate

the number of jellies counted by the code and the number of jellies counted manually. The JellyCounter code can be seen in Appendix A. The manual point code can be seen in Appendix B.

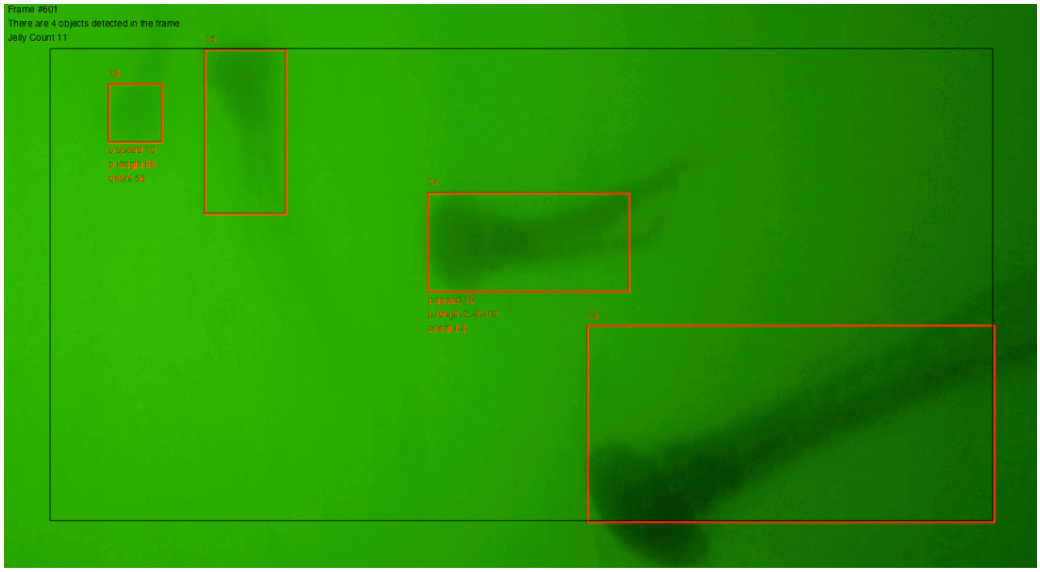

Figure 1: JellyCounter identifying jellyfish with red bounding box in LRAUV video on MATALB

2.2 Creation of JellyBehavior

After plotting the code and manual locations of the jellyfish, a new code JellyBehavior was created to enhance JellyCounter and correctly identify the jellyfish in the video. First the video was read into MATLAB again using the VideoReader function and stepped through the video frame by frame and cropped to the same size as JellyCounter. The video was greyscaled and the complement of the image was taken; the dark jellies turned into bright objects on a dark background. From there, different image processing functions were used.

Top-hat filtering was used to remove any uneven background illumination. A circular structuring element with a radius of 60 pixels was used in JellyCounter, but was then brought down to 50 pixels in JellyBehavior to reduce the noise even more, which made it easier to threshold the rest of the video. A low threshold of 0.075 was chosen, because top-hat filtering was being done. Another structuring element was used to clean up extra noise and fill in holes in the image. In JellyCounter an element with a radius of

20 was chosen, but that was greatly reduced to a radius of 5 in JellyBehavior. This provides the user with a precisely cleaned and segmented binary image. A border was also put around the cropped image. This told the code not to count jellies that were touching the border. The image was then cropped again to the same size as JellyCounter to remove any unwanted noise near the edges. The MATLAB function regionprops was utilized to record: area, bounding box, centroid, major axis length, minor axis length, and orientation.

2.3 Determination of Orientation

In order to determine the correct orientation of all the jellyfish identified in the videos, the data from JellyBehavior had to be sorted through. Using the data from JellyBehavior an ellipse was plotted around the identified jellyfish as seen in Figure 2. Unlike a bounding box an ellipse directly shows the direction and orientation that the jellyfish is traveling. The data was then sorted by first identifying which objects were actually jellyfish. This was done by looking at how many times the objects were counted, if they were counted ten or more times they were considered jellyfish. Then the jellyfish with the largest major axis length was found; the largest major axis length was chosen so there would be a better chance the ellipse would encompass the whole jellyfish body, and therefore give a more accurate orientation. Once all the JellyBehavior data was sorted, it was compared to the manual data. To determine which manual jellyfish corresponded to the jellyfish JellyBehavior identified, the manual data was sorted using the frame number that the JellyBehavior jellyfish were identified in. Then the manual points collected were compared to the centroid of the JellyBehavior points; if they were within 45 pixels of each other the manual points and code points were considered to be the same jellyfish identified. From there, the manual and code orientation could be compared and the percent error was found between them. The JellyBehavior code can be found in Appendix C.

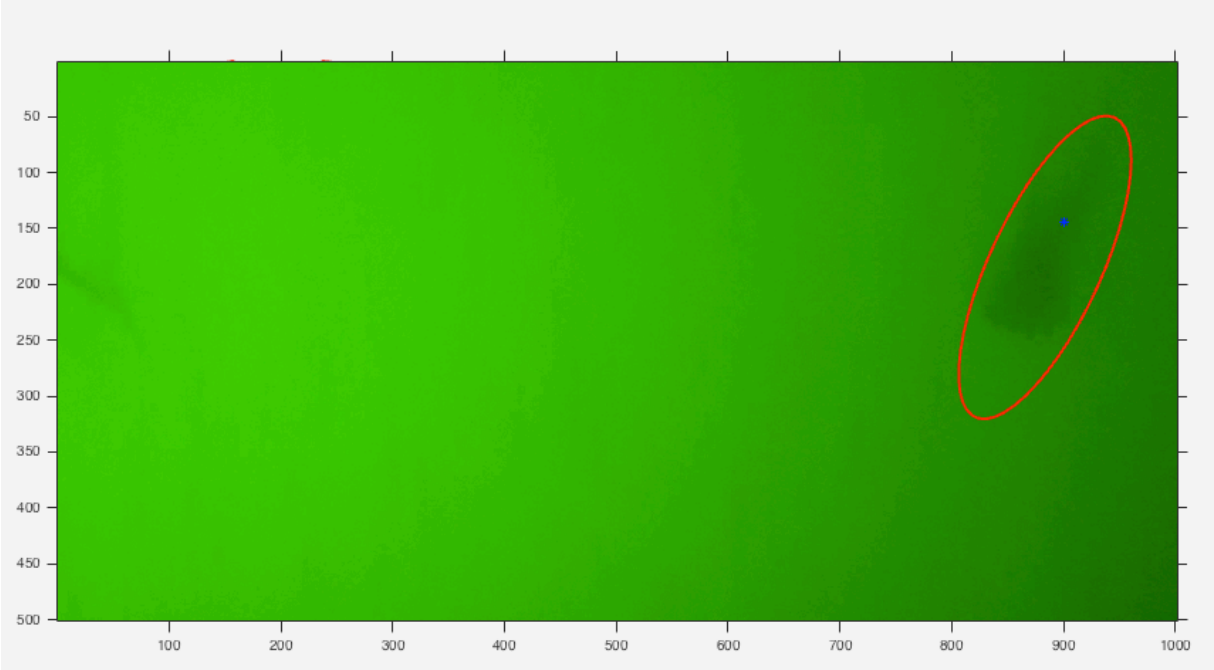

Figure 2: JellyBehavior plotting ellipse around jellyfish identified in LRAUV video

RESULTS AND DISCUSSION

The previous code JellyCounter went frame by frame through the video collected by LRAUV Daphne and identified objects. When the object was first identified a yellow bounding box was put around it and an identification (id) number is given to the specific object. As the code moved on to the next frame and picked up another object, it could tell if that object was the same as the previous by seeing if its centroid was within 50 pixels of the object in the previous frame; if it was within that threshold the object was given the same id number as the previous frame and was considered the same object. If it appears ten or more times the bounding box then turned from yellow to red signifying that that object was a jellyfish. However, in some instances the object the code identified with a red bounding box was not a whole jellyfish body as seen in the figure below. As seen in Figure 3 the code identified part of the tentacles of a single jellyfish as one full jellyfish body.

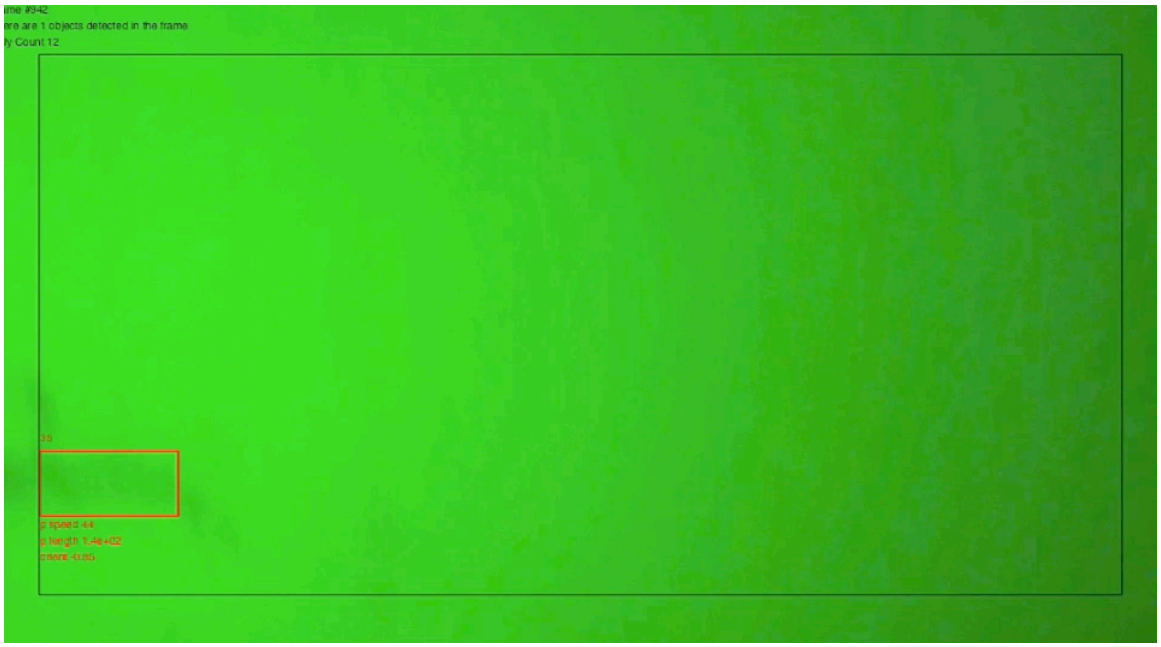

Figure 3: JellyCounter identifying part of tentacle as one full jellyfish body

The code was supposed to identify full bodies of jellyfish not just parts. JellyCounter discarded jellyfish smaller than three pixels, and if they were too faint they were segmented out of the background. This code also could not tell when a jelly appeared behind one; it would just consider it as one animal as seen in Figure 4**.** In order to get an accurate code, jellyfish should not be discarded based on how small they are. If a code can be created that is able to pick up these animals and discard ones that are too close to the edge and whose full bodies are visible, it should be created. To get more accurate results the manual approach was used to validate JellyCounter and new image processing tools were used to make JellyBehavior.

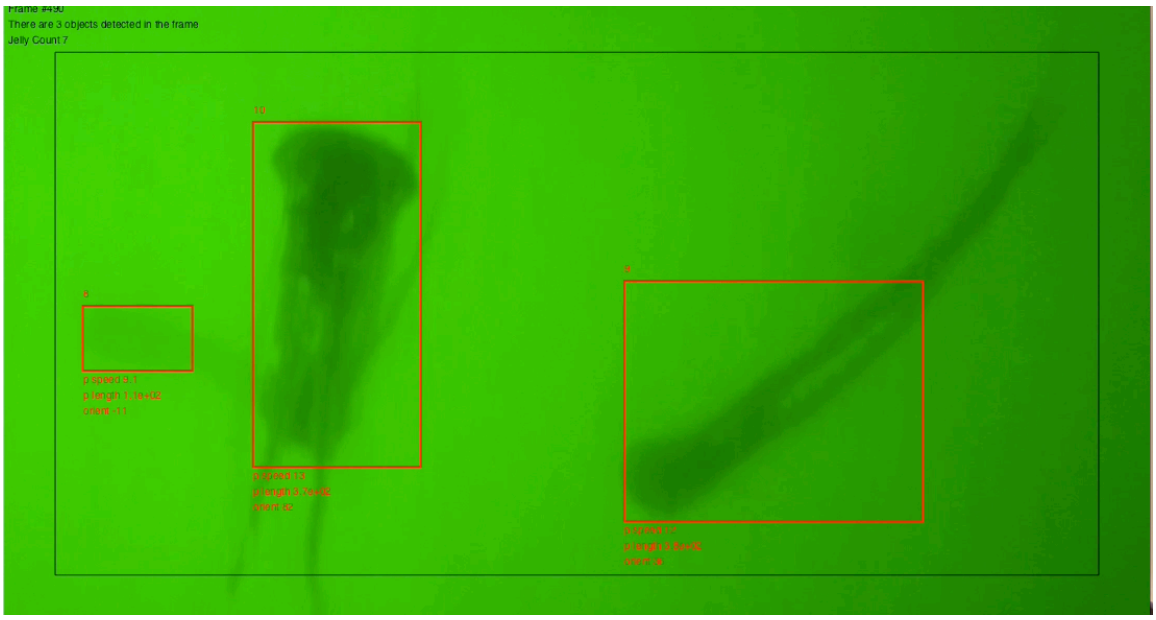

Figure 4: JellyCounter identifying two jellyfish as one

Using the new code JellyBehavior it can be seen in Figure 5 that it does not identify the two jellyfish that are on top of each other, whereas the previous code JellyCounter does. JellyCounter is identified using the bounding box, JellyBehavior is seen with the black ellipses, and the manual points are the blue asterisks.

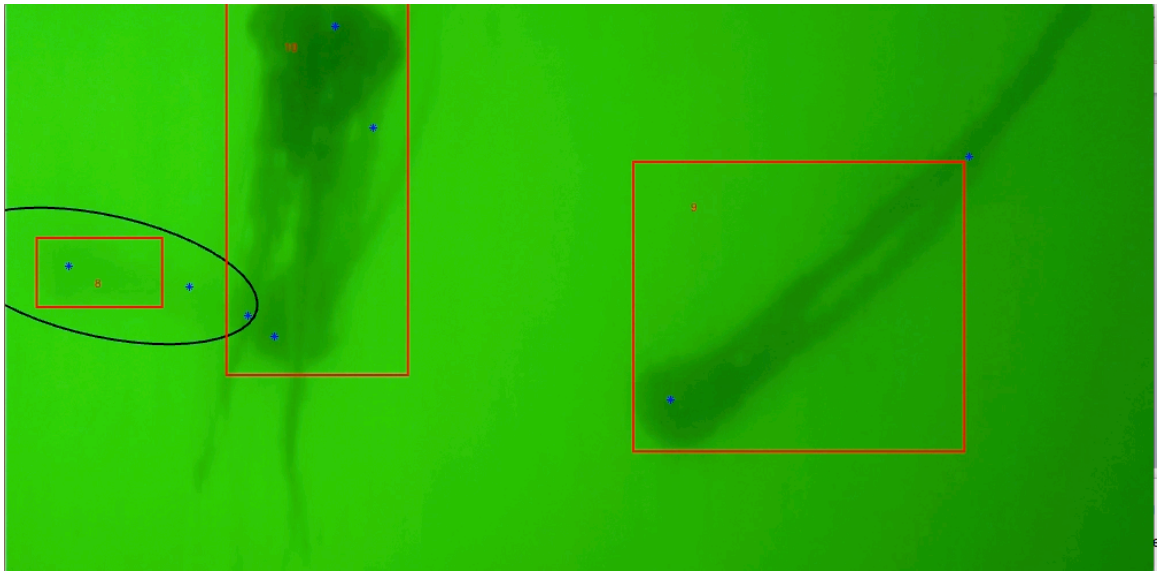

Figure 5: Plotting bounding box, ellipse, and manual points around jellyfish identified

In the table below, the number of jellies collected by the manual approach, by JellyCounter, and by JellyBehavior is shown. Both the manual approach and JellyBehavior identified less jellyfish than JellyCounter. Although the code was

developed to try and identify more jellies, JellyBehavior is a refinement of JellyCounter and it eliminates all the jellyfish that cannot be used for data analysis. It get rids of jellies near edge and jellies that are on top of each other. The orientation data that would be obtained from these two types of jellies would not be accurate. The percent error is also shown between the manual approach and JellyCounter, and the manual approach and JellyBehavior. It shows that JellyBehavior is more accurate. There is still an error, because JellyBehavior picked up two jellyfish that could not be seen in the manual approach. In the next column is the orientation error. The percent error between the manual orientation and the code generated orientation. Overall the percent error is low enough, where it can be concluded that the orientation given by the code is accurate.

	Number of jellies	**Percent error number of jellies**	**Percent error of orientation angle**
Manual	12	-	-
JellyCounter	15	25.0%	-
JellyBehavior	14	16.7%	14.7%

Table 1: Table of results obtained from manual code, JellyCounter, and JellyBehavior

Figure 6 shows a plot of the orientation values of the LRAUV videos in radians. The orientation values are distributed into bins based on the orientation value, and the bin length depends on how many values fall within a certain angle group. It can be seen that not all the jellyfish have the same orientation, but they do favor between 200° and 220°**.** The variety of the orientation values could be due to the fact that the LRAUV is yo-yoing and the orientation values are sometimes taken at different depths. Jellyfish at different depths are at different parts of the water column; therefore they are experience a different current, which could alter the orientation of these animals.

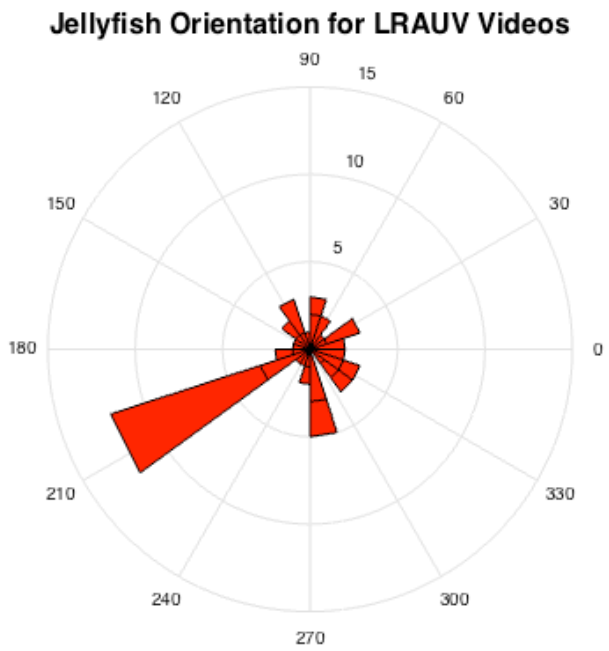

Figure 6: Orientation of all jellyfish identified in LRAUV video

4. CONCLUSIONS AND RECOMMENDATIONS

4.1 Conclusion

Overall a code was developed that can determine the orientation of jellyfish from videos taken by an LRAUV. This code can be applied to all the videos taken to be able to see the orientation of many jellyfish rather than a few. With the determination of the head location of the jellies in the videos a more precise orientation with direction will be able to be known and can then be compared with ocean current data.

4.2 Further Research

In order to better the code that determines the orientation of the jellyfish in the LRAUV videos, the direction of the jellyfish needs to be known. Right now when the MATLAB code determines the orientation of the jellyfish it always does from the left most point, whether that is the head or the tail; it doesn't take into account the direction. In order to get the correct orientation the location of the head needs to be known and that

is where the orientation has to be taken from so the direction is also known. This could be implemented in many ways, but could most easily be done using the regionprops function in MATLAB. The orientation, centroid, and extrema data can be used from the regionprops function. The first step is getting all the jellyfish horizontal, because they are usually all orientated at some angle. To do this they are simply just rotated by their orientation value to become horizontal. Once they are horizontal the left and right most exterior points (extrema) can be found. By looking at the centroid and the extrema points, the head location can be determined by seeing which side the centroid is most close to. For example, if the centroid is situated more to the left, it can be assumed that the head is located on the left side, because the centroid will be closest to wherever the most weight is. This can be applied to all the jellyfish identified by JellyBehavior in order to get the head location and then the correct orientation with respect to the direction.

Once the correct orientation is determined this data can be compared with ocean current data and the direction these jellyfish orient themselves with respect to the ocean currents in the Monterey Bay can be found. This allows us to examine how these animals travel through the ocean and find food and reproduce without getting stranded.

ACKNOWLEDGEMENTS

I would like to thank my mentor Brett Hobson, for help and moral support and making sure what I was doing was something I actually enjoyed and could do. I would like to thank Jordan Stanway for providing me data and helping me get started in the and also Alana Sherman**.** I would especially like to thank Kakani Katija for taking time out of her morning every morning to meet with me and get me on track. She greatly increased my confidence in my MATLAB abilities. Also Fred Bahr, for providing me with CeNCOOS data. Thank you to Brian Kieft for getting me data and Brian Schlining for great MATLAB help. The MBARI internship was such an amazing experience and couldn't be done without George and Linda, so I really want to thank them for running it. And finally the Packard Foundation, without them none of this would be possible.

Appendix A:

JellyCounter MATLAB code

```
clc; close all; clear;

tic
jVid = VideoReader('/Users/sarahblack/Documents/MBARI Summer
2015/Videos/GOPR6616.MP4');
nFrames = jVid.NumberOfFrames;
id = 1;
jCount = 0;
% c = cell([2000,5]); %c{:,1}frame#,c{:,2}id#,c{:,3}(1)centroidx,pixel
distance traveled,#hits
cLength = 1;
%%
for i =1:nFrames
frame = read(jVid,i);
frameCrop = imcrop(frame,[550,150,1100,600]); %xmin ymin width height
frameGrey = rgb2gray(frameCrop);
frameComp = imcomplement(frameGrey);
frameTopHat = imtophat(frameComp,strel('disk',60)); %55
frameThresh = im2bw(frameTopHat,0.03); %0.0265
frameFilled = imfill(frameThresh,'holes');
frameOpen = imopen(frameFilled,strel('disk',20)); %20
frameCrop2 = imcrop(frameOpen,[50,50,1000,500]);
imshow(frameCrop);
[frameLabeled,numObj]=bwlabel(frameCrop2);
stats =
regionprops(frameLabeled,'BoundingBox','Centroid','MajorAxisLength','Or
ientation','MinorAxisLength','Area');
for k = 1:length(stats)

c{cLength,1} = i;
c{cLength,2} = id;
c{cLength,3} = stats(k).Centroid;
c{cLength,4} = 0;
c{cLength,5} = 0;
c{cLength,6} = stats(k).Orientation;
c{cLength,7} = stats(k).MajorAxisLength;
c{cLength,8} = stats(k).MinorAxisLength;
c{cLength,9} = stats(k).Area;
c{cLength,10} = stats(k).BoundingBox;


    for m = cLength-1:-1:1
        if i-c{m,1}>10
            break
        end
        if abs(c{m,3}(1)-stats(k).Centroid(1)) < 50 && abs(c{m,3}(2) -
stats(k).Centroid(2)) < 50
            c{cLength,2} = c{m,2};
            id = id - 1;
            c{cLength,4} = (stats(k).Centroid(2)-c{m,3}(2))/(i-c{m,1});
```

```
            c{cLength,5} = c{m,5}+1;
            break
        end
    end


rect = rectangle('Position',stats(k).BoundingBox+[50,50,0,0],
'LineWidth',2,'Tag',int2str(k));
numLab = text(stats(k).BoundingBox(1)+50,stats(k).BoundingBox(2)+40,
int2str(c{cLength,2}));
if c{cLength,5}==10
    jCount = jCount + 1;
end
if c{cLength,5}<10
   set(rect,'EdgeColor','y');
   set(numLab,'Color','y');
else
   set(rect,'EdgeColor','r');
   set(numLab,'Color','r');
end

if stats(k).BoundingBox(2) > 1 &&
stats(k).BoundingBox(2)+stats(k).BoundingBox(4) < 500 &&
c{cLength,5}>=10

text(stats(k).BoundingBox(1)+50,stats(k).BoundingBox(2)+stats(k).Boundi
ngBox(4)+60,['p speed ',num2str(c{cLength,4},2)],'Color','r');

text(stats(k).BoundingBox(1)+50,stats(k).BoundingBox(2)+stats(k).Boundi
ngBox(4)+75,['p length
',num2str(stats(k).MajorAxisLength,2)],'Color','r');
text(stats(k).BoundingBox(1)+50,stats(k).BoundingBox(2)+stats(k).Boundi
ngBox(4)+90,['orient ',num2str(stats(k).Orientation,2)],'Color','r');
end
cLength = cLength + 1;
id = id + 1;
end
rectangle('Position',[50,50,1000,500]);
text(5,10,['Frame #',int2str(i)]);
text(5,25,['There are ',int2str(numObj),' objects detected in the
frame']);
text(5,40,['Jelly Count ',int2str(jCount)]);

pause(.01);


end
toc
```

Appendix B:

Manual Jellyfish identification MATLAB code

```
%manual determination of jellyfish orientation

clc; clear; close all;
video_file='/Users/sarahblack/Documents/MBARI Summer
2015/Videos/GOPR6616.MP4';
vidObj=VideoReader(video_file);
nFrames=vidObj.Duration*vidObj.FrameRate;

j=1;
for k=1:nFrames
    img=read(vidObj,k);
    frameCrop = imcrop(img,[550,150,1100,600]);
    frameCrop=imcrop(frameCrop,[50,50,1000,500]);
    imnew=rgb2gray(frameCrop);
    imnew=imsubtract(imnew,15);
    imnew=imadjust(imnew);
    figure(1)
    imshow(imnew,[]);
    b=input('number of jellyfish: ');
    pts=ginput(2*b);
    output(j).pts=pts;
    j=j+1;
    for i=1:1:b
        angle(i,1)=atan2(pts(i+1,2)-pts(i,2),pts(i+1,1)-pts(i,1));
    end
    output(j).angle=angle;
end

% save('video_output.mat','output','angle');
```

Appendix C:

JellyBehavior MATLAB code

```
%jellycounter
%save c cell as steck_output
clc; clear; close all;

tic
jVid = VideoReader('/Users/sarahblack/Documents/MBARI Summer
2015/Videos/GOPR6616.MP4');
disp=40;    %distance (in pixels) that one object may travel in between
frames and still be considered the same object
numViews=10;    %number of counts it takes before object is counted as
jellyfish
min_area=150;
max_area=24500;
nFrames = jVid.NumberOfFrames;
id = 1;     %initializing object numbers
jCount = 0; %initializing jellyfish counter
cLength = 1;
%%
for i = 1:1:nFrames
    % Jon Steck's old image operations
    frame = read(jVid,i);
    frameCrop = imcrop(frame,[550,150,1100,600]); %xmin ymin width
height
    frameGrey = rgb2gray(frameCrop);
    frameComp = imcomplement(frameGrey);
    frameTopHat = imtophat(frameComp,strel('disk',60));
    frameThresh = im2bw(frameTopHat,0.02); %0.03
    frameFilled = imfill(frameThresh,'holes');
    frameOpen = imopen(frameFilled,strel('disk',4));
    frameCrop2 = imcrop(frameOpen,[50,50,1000,500]);

    % new image operations
    frameAdjust=imadjust(frameGrey);
    frameCompnew = imcomplement(frameAdjust);
    frameTopHatnew = imtophat(frameCompnew,strel('disk',50));
    % frameBorder=imclearborder(frameTopHatnew,4);
    % index=find(frameTopHat>20);
    % frameTopHat(index)=frameTopHat(index)*10;
    % frameAdjust=imadjust(frameTopHat);
    level=graythresh(frameTopHatnew);
    frameThreshnew = im2bw(frameTopHatnew,0.077);
    frameFillednew = imfill(frameThreshnew,'holes');
    frameBorder = imclearborder(frameFillednew,4);
    frameOpennew = imopen(frameBorder,strel('disk',5));
    frameErode=imerode(frameOpennew,strel('disk',5));
    frameCrop2new = imcrop(frameErode,[50,50,1000,500]);


    %% defining objects, determining statistics
%     [frameLabeled,numObj]=bwlabel(frameCrop2);
```

```
    [frameLabeled,numObj]=bwlabel(frameCrop2new);
    stats =
regionprops(frameLabeled,'BoundingBox','Centroid','MajorAxisLength','Or
ientation','Area','MinorAxisLength');
    index=[];
    for k=1:length(stats)
        if stats(k).Area > min_area && stats(k).Area < max_area
            index=[index,k];
        end
    end
    newstats=stats(index,:);

    figure(3)
    imshow(frameCrop)

    %% collecting data into matrix c
    for k = 1:length(newstats)
        %clength is the index corresponding to the current object
        %c{:,1}frame#,c{:,2}id#,c{:,3}centroid,c{:,4}pixel distance
traveled,c{:,5}#hits,c{:,6}orientation
            c{cLength,1} = i;   %saving frame number
            c{cLength,2} = id;  %initializing object number column
            c{cLength,3} = newstats(k).Centroid;   %saving object's
centroid position
            c{cLength,4} = 0;   %initializing swimming speed column
            c{cLength,5} = 0;   %initializing number of times object
has been counted column
            c{cLength,6} = newstats(k).Orientation;    %saving object's
orientation
            c{cLength,7} = newstats(k).MajorAxisLength;
            c{cLength,8} = newstats(k).MinorAxisLength;
            c{cLength,9} = newstats(k).Area;
%             c{cLength,10}=newstats(k).BoundingBox;

        for m = cLength-1:-1:1
            if i-c{m,1}>10
                break   %if looking outside of 10 frames from current,
stop loop
            end
            % if distance between object in next time step less than
disp
            if abs(c{m,3}(1)-newstats(k).Centroid(1)) < disp &&
abs(c{m,3}(2) -newstats(k).Centroid(2)) < disp
                c{cLength,2} = c{m,2}; %object of this frame is the
same number as the previous one
                id = id - 1;    %this object should have the same
number as the previous one
                c{cLength,4} = (newstats(k).Centroid(2)-c{m,3}(2))/(i-
c{m,1}); %calculating swimming speed from current and previous frame
                c{cLength,5} = c{m,5}+1;    %counting the number of
frames the object is present
                break
            end
        end

        % adding information and details to the plot in figure 3
        rect =
```

```
rectangle('Position',newstats(k).BoundingBox+[50,50,0,0],
'LineWidth',2,'Tag',int2str(k));  %plotting bounding box around each
object
        numLab =
text(newstats(k).BoundingBox(1)+50,newstats(k).BoundingBox(2)+40,
int2str(c{cLength,2}));    %displaying object number
        if c{cLength,5}==numViews %if length of time object identified
is 10 times, count as jellyfish
            jCount = jCount + 1;
        end
        if c{cLength,5}<numViews  %if length of time object identified
is less than 10 times, box is yellow
            set(rect,'EdgeColor','y');
            set(numLab,'Color','y');
        else
            set(rect,'EdgeColor','r');  %if length of time object
identified is greater or equal to 10, box is red
            set(numLab,'Color','r');
        end
        if newstats(k).BoundingBox(2) > 1 &&
newstats(k).BoundingBox(2)+newstats(k).BoundingBox(4) < 500 &&
c{cLength,5}>=10 %if y-box is greater than 1 && y-box+y-height < 500 &&
object has been counted as jellyfish
            % display speed, length, and orientation of object

text(newstats(k).BoundingBox(1)+50,newstats(k).BoundingBox(2)+newstats(
k).BoundingBox(4)+60,['p speed ',num2str(c{cLength,4},2)],'Color','r');

text(newstats(k).BoundingBox(1)+50,newstats(k).BoundingBox(2)+newstats(
k).BoundingBox(4)+75,['p length
',num2str(newstats(k).MajorAxisLength,2)],'Color','r');
text(newstats(k).BoundingBox(1)+50,newstats(k).BoundingBox(2)+newstats(
k).BoundingBox(4)+90,['orient
',num2str(newstats(k).Orientation,2)],'Color','r');
        end
        cLength = cLength + 1; %counter for number of objects
        id = id + 1;    %counter for number of objects
        %how are cLength and id different????
    end
    %displaying outer rectangle, frame number, # objects in frame, and
#jellies counted so far
    rectangle('Position',[50,50,1000,500]);
    text(5,10,['Frame #',int2str(i)]);
    text(5,25,['There are ',int2str(numObj),' objects detected in the
frame']);
    text(5,40,['Jelly Count ',int2str(jCount)]);
    pause(.01);
end
toc
```

References:

Berline, L., K. Guihou, A. Molcard, Y. Ourmières, and B. Zakardjian (2013). Modeling jellyfish *Pelagia noctiluca* transport and stranding in the Ligurian Sea. *Marine Pollution Bulletin,* 70, 90-99.

Fossette, S., A.C. Gleiss, M. Karpytchev, and G.C. Hays (2015). Current-Oriented Swimming by Jellyfish and Its Role in Bloom Maintenance. *Current Biology,* 25, 342-347.

Hayami, Y., A. Kaneda, T. Kohama, S. Magome, S. Takahashi, H. Takeoka, and T. Yamashita (2007). Jellyfish Patch Formation Investigated by Aerial Photography and Drfiter Experiment. *Journal of Oceanography,* 63, 761-773.