COT 5930 DIGITAL IMAGE PROCESSING

DR. HARI KALVA

SPRING 2024

IAN MORGAN-GRAHAM

MATLAB ASSIGNMENT #3 - IMPACT ON OBJECT DETECTION OF FRAME-BY-FRAME VIDEO MANIPULATION

TABLE OF CONTENTS

SECTION 1: INTRODUCTION AND SET UP

RETURN TO TABLE OF CONTENTS

-- INTRODUCTION

The overall approach of this project was to explore the impact of frame-by-frame video manipulation on subsequent object detection. The source video clip used in the project shows a captive chambered nautilus (*Nautilus pompilius*) in a blue-walled aquarium with white sand. In the clip the nautilus slowly moves forward from right to left and consumes three pieces of pink shrimp from the sandy bottom.

The chambered nautilus is a nautiloid, a group of ancient cephalopods related to squids and octopi that have been a part of the fossil record for around 500 million years. Nautiloids have managed to survive several of this planet's mass extinction events including asteroid strikes and global volcanic activity. Fossil recods show that starting around thirty million years ago in the late Eocene epoch, their numbers began to decline from near

global distribution (figure 1). By the Plio-Pleistocene their numbers had fallen and their habitat was limited to the deep waters of the tropical Indo-West Pacific (figure 2). In 2022 researchers (Kiel et. al, 2022) suggested that a reasonable cause of nautiloid decline may be the evolutionary arrival of pinnepeds (seals and their relatives) around 28 million years ago. Fossil records suggest that nautiloids could not withstand pinneped predation (figure3). Currently, all nautilus species are threatened due to overfishing for their shell, which is primarily used for jewelry and other ornamental artifacts. In 2016, they were moved to CITES Appendix II, which restricts international trade, and later the chambered nautilus was recognized as a threatened species under the Endangered Species Act.

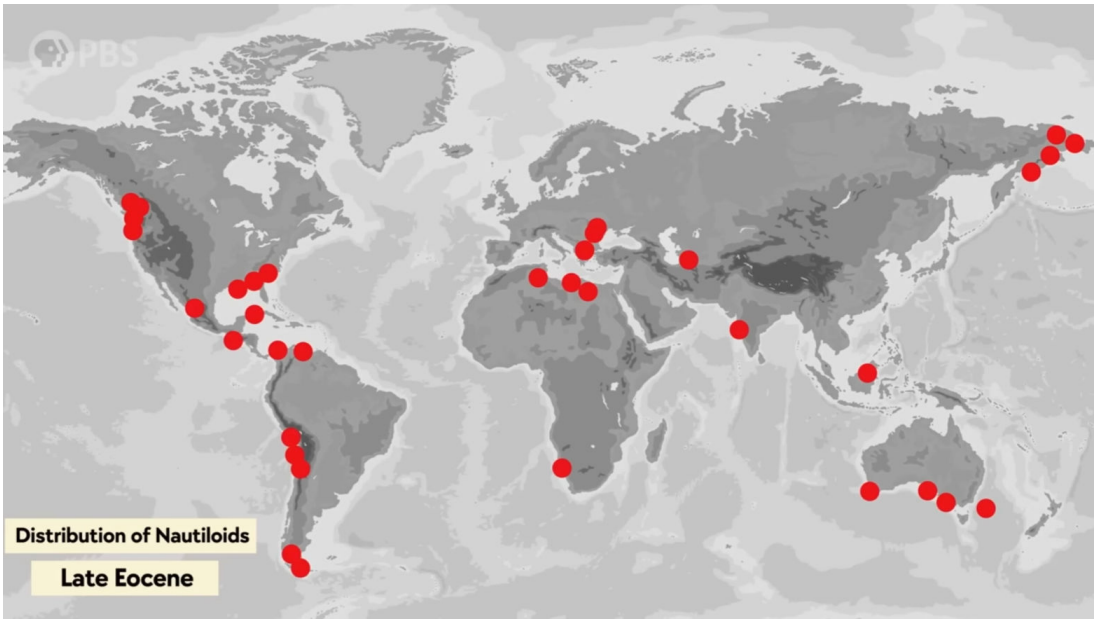

Figure 1. Late Eocene Nautilod Distribution (about 30 million years ago)

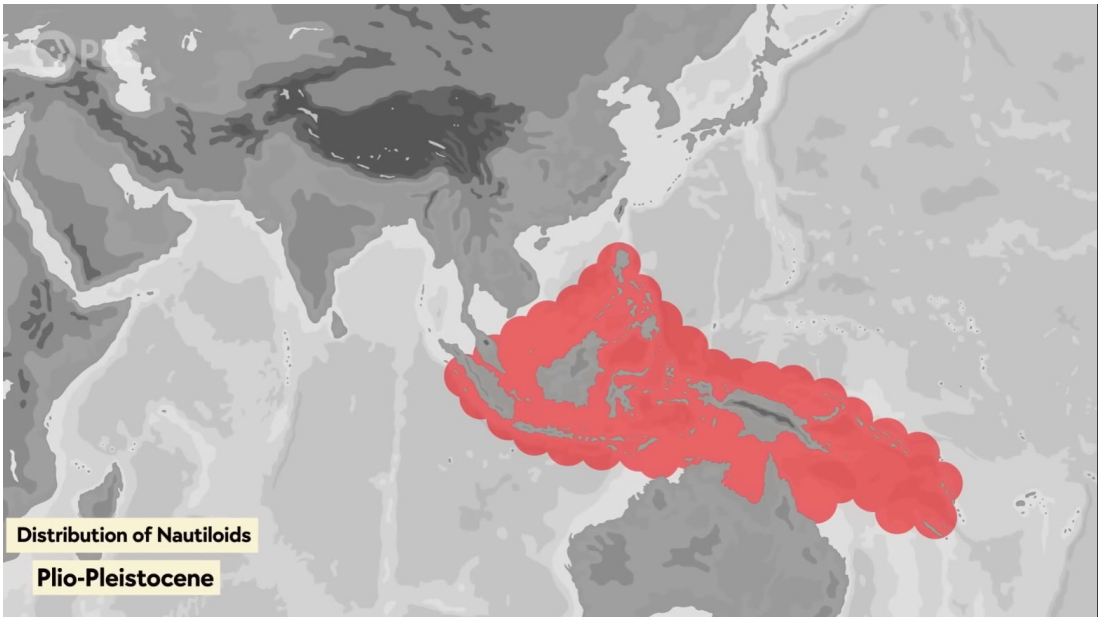

Figure 2. Plio-Pleistocene Nautilod Distribution (about 5 million years ago)

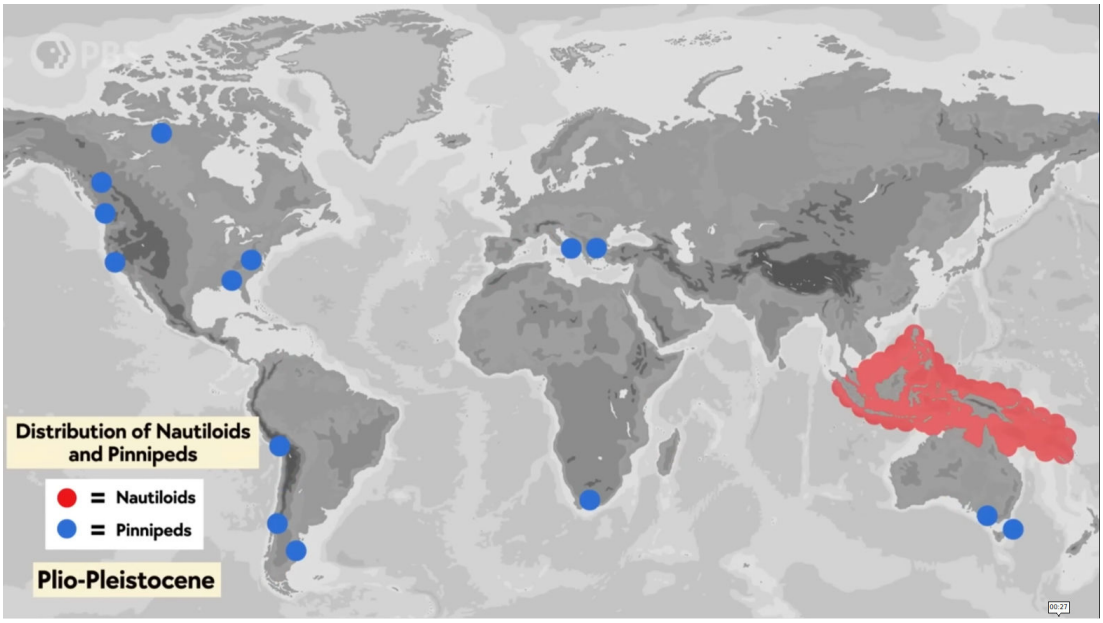

Figure 3. Plio-Pleistocene Nautilod and Pinneped Distribution (about 5 million years ago)

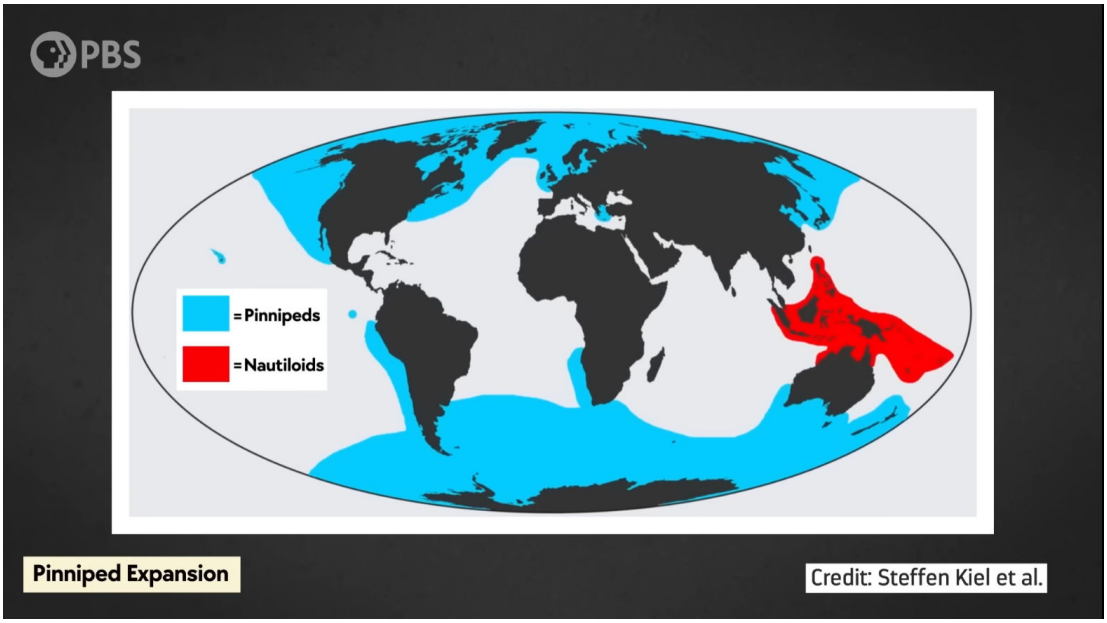

Figure 4. Global Pinneped Distribution

Kiel, S., Goedert, J. L., & Tsai, C. (2022). Seals, whales and the Cenozoic decline of nautiloid cephalopods. *Journal of Biogeography*, *49*(11), 1903–1910. https://doi.org/10.1111/jbi.14488

YouTube. (2023, March 28). *Nautiloids thrived for 500 million years until these guys showed up*. YouTube. https://www.youtube.com/watch?v=3vQ55ToQeWI

Wikimedia Foundation. (2023, November 3). *Chambered nautilus*. Wikipedia. https://en.wikipedia.org/wiki/Chambered_nautilus#cite_note-6

-- SET UP **RETURN TO TABLE OF CONTENTS**

1.) Load needed files into the same contents folder and make a note of the directory path leading to this contents folder.

Main Matlab file:
-- Assignment_3_Frame_by_frame_video_manipulation.mlx
Image folders:
-- FRAMES FOLDER -- OUTPUT FOLDER
Files:
-- chambered_nautilus_original.mp4 -- gTruth_nautilus.mat

Table 1. Files and folder loaded in contents folder

2.) Determine the present working directory with the "pwd" command in Matlab Command Window. Note the ans = 'result' where 'result' is the current working directory.

3.) In the Matlab Command Window, enter "cd" followed by the directory path leading to the contents folder noted in 1.). This will change the current working directory to the directory path leading to the contents folder.

4.) Verify the new present working directory change by retyping "pwd" in the Matlab Command Window.

SECTION 2: PROCEDURE

RETURN TO TABLE OF CONTENTS

All work was done on a linux Ubuntu 22.04.3 LTS operating system.

The project began with a search of YouTube for a cllip of the chambered nautilus (*Nautilus pompilius*). Once a suitable video was located, an open source, linux based video screen capture tool, SimpleScreenRecorder version 0.3.11 (shown in figure 1 and figure 2 below), was used to capture roughly 30 seconds of raw video footage. The SimpleScreenRecorder software allows for selection of specific sections of the screen and output file format (here, .mp4 format was used).

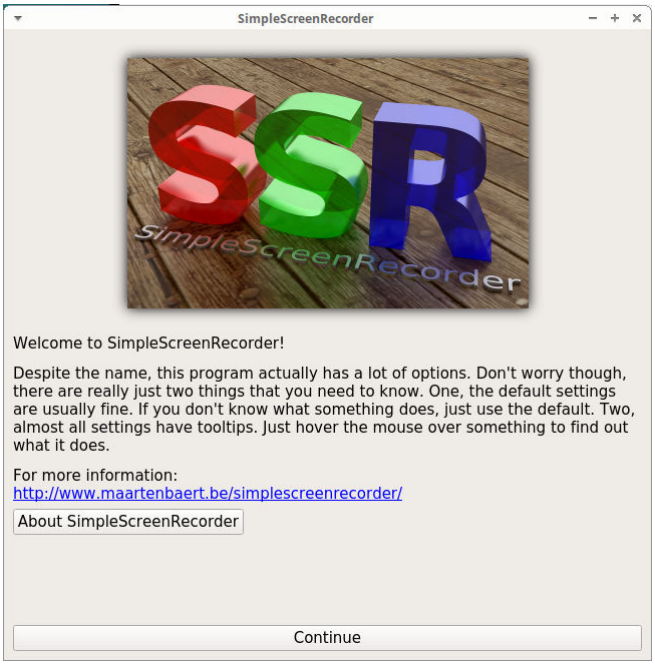

Figure 1. SimpleScreenRecorder start up dialog box

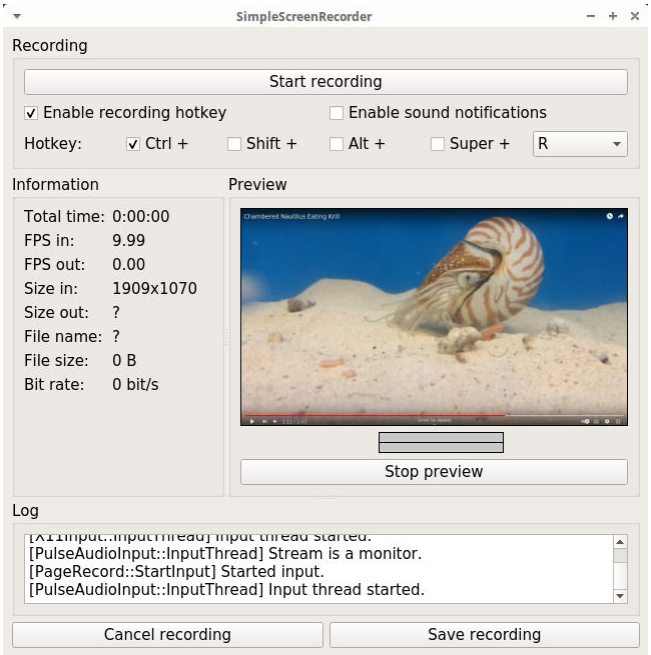

Figure 2. SimpleScreenRecorder capturing section of video from YouTube

The next step was selecting the best 10 seconds from the raw screen captured video. This was done using an open source, linux based video editing tool, Kdenlive version 21.12.3 (shown in figure 3 below). Figure 3 shows a roughly 30 second long raw .mp4 video that was converted into "chambered_nautilus_original.mp4".

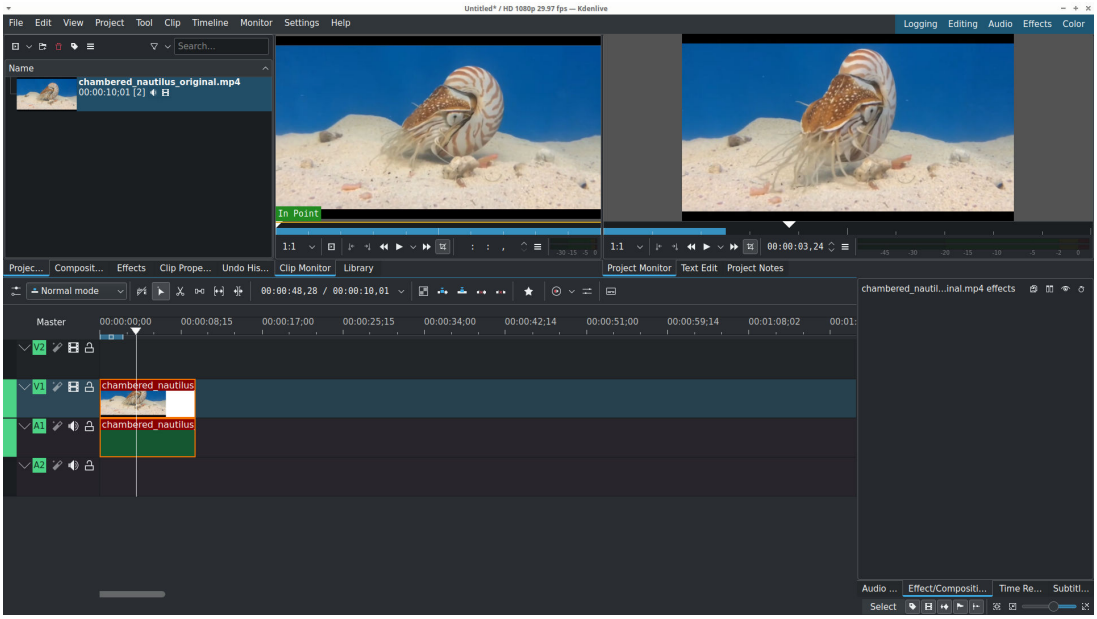

Figure 3. Kdenlive video editing screen.

In the next step the file, "chambered_nautilus_original.mp4", was modified using a frame by frame approach. The modifications were made using the functions shown at the end of this project and described in table 1. Using the functions the original file was either tinted red, tinted green, tinted blue, tinted purple, flipped horizontally (left to right), reversed in play direction, or masked with a center-cross, window-like effect.

Modification Function Name (with inputs)	**Action**
frame_extract_video_deblur (file)	removes any blur from chambered_nautilus_original.mp4 and returns video as a set of frames
colored_video_maker (color_multipliers, frames_bin)	accepts frames and returns tinted frames defined by color_multipliers
video_reverser (frames_bin)	accepts frames and returns frames in reversed play order
left_right_video_flipper (frames_bin)	accepts frames and returns frames flipped horizontally
window_adder (frames_bin)	accepts frames and returns frames with center-cross, window-like effect
video_mash_up (mash_code, input_frames)	accepts deblurred frames and a mash_code then sends the deblurred frames to the other functions for color tinting, reversal, flipping, and windowing if required by the mash_code. Returns the modifified frames and a corrresponding name label string that reflects the modifications (see tables 2 and 3)
video_maker_namer (frames_bin, label_string)	accepts modified frames, and a name label string then merges

	the modified frames into a .avi file with name determined by the name label string (see tables 2 and 3)
video_annotate_function(ground_truth, truth_label, ... video_to_label, output_name, score_cutoff)	accepts ground truth table and video to be labeled using ground truth then returns label annotated .avi file

Table 1. Modification functions RETURN TO TABLE OF CONTENTS

	1	0
First digit	apply red tint	do not apply red tint
Second digit	apply green tint	do not apply greeen tint
Third digit	apply blue tint	do not apply blue tint
Fourth digit	apply purple tint	do not apply purple tint
Fifth digit	apply horizontal flip	do not apply horizontal flip
Sixth digit	apply reverse play	do not apply reverse play
Seventh digit	apply apply window	do not apply window

Table 2. Mash_code interpretation RETURN TO TABLE OF CONTENTS

red(+) = red tint applied	red(-) = red tint not applied
green(+) = green tint applied	green(-) = green tint not applied
blue(+) = blue tint applied	blue(-) = blue tint not applied
purple(+) = purple tint applied	purple(-) = purple tint not applied
flp(+) = horizontal flip applied	flp(-) = horizontal flip not applied
rev(+) = reverse play applied	rev(-) = reverse play not applied
win(+) = window applied	win(-) = window not applied

Table 3. Modified video naming convention RETURN TO TABLE OF CONTENTS

EXAMPLE:

Construct a red tinted, reversed, horizontal flipped, windowed version of frames from original video clip, "chambered_nautilus_deblurred.avi"

Associated mash_code = [1,0,0,0,1,1,1]

Resulting modified video name: "chambered_nautilus_red(+)grn(-)blu(-)prp(-)flp(+)rev(+)win(+).avi"

Overall, the procedure was:

i.) Deblur "chambered_nautilus_original.mp4" into "chambered_nautilus_deblurred.avi" with associated deblurred frames.

ii.) Modify the deblurred frames into modified .avi files using various mash_codes to direct the modifications functions.

iii.) Use Matlab VideoLabeler app and the file "chambered_nautilus_deblurred.avi" to generate a ground truth table.

iv.) Use the ground truth table to attempt to label the modified .avi tables to crudely assess the impact of frame-by-frame

modification on object detection and video annotation.

-- REMOVE BLUR FROM ORIGINAL VIDEO

RETURN TO TABLE OF CONTENTS

```
% Extract frames from original .mp4 video into folder
% Construct original deblurred video
deblurred_frames_bin = frame_extract_video_deblur
('chambered_nautilus_original.mp4');
```

```
Deblurred frames are saved in the folder: FRAMES FOLDER
Deblurred frames are in workspace as 'deblurred_frames_bin'
The deblurred .avi clip is in current working folder
```

-- CONSTRUCTING VARIANT VIDEOS

RETURN TO TABLE OF CONTENTS

```
% Call video_mash_up and video_maker_namer functions to construct various
%    modified videos

% Construct a red tinted version of original video clip
mash_code_0 = [1,0,0,0,0,0,0];
[result_frames, label_string] = video_mash_up (mash_code_0,
deblurred_frames_bin);
video_maker_namer (result_frames, label_string);
```

```
A new modified .avi clip is now in the working folder
```

```
% Construct a blue tinted, reversed, horizontal flipped, windowed version of
original video clip
mash_code_1 = [0,0,1,0,1,1,1];
[result_frames, label_string] = video_mash_up (mash_code_1,
deblurred_frames_bin);
video_maker_namer (result_frames, label_string);
```

```
A new modified .avi clip is now in the working folder
```

```
% Construct a reversed, windowed version of original video clip
```

```
mash_code_2 = [0,0,0,0,0,1,1];
[result_frames, label_string] = video_mash_up (mash_code_2,
deblurred_frames_bin);
video_maker_namer (result_frames, label_string);
```

```
A new modified .avi clip is now in the working folder
```

```
% Construct a purple tinted, horizontal flipped version of original video
clip
mash_code_3 = [0,0,0,1,0,1,1];
[result_frames, label_string] = video_mash_up (mash_code_3,
deblurred_frames_bin);
video_maker_namer (result_frames, label_string);
```

```
A new modified .avi clip is now in the working folder
```

```
% Construct a green tinted, flipped, reversed version of original video clip
mash_code_4 = [0,1,0,0,1,1,0];
[result_frames, label_string] = video_mash_up (mash_code_4,
deblurred_frames_bin);
video_maker_namer (result_frames, label_string);
```

```
A new modified .avi clip is now in the working folder
```

```
% Construct a purple tinted, windowed version of original video clip
mash_code_5 = [0,0,0,1,0,0,1];
[result_frames, label_string] = video_mash_up (mash_code_5,
deblurred_frames_bin);
video_maker_namer (result_frames, label_string);
```

```
A new modified .avi clip is now in the working folder
```

-- EXPLORING IMPACT OF FRAME-BY-FRAME VIDEO MANIPULATION ON OBJECT DETECTION

RETURN TO TABLE OF CONTENTS

```
% Investigate influence of blue color, horizontal vido flipping, video
reversal
%    and window presence on object detection
% clear previous outputs and workspace data
close all;
clear;
clc;
% load appropriate ground truth table (gTruth)
load('gTruth_nautilus.mat');
```

```
% call function that will use ground truth table to label objects in a new
video file
% provide "video_annotate_function" function with :
%   ground truth table
%   label name within ground truth table
%   video file to be labeled with label name
%   cutoff for detected matches between ground truth table and objects in
video file to
%       be labeled
video_to_label =
"chambered_nautilus_red(-)grn(-)blu(+)prp(-)flp(+)rev(+)win(+).avi"
```

```
video_to_label =
"chambered_nautilus_red(-)grn(-)blu(+)prp(-)flp(+)rev(+)win(+).avi"
```

```
video_new_name = "version_blu(+)flp(+)rev(+)win(+)"
```

```
video_new_name =
"version_blu(+)flp(+)rev(+)win(+)"
```

```
video_annotate_function(gTruth, "nautilus", video_to_label, video_new_name,
25)
```

```
Write images extracted for training to folder:
    /media/ijmg/SSD_FOUR_TB/ACADEMICS_101/a_Florida Atlantic University/MSOE/COT 5930 Dig Image Process/As

Writing 31 images extracted from chambered_nautilus_clip.mp4...Completed.
ACF Object Detector Training
The training will take 5 stages. The model size is 638x683.
Sample positive examples(~100% Completed)
Compute approximation coefficients...Completed.
Compute aggregated channel features...Completed.
----------------------------------------
Stage 1:
Sample negative examples(~100% Completed)
Compute aggregated channel features...Completed.
Train classifier with 31 positive examples and 155 negative examples...Completed.
The trained classifier has 19 weak learners.
----------------------------------------
Stage 2:
Sample negative examples(~100% Completed)
Found 1 new negative examples for training.
Compute aggregated channel features...Completed.
Train classifier with 31 positive examples and 155 negative examples...Completed.
The trained classifier has 19 weak learners.
----------------------------------------
Stage 3:
Sample negative examples(~100% Completed)
Found 1 new negative examples for training.
Compute aggregated channel features...Completed.
Train classifier with 31 positive examples and 155 negative examples...Completed.
The trained classifier has 19 weak learners.
----------------------------------------
Stage 4:
Sample negative examples(~100% Completed)
Found 1 new negative examples for training.
Compute aggregated channel features...Completed.
Train classifier with 31 positive examples and 155 negative examples...Completed.
The trained classifier has 19 weak learners.
----------------------------------------
```

```
Stage 5:
Sample negative examples(~100% Completed)
Found 1 new negative examples for training.
Compute aggregated channel features...Completed.
Train classifier with 31 positive examples and 155 negative examples...Completed.
The trained classifier has 19 weak learners.
---------------------------------------
ACF object detector training is completed. Elapsed time is 89.7215 seconds.
workingDir =
'./OUTPUT/'
A new annotated .avi clip is now in the folder, 'OUTPUT'
```

```
% Investigate influence of video reversal and window presence on object
detection
% clear previous outputs and workspace data
close all;
clear;
clc;
% load appropriate ground truth table (gTruth)
load('gTruth_nautilus.mat');
video_to_label =
"chambered_nautilus_red(-)grn(-)blu(-)prp(-)flp(-)rev(+)win(+).avi"
```

```
video_to_label =
"chambered_nautilus_red(-)grn(-)blu(-)prp(-)flp(-)rev(+)win(+).avi"
```

```
video_new_name = "version_rev(+)win(+)"
```

```
video_new_name =
"version_rev(+)win(+)"
```

```
video_annotate_function(gTruth, "nautilus", video_to_label, video_new_name,
25)
```

```
Write images extracted for training to folder:
    /media/ijmg/SSD_FOUR_TB/ACADEMICS_101/a_Florida Atlantic University/MSOE/COT 5930 Dig Image Process/As

Writing 31 images extracted from chambered_nautilus_clip.mp4...Completed.
ACF Object Detector Training
The training will take 5 stages. The model size is 638x683.
Sample positive examples(~100% Completed)
Compute approximation coefficients...Completed.
Compute aggregated channel features...Completed.
---------------------------------------
Stage 1:
Sample negative examples(~100% Completed)
Compute aggregated channel features...Completed.
Train classifier with 31 positive examples and 155 negative examples...Completed.
The trained classifier has 19 weak learners.
---------------------------------------
Stage 2:
Sample negative examples(~100% Completed)
Found 1 new negative examples for training.
Compute aggregated channel features...Completed.
Train classifier with 31 positive examples and 155 negative examples...Completed.
The trained classifier has 19 weak learners.
---------------------------------------
```

```
Stage 3:
Sample negative examples(~100% Completed)
Found 1 new negative examples for training.
Compute aggregated channel features...Completed.
Train classifier with 31 positive examples and 155 negative examples...Completed.
The trained classifier has 19 weak learners.
---------------------------------------
Stage 4:
Sample negative examples(~100% Completed)
Found 1 new negative examples for training.
Compute aggregated channel features...Completed.
Train classifier with 31 positive examples and 155 negative examples...Completed.
The trained classifier has 19 weak learners.
---------------------------------------
Stage 5:
Sample negative examples(~100% Completed)
Found 1 new negative examples for training.
Compute aggregated channel features...Completed.
Train classifier with 31 positive examples and 155 negative examples...Completed.
The trained classifier has 19 weak learners.
---------------------------------------
ACF object detector training is completed. Elapsed time is 89.1416 seconds.
workingDir =
'./OUTPUT/'
A new annotated .avi clip is now in the folder, 'OUTPUT'
```

```
% Investigate influence of red color on object detection
% clear previous outputs and workspace data
close all;
clear;
clc;
% load appropriate ground truth table (gTruth)
load('gTruth_nautilus.mat');
video_to_label =
"chambered_nautilus_red(+)grn(-)blu(-)prp(-)flp(-)rev(-)win(-).avi"
```

```
video_to_label =
"chambered_nautilus_red(+)grn(-)blu(-)prp(-)flp(-)rev(-)win(-).avi"
```

```
video_new_name = "version_red(+)"
```

```
video_new_name =
"version_red(+)"
```

```
video_annotate_function(gTruth, "nautilus", video_to_label, video_new_name,
25)
```

```
Write images extracted for training to folder:
    /media/ijmg/SSD_FOUR_TB/ACADEMICS_101/a_Florida Atlantic University/MSOE/COT 5930 Dig Image Process/As

Writing 31 images extracted from chambered_nautilus_clip.mp4...Completed.
ACF Object Detector Training
The training will take 5 stages. The model size is 638x683.
Sample positive examples(~100% Completed)
Compute approximation coefficients...Completed.
Compute aggregated channel features...Completed.
---------------------------------------
```

```
Stage 1:
Sample negative examples(~100% Completed)
Compute aggregated channel features...Completed.
Train classifier with 31 positive examples and 155 negative examples...Completed.
The trained classifier has 19 weak learners.
----------------------------------------
Stage 2:
Sample negative examples(~100% Completed)
Found 1 new negative examples for training.
Compute aggregated channel features...Completed.
Train classifier with 31 positive examples and 155 negative examples...Completed.
The trained classifier has 19 weak learners.
----------------------------------------
Stage 3:
Sample negative examples(~100% Completed)
Found 1 new negative examples for training.
Compute aggregated channel features...Completed.
Train classifier with 31 positive examples and 155 negative examples...Completed.
The trained classifier has 19 weak learners.
----------------------------------------
Stage 4:
Sample negative examples(~100% Completed)
Found 1 new negative examples for training.
Compute aggregated channel features...Completed.
Train classifier with 31 positive examples and 155 negative examples...Completed.
The trained classifier has 19 weak learners.
----------------------------------------
Stage 5:
Sample negative examples(~100% Completed)
Found 1 new negative examples for training.
Compute aggregated channel features...Completed.
Train classifier with 31 positive examples and 155 negative examples...Completed.
The trained classifier has 19 weak learners.
----------------------------------------
ACF object detector training is completed. Elapsed time is 88.4349 seconds.
workingDir =
'./OUTPUT/'
A new annotated .avi clip is now in the folder, 'OUTPUT'
```

SECTION 3: RESULTS

RETURN TO TABLE OF CONTENTS

After completing the above sections of code:

i.) The folder, "FRAMES FOLDER", will be created (if not already present) and loaded with 300 deblurred frames that make up the file, "chambered_nautilus_deblurred.avi"

ii.) The file, "chambered_nautilus_deblurred.avi" should be present in the working folder

iii.) The working folder should contain several modified video files:

-- "chambered_nautilus_red(+)grn(-)blu(-)prp(-)flp(-)rev(-)win(-).avi"

-- "chambered_nautilus_red(-)grn(+)blu(-)prp(-)flp(+)rev(+)win(-).avi"

-- "chambered_nautilus_red(-)grn(-)blu(+)prp(-)flp(+)rev(+)win(+).avi"

-- "chambered_nautilus_red(-)grn(-)blu(-)prp(+)flp(-)rev(+)win(+).avi"

-- "chambered_nautilus_red(-)grn(-)blu(-)prp(+)flp(-)rev(-)win(+).avi"

-- "chambered_nautilus_red(-)grn(-)blu(-)prp(-)flp(-)rev(+)win(+).avi"

iv.) The folder, "OUTPUT", should be created (if not already present) and contain three attempts to ground truth annotate several of the modified files:

-- "version_blu(+)flp(+)rev(+)win(+).avi"

-- "version_red(+).avi"

-- "version_rev(+)win(+).avi"

SECTION 4: FUNCTIONS

RETURN TO TABLE OF CONTENTS

```
function [deblurred_frames_bin] = frame_extract_video_deblur (file)
    % Create a video_reader object
    video_reader_object = VideoReader(file);

    % Create a video_writer object to construct reversed video
    deblurred_video = VideoWriter('chambered_nautilus_deblurred.avi',
'Uncompressed AVI');
    open(deblurred_video);

    % Create a folder to store the frames if not already constructed
    extracted_frames_folder = 'FRAMES FOLDER';
    if ~exist(extracted_frames_folder, 'dir')
        mkdir(extracted_frames_folder);
    end

    % Get information about the video
    num_frames = video_reader_object.NumFrames;

    % Dictionary to hold deblurred frames
    deblurred_frames_bin = {};

    % Make a Wiener deconvolution filter to deblur each frame
    PSF_kernel = 15;  % Point spread function (PSF) kernel determined by
trial and error
    noise_level_estimate = 0.1;  % Estimated noise level
    wiener_filter = fspecial('gaussian', [PSF_kernel, PSF_kernel],
noise_level_estimate);

    % --- 1.) Read and deblur each frame
    % --- 2.) Save each current deblurred frame as a JPEG image into 'FRAMES
FOLDER'
    % --- 3.) Save each current deblurred frame to deblurred frames bin
    % --- 4.) Write each current deblurred frame to deblurred video
    % Loop through each frame
```

```
    for frame_index = 1:num_frames
        % --- 1.) Read and deblur each frame
        current_blurred_frame = read(video_reader_object, frame_index);
        % Deblur each frame using Wiener deconvolution
        % Given (true image detail) [convolved with] (PSF) = (blurred
image), to recover the
        %    true image detail use (true image detail) = (blurred image)
[deconvolved with] (PSF)
        current_deblurred_frame = deconvwnr(current_blurred_frame,
wiener_filter, noise_level_estimate);

        % --- 2.) Save each current deblurred frame as a JPEG image into
'FRAMES FOLDER'
        % At about 30 fps with a 10 second .mp4 clip, about 300 frames
of .jpeg
        %    images should result in the FRAMES FOLDER
        current_deblurred_frame_name = sprintf('frame_%04d.jpeg',
frame_index);
        imwrite(current_deblurred_frame, fullfile(extracted_frames_folder,
current_deblurred_frame_name));

        % --- 3.) Save each current deblurred frame to deblurred frames bin
        % At about 30 fps with a 10 second .mp4 clip, about 300 deblurred
        %    frames should result in the "deblurred_frames_bin"
        deblurred_frames_bin{frame_index} = current_deblurred_frame;

        % --- 4.) Write each current deblurred frame to deblurred video
        writeVideo(deblurred_video, deblurred_frames_bin{frame_index});
    end
    % Close the deblurred video
    close(deblurred_video);
    disp(['Deblurred frames are saved in the folder: '
extracted_frames_folder]);
    disp(["Deblurred frames are in workspace as 'deblurred_frames_bin'" ]);
    disp(['The deblurred .avi clip is in current working folder']);
end

function [tinted_frames_bin] = colored_video_maker (color_multipliers,
frames_bin)
    % Dictionary to hold tinted frames
    tinted_frames_bin = {};
    % color multipliers
    red_mult = color_multipliers(1);
    green_mult = color_multipliers(2);
    blue_mult  = color_multipliers(3);
    % Write frames in order to the video
    for frame_index = 1:1:numel(frames_bin)
        % Extract the red, green, and blue channels
        red_channel = frames_bin{frame_index}(:, :, 1);
        green_channel = frames_bin{frame_index}(:, :, 2);
```

```
        blue_channel = frames_bin{frame_index}(:, :, 3);
        tinted_frame = cat(3, red_mult * red_channel, green_mult *
green_channel, blue_mult * blue_channel);
        % add tinted frame to tinted_frames_bin
        tinted_frames_bin{frame_index} = tinted_frame;
    end
end

function [windowed_frames_bin] = window_adder (frames_bin)
    % Dictionary to hold windowed frames
    windowed_frames_bin = {};
    % Write frames in order to the video
    for frame_index = 1:1:numel(frames_bin)
        % Define the cross width (1/25th of the image width)
        cross_width = round(size(frames_bin{frame_index}, 2) / 25);
        % Get the size of the original image
        [height, width, ~] = size(frames_bin{frame_index});
        % Create a new image as a copy of the original
        image_with_cross = frames_bin{frame_index};
        % Define the central region for the cross
        center_X = round(width / 2);
        center_Y = round(height / 2);
        % Add horizontal black line
        image_with_cross(center_Y - round(cross_width/2) : center_Y +
round(cross_width/2), :, 1:3) = 0;
        % Add vertical black line
        image_with_cross(:, center_X - round(cross_width/2) : center_X +
round(cross_width/2), 1:3) = 0;
        % Define the border width (1/10th of the image width)
        border_width = round(size(image_with_cross, 2) / 10);
        % Create a new image with a black border
        image_with_border = uint8(zeros(size(image_with_cross, 1) +
2 * border_width, size(image_with_cross, 2) + 2 * border_width,
size(image_with_cross, 3)));
        % Add horizontal black line
        image_with_cross(center_Y - round(cross_width/2) : center_Y +
round(cross_width/2), :, 1:3) = 0;
        % Add vertical black line
        image_with_cross(:, center_X - round(cross_width/2) : center_X +
round(cross_width/2), 1:3) = 0;
        % Copy the original image into the center of the new image
        image_with_border(border_width + 1:end - border_width, border_width
+ 1:end - border_width, :) = image_with_cross;
        % add windowed frame to windowed_frames_bin
        windowed_frames_bin{frame_index} = image_with_border;
    end
end

function [flipped_frames_bin] = left_right_video_flipper (frames_bin)
    % Dictionary to hold flipped frames
```

```
    flipped_frames_bin = {};
    % Write frames in reverse order to the reversed video
    for frame_index = 1:1:numel(frames_bin)
        flipped_frame = fliplr(frames_bin{frame_index});
        % add flipped frame to flipped_frames_bin
        flipped_frames_bin{frame_index} = flipped_frame;
    end
end

function [reversed_frames_bin] = video_reverser (frames_bin)
    % Dictionary to hold flipped frames
    reversed_frames_bin = {};
    rising_frame_index = 1;
    % Write frames in reverse order to the reversed video
    for falling_frame_index = numel(frames_bin):-1:1
        % add reversed order frame to reverse_frames_bin
        reversed_frames_bin{rising_frame_index} =
frames_bin{falling_frame_index};
        rising_frame_index = rising_frame_index + 1;
    end
end

function [final_frames, name_string] = video_mash_up (mash_code,
input_frames)
    intermediate_frames_1 = {}; % Empty all dictionaries at start of each
function call
    intermediate_frames_2 = {};
    intermediate_frames_3 = {};
    intermediate_frames_4 = {};
    intermediate_frames_5 = {};
    intermediate_frames_6 = {};
    intermediate_frames_7 = {};
    intermediate_frames_8 = {};
    final_frames = {};

    intermediate_frames_1 = input_frames;
    name_string = "";
    % apply red tinting?
    if mash_code(1) == 1
        intermediate_frames_2 = colored_video_maker ([1.5, 0, 0],
intermediate_frames_1);
        name_string = name_string + 'red(+)';
    elseif mash_code(1) == 0
        intermediate_frames_2 = intermediate_frames_1;
        name_string = name_string + 'red(-)';
    end
    % apply green tinting?
    if mash_code(2) == 1
        intermediate_frames_3 = colored_video_maker ([0, 1.5, 0],
intermediate_frames_2);
```

```
        name_string = name_string + 'grn(+)';
    elseif mash_code(2) == 0
        intermediate_frames_3 = intermediate_frames_2;
        name_string = name_string + 'grn(-)';
    end
    % apply blue tinting?
    if mash_code(3) == 1
        intermediate_frames_4 = colored_video_maker ([0, 0, 1.5],
intermediate_frames_3);
        name_string = name_string + 'blu(+)';
    elseif mash_code(3) == 0
        intermediate_frames_4 = intermediate_frames_3;
        name_string = name_string + 'blu(-)';
    end
    % apply purple tinting?
    if mash_code(4) == 1
        intermediate_frames_5 = colored_video_maker ([1.5, 0, 1.5],
intermediate_frames_4);
        name_string = name_string + 'prp(+)';
    elseif mash_code(4) == 0
        intermediate_frames_5 = intermediate_frames_4;
        name_string = name_string + 'prp(-)';
    end
    % apply horizontal flip?
    if mash_code(5) == 1
        intermediate_frames_6 = left_right_video_flipper
(intermediate_frames_5);
        name_string = name_string + 'flp(+)';
    elseif mash_code(5) == 0
        intermediate_frames_6 = intermediate_frames_5;
        name_string = name_string + 'flp(-)';
    end
    % apply reversal?
    if mash_code(6) == 1
        intermediate_frames_7 = video_reverser (intermediate_frames_6);
        name_string = name_string + 'rev(+)';
    elseif mash_code(6) == 0
        intermediate_frames_7 = intermediate_frames_6;
        name_string = name_string + 'rev(-)';
    end
    % apply window?
    if mash_code(7) == 1
        intermediate_frames_8 = window_adder (intermediate_frames_7);
        name_string = name_string + 'win(+)';
    elseif mash_code(7) == 0
        intermediate_frames_8 = intermediate_frames_7;
        name_string = name_string + 'win(-)';
    end
    final_frames = intermediate_frames_8;
end
```

```
function [] = video_maker_namer (frames_bin, label_string)
    % Create a VideoWriter object to construct reversed video
    video_object = VideoWriter('chambered_nautilus_'+label_string+'.avi',
'Uncompressed AVI');
    open(video_object);
    % Write frames in reverse order to the reversed video
    for frame_index = 1:1:numel(frames_bin)
        % add windowed frame to video
        writeVideo(video_object, frames_bin{frame_index});
    end
    % Close the reversed video
    close(video_object);
    disp(["A new modified .avi clip is now in the working folder"]);
end

function [] = video_annotate_function(ground_truth, truth_label,
video_to_label, output_name, score_cutoff)
    % Create a folder to store the frames if not already constructed
    extracted_frames_folder = 'OUTPUT';
    if ~exist(extracted_frames_folder, 'dir')
        mkdir(extracted_frames_folder);
    end
    % load ground truth table
    truth = selectLabels(ground_truth, truth_label);
    % extract training data for detector from truth table
    trainingData = objectDetectorTrainingData(truth);
    % construct detector using training data
    detector = trainACFObjectDetector(trainingData, 'Numstages', 5);
    % load input video file to be labeled into video reader
    loaded_video = VideoReader(video_to_label);
    % create video player object
    video_player = vision.DeployableVideoPlayer;

    % relative path to output directory folder, "OUTPUT"
    workingDir = './OUTPUT/'
    % declare output .avi video file and prepare to add video frames to it
to build
    %   final video
    outputVideo = VideoWriter(fullfile(workingDir, output_name + ".avi"));
    outputVideo.FrameRate = loaded_video.FrameRate;
    open(outputVideo)

    while hasFrame(loaded_video)
        % load input video frames one at a time
        frame = readFrame(loaded_video);
        % use detector and ground truth to locate possible matching objects
        [bbox,score] = detect(detector,frame);
        % assign probability of match rank to possible matching objects
        % retain only those match rank probabilities above score cut off
```

```
        %   provided
        bbox = bbox(score > score_cutoff,:);
        score = score(score > score_cutoff,:);
        [selectedBbox,selectedScore] = selectStrongestBbox(bbox,score,
OverlapThreshold=0.1);

        numBoxes = size(selectedBbox,1);
        % annotate possible matching objects
        str = "OBJECTS DETECTED : " + numBoxes;
        img = insertObjectAnnotation(frame,"rectangle",
selectedBbox,truth_label + " : " + selectedScore);
        img = insertText(img,[250 550],str,TextColor=[1 1 0]);

        % display each annotated video frame in video player object
        step(video_player, img)
        % write each annotated video frame to declared output .avi video file
        writeVideo(outputVideo,img)
    end
    close(outputVideo)
    disp(["A new annotated .avi clip is now in the folder, 'OUTPUT'"]);
end
```

SECTION 5: CONCLUSION

RETURN TO TABLE OF CONTENTS

Overall, the goals of the project seem to be met. Frame-by-frame video manipulation and its impact on subsequent video annotation/labeling using ground truth were both explored. Results suggest that shorter wavelength tinting (blue) and video windowing as combined modifications had the largest negative impact on successful video annotation. Separate modifications of long wavelengths (red tinting) or video windowing did not have as large an impact on successful video annotation.

A search of indeed.com and other job boards suggests the need to learn both Matlab and Python for image/video processing and computer vision. In light ot this, given additional time, this project would have been replicated using Python OpenCV.

SECTION 6: REFERENCES

RETURN TO TABLE OF CONTENTS

https://matlabacademy.mathworks.com/details/computer-vision-onramp/orcv

MathWorks | Training Services

Course Completion Certificate

Ian Morgan-Graham

has successfully completed **100%** of the self-paced training course

Computer Vision Onramp

DIRECTOR, TRAINING SERVICES

3 February 2024

https://www.youtube.com/watch?v=ow_B_30WU1s&t=727s&ab_channel=MATLAB

https://www.youtube.com/watch?v=UnXDQmjYvDk&t=414s&ab_channel=MATLAB

https://www.mathworks.com/help/vision/ug/get-started-with-the-video-labeler.html

https://www.mathworks.com/help/matlab/import_export/convert-between-image-sequences-and-video.html

https://chat.openai.com

https://www.mathworks.com/matlabcentral/answers/index

Kiel, S., Goedert, J. L., & Tsai, C. (2022). Seals, whales and the Cenozoic decline of nautiloid cephalopods. *Journal of Biogeography*, *49*(11), 1903–1910. https://doi.org/10.1111/jbi.14488

YouTube. (2023, March 28). *Nautiloids thrived for 500 million years until these guys showed up*. YouTube. https://www.youtube.com/watch?v=3vQ55ToQeWI

Wikimedia Foundation. (2023, November 3). *Chambered nautilus*. Wikipedia. https://en.wikipedia.org/wiki/Chambered_nautilus#cite_note-6