Automatic Recognition of Fish Diseases in Fish Farms

By:

Ahmed Waleed

Hadeer Medhat

Mariam Esmail

kareem Osama

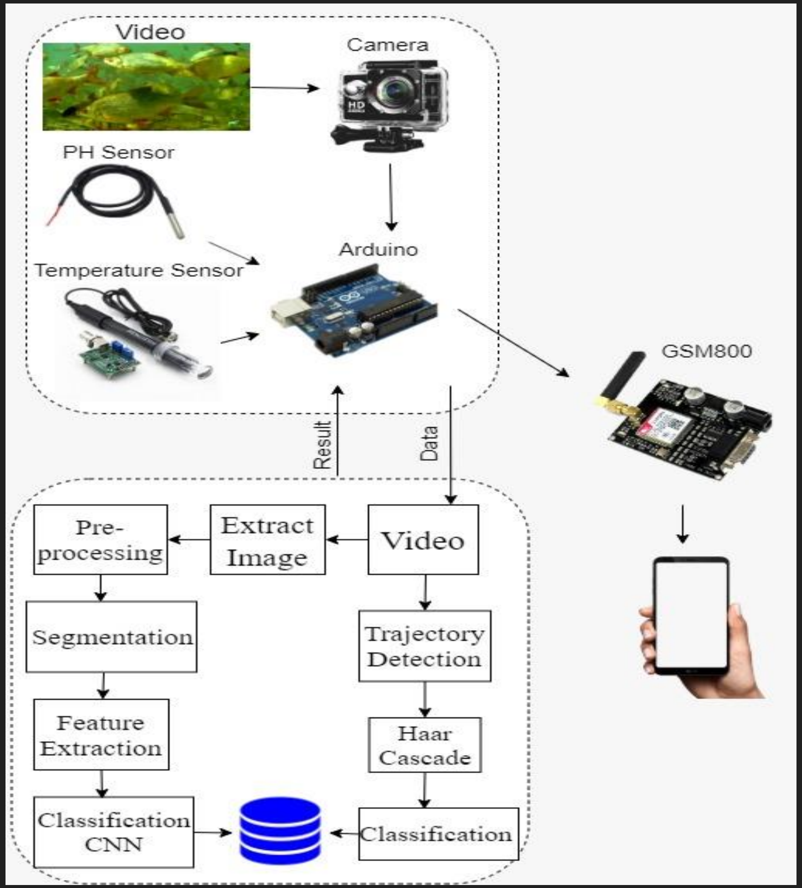
Supervised By: DR.Taraggy M Ghani
Eng.Radwa Samy

Introduction

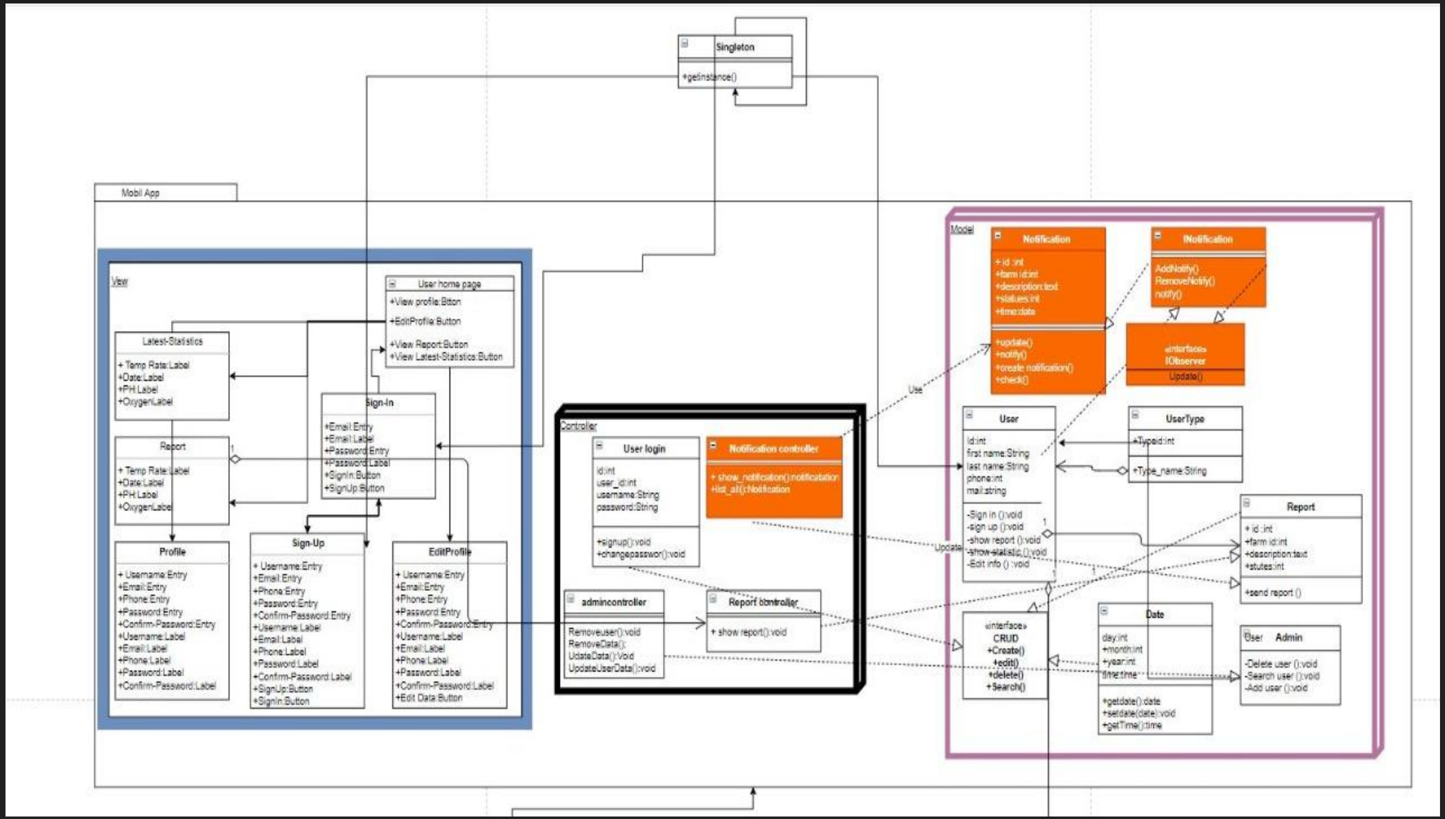
- The purpose of this Software design document is to describe the architecture and system design for our Automatic recognition of Fish Diseases in Fish Farms system. Our system mainly detects and diagnose fish diseases in fish farms au-tomatically and examine water quality

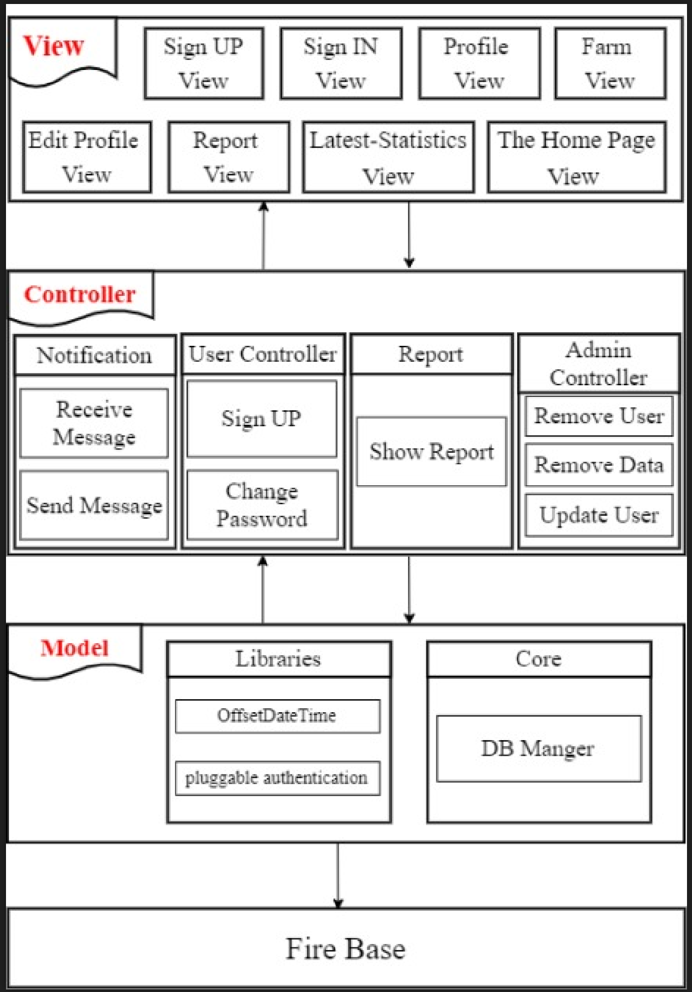
Overview

- Our system is created to performs detection and identification of fish diseases and analyze fish abnormal behavior. Its goal is to prevent and control spreading of diseases by early identification and diagnosis

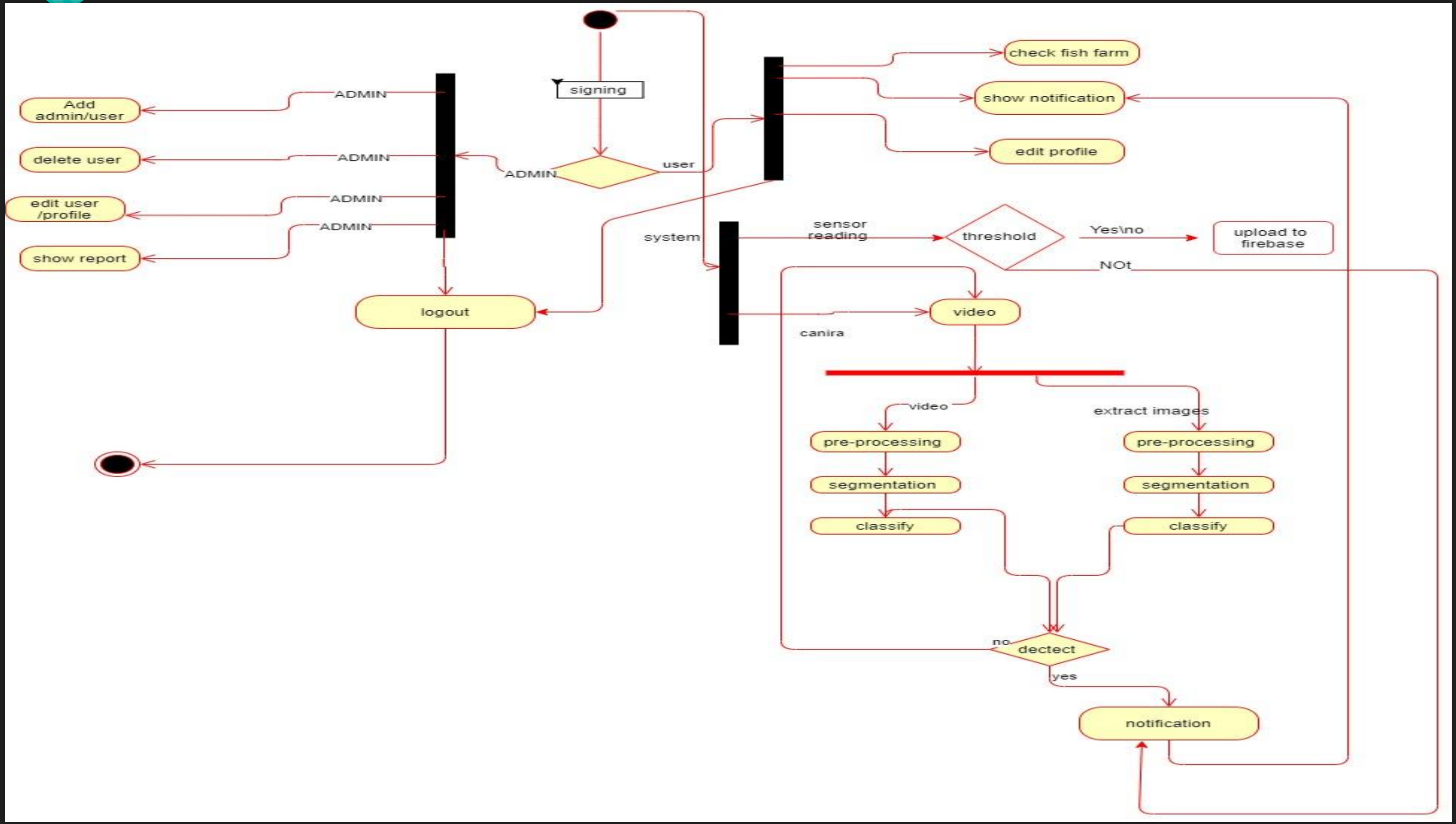


Architectural Design

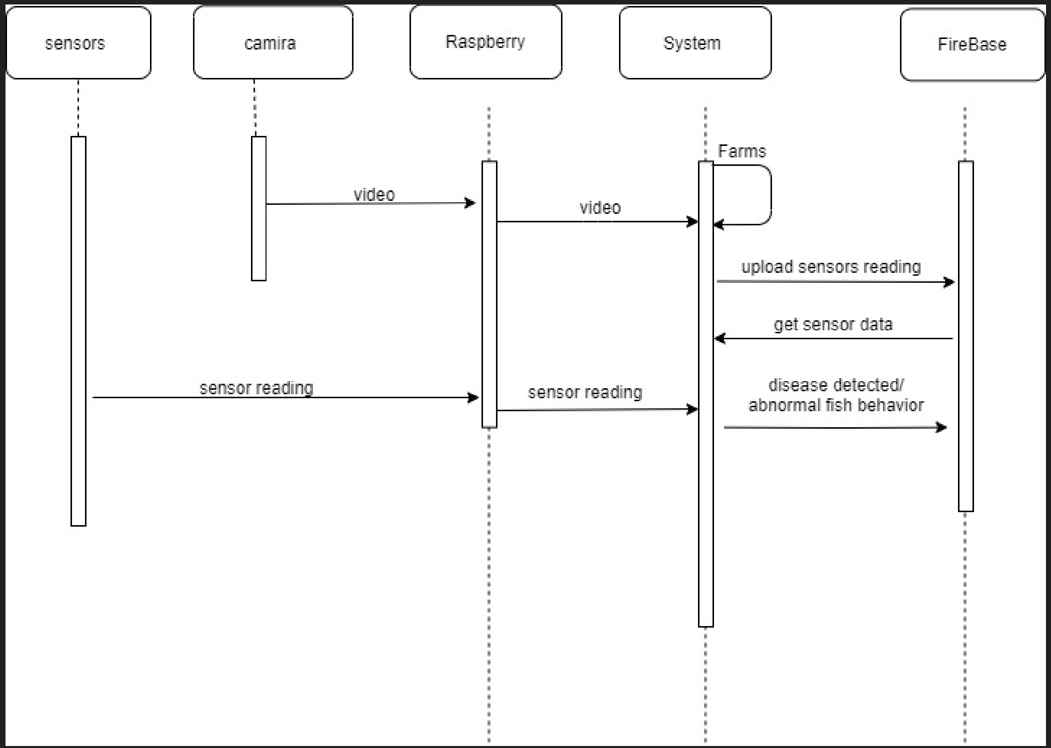


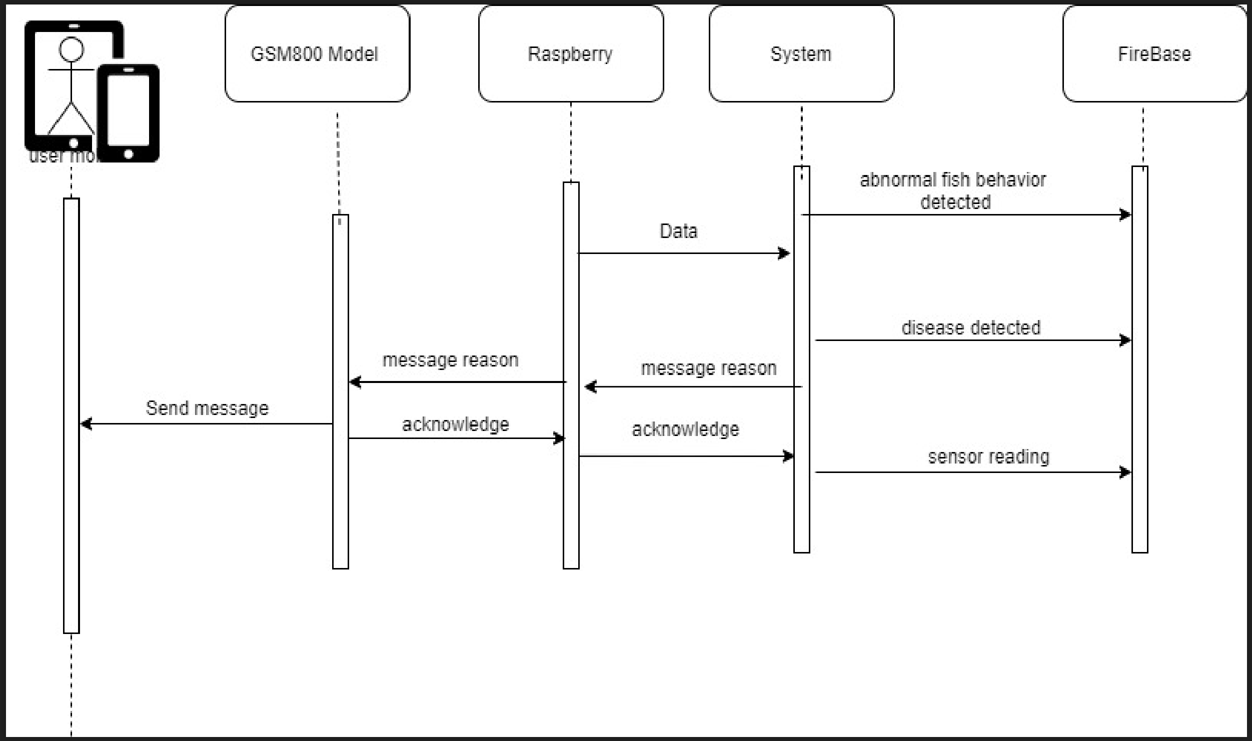


Activity Diagram



Sequence Diagram

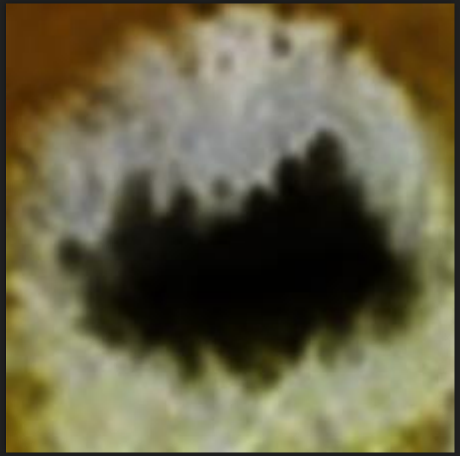




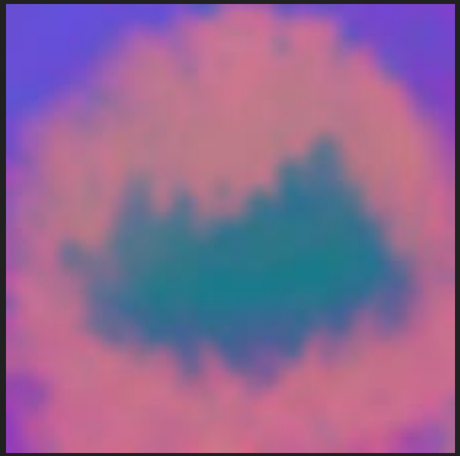
Component Design

1)Pre-processing

RGB



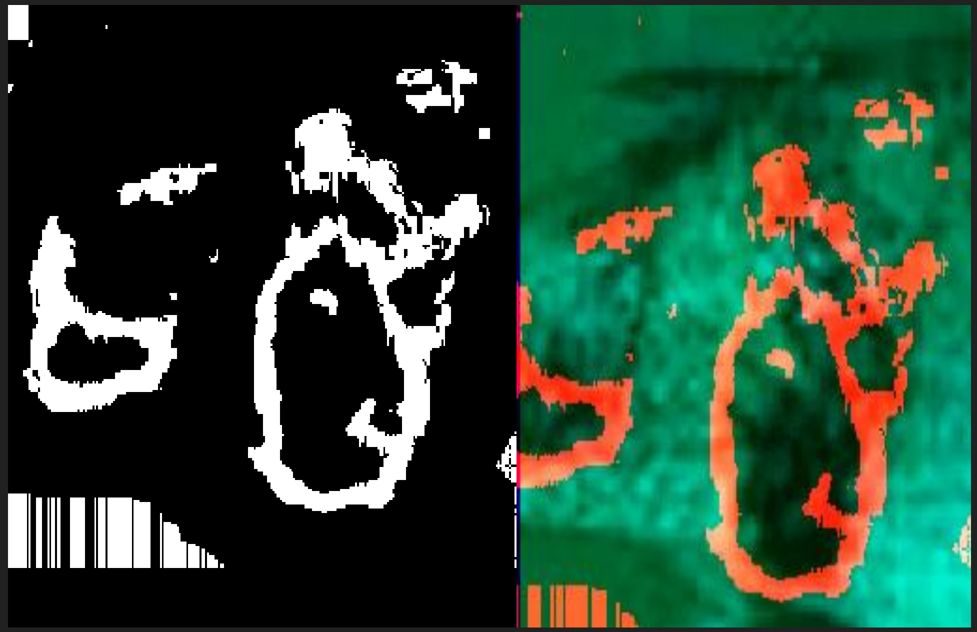
YCBCR



XYZ

Color-Based Gaussian distribution model

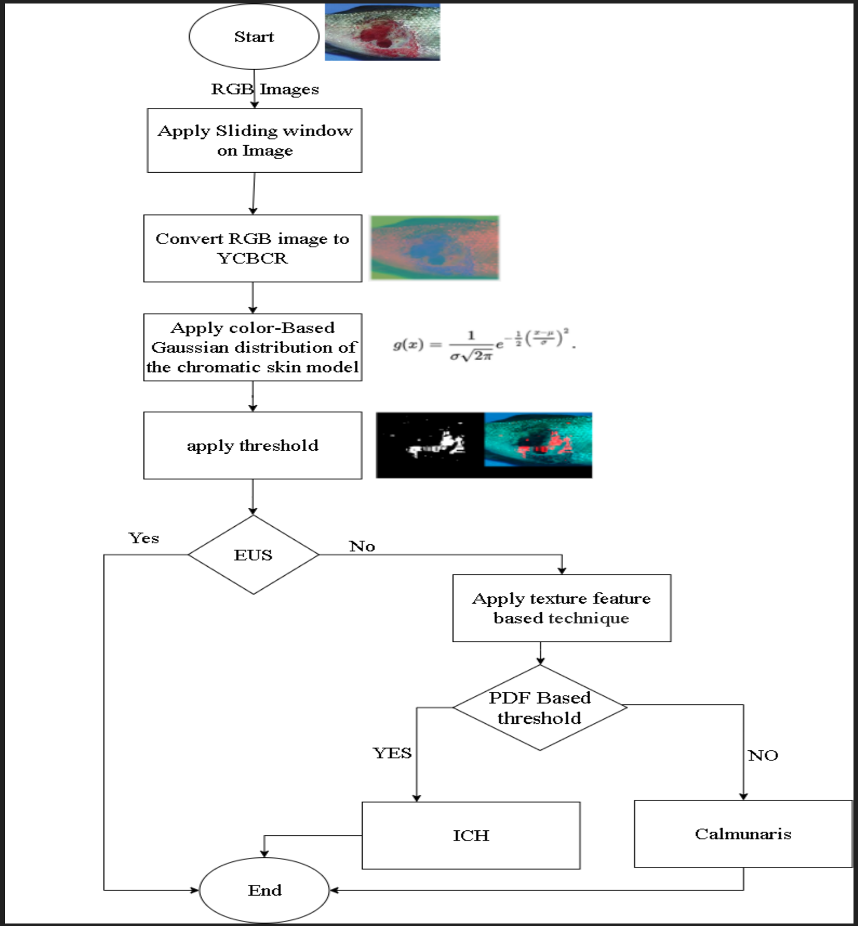
- Convert Image to YCBCR
- Get CB and CR
- Get mean and covariance
- Build gaussian distribution model
- Apply threshold

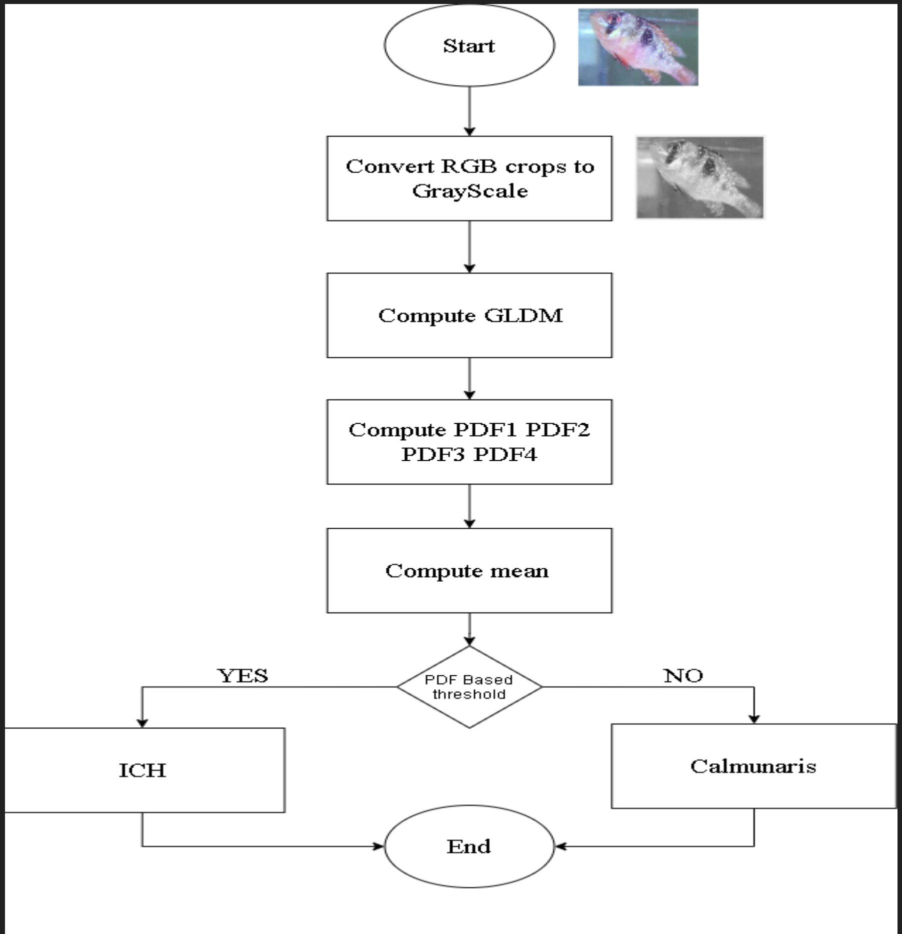


Texture Feature Based

Convert RGB image to Grayscale → Compute GLDM and get PDF1 PDF2 PDF3 PDF4 → Compute Mean → Apply threshold

Multi staging hybrid segmentation technique





CNN Classification

ResNet
VGG
Sequeeze net
AlexNet
Google Net
MobileNetv2
Xception
ShuffleNet
DenseNet
Inceptionv3

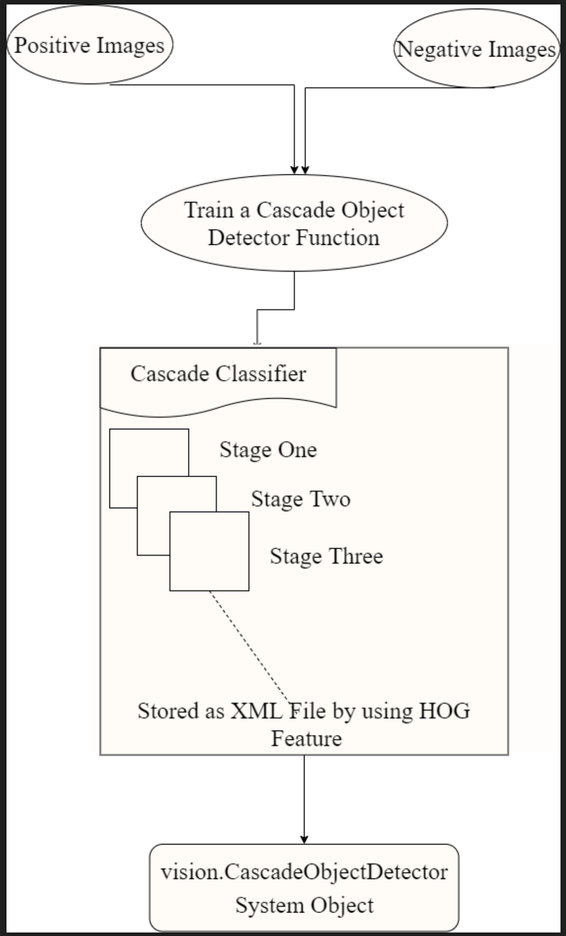
Model	Versions	Layers	Activation function	Input size	Convolution kernel size	Innovative point	Applications
ResNet ??	18 50 101	18 layers 50 Layers 101 Layers	RelU	224x224x3	ResNet initial convolution: 7x7. Resnet 50 and 101: 1x1, 3x3 and 1x1. ResNet 18: 3x3	ResNet overcomes degradation and vanishing gradient problems residual blocks that increases the number of hidden layers. The core idea of ResNet is introducing "identity shortcut connection" that skips one or more layers	ImageNet 2012 dataset ?? and Automatic Spoofing Detection [1]
VGG [2]	16 19	16 layers 19 layers	ReLU non-linear	224x224x3	VGG: 3x3	This network uses only 3x3 convolutional layers stacked on top of each other in increasing depth. VGG makes improvement over AlexNet by replacing large kernel-sized	ImageNet (ILSVRC-2012) and Classifying Cooking Object's State [3]
SqueezeNet [4]		18 layers	RelU	227x227x3	SqueezeNet: 1x1 and 3x3	SqueezeNet goal was to create small neural network with fewer parameters. It was able to achieve a 50X reduction in model size compared to AlexNet. SqueezeNet has advantage of fire module, which uses less filters to decrease number of parameters	ImageNet dataset, fine-grained object recognition [5] and Generic Visual Recognition [6]
DenseNet [7]	201	201 layers	ReLU	224x224x3	DenseNet: 7x7	In DenseNet, the feature-maps of all preceding layers are used as inputs, and its own feature-maps are used as inputs into all subsequent layers. DenseNet alleviate the vanishing-gradient problem and reduce the number of parameters	ImageNet (ILSVRC-2012) [8] and optical flow []
AlexNet [9]		8 layers	ReLU non-linear	227x227x3	Alexnet: 11x11, 5x5 and 1x1	AlexNet has large number of filters to perform the convolution operation of sizes 11×11, 5×5 and 3×3	ImageNet(LSVRC-2010) and high Spatial Resolution Remote Sensing [10]
GoogleNet [11]		22 layers	ReLU	224x224x3	GoogleNet: 5X5, 3X3 and 1X1	GoogleNet increases the depth of the network and gain a higher performance level. It is based on the concept of the inception module, it is the collection of convolution and pooling operation performed in a parallel manner so that features can be extracted using different scales	ImageNet(ILSVRC14) and High performance offline handwritten Chinese character recognition [12]
Mobilenetv2 [13]	2	54 layers	ReLU6	224x224x3	Mobilenetv2: 1x1, 3x3	Mobilenetv2 improves the performance of mobile models on multiple tasks. Mobilenetv2 is based on an inverted residual structure	ImageNet, Face Attribute Detection [14]
Xception [15]		71 layers	ReLU	299x299x3	Xception: 1x1,3x3	Xception involves Depthwise Separable Convolutions,it is supposed to be more efficient than classical convolution in terms of computation time. Xception relies on Shortcuts between Convolution blocks	ImageNet datase [16], Audio Event Detection and Tagging [17]
Inceptionv3 [18]	3	48 layers	ReLU	299x299x3	Inceptionv3: 3x3	In inceptionv3, computational efficiency and fewer parameters are realized	ImageNet(ILSVRC 2012) and flower classification [19]
ShuffleNet [20]			RelU	224x224x3	Shufflenet: 1x1,3x3	Shufflenet aim to explore a highly efficient architecture specially designed for limited computing ranges. Shufflenet allows more feature map channels and it is especially critical to the performance of very small networks. ShuffleNet achieves 13× actual speedup over AlexNet while maintaining comparable accuracy	ImageNet, Mobile devices [21] .

CNN Classification Result

Cnn Architectures	RGB(0.001)	RGB(0.01)	RGB(0.1)	YCBCR(0.001)	YCBCR(0.01)	YCBCR(0.1)	XYZ(0.001)	XYZ(0.01)	XYZ(0.1)
ResNet18	17,60%	14,50%	24,60%	13,60%	12,50%	12,50%	17,70%	18,50%	15,50%
ResNet50	18,70%	18,60%	15,60%	14,60%	15,50%	15,50%	24,50%	16,50%	16,50%
ResNet101	16,**80**%	14,%	16,50%	13,50%	14,50%	14,50%	15,50%	16,50%	19,50%
Alex-Net	12,60%	7,50%	9,60%	7,60%	9,50%	9,50%	8,50%	8,50%	8,60%
VGG16	14,50%	14,60%	14,50%	15,50%	18,40%	18,50%	16,60%	16,50%	15,50
VGG19	14,50%	14,60%	15,50%	21,50%	17,60%	14,60%	16,50%	15,50%	15,50%
Mobilenetv2	13,50%	7,60%	8,50%	17,60%	7,40%	7,40%	10,60%	9,60%	9,60%
Xception	12,50%	12,50%	11,50%	11,50%	16,60%	13,50%	13,50%	12,50%	12,60%
Inceptionresnetv2	11,50%	16,60%	12,60%	12,60%	11,50%	11,60%	15,50%	12,50%	12,60%
Shufflenet	7,50%	7,60%	7,50%	7,60%	7,50%	8,50%	21,50%	19,60%	15,60%
Nasnetmobile	9,50%	7,50%	7,60%	7,50%	7,50%	8,60%	15,60%	15,60%	15,60%
Nasnetlarge	13,60%	13,50%	15,50%	14,50%	14,60%	13,60%	14,50%	14,60%	14,60%
Squeezenet	7,50%	7,60%	7,50%	7,50%	7,50%	8,50%	8,60%	11,50%	8,50%
Inceptionv3	15,50%	14,40%	12,50%	15,60%	11,60%	11,50%	14,50%	16,50%	13,50%
Densenet201	**6.53**,60%	13,50%	7,60%	16,60%	7,50%	10,60%	12,60%	10,50%	7,60%
Googlenet	15,60%	14,50%	14,50%	14,50%	14,50%	16,50%	20,50%	16,50	19,50%

Train Cascade Object Detector

- The vision.CascadeObjectDetector System object detects objects in images by sliding a window over the image.
- The detector then uses a cascade classifier to decide whether the window contains the object of interest.

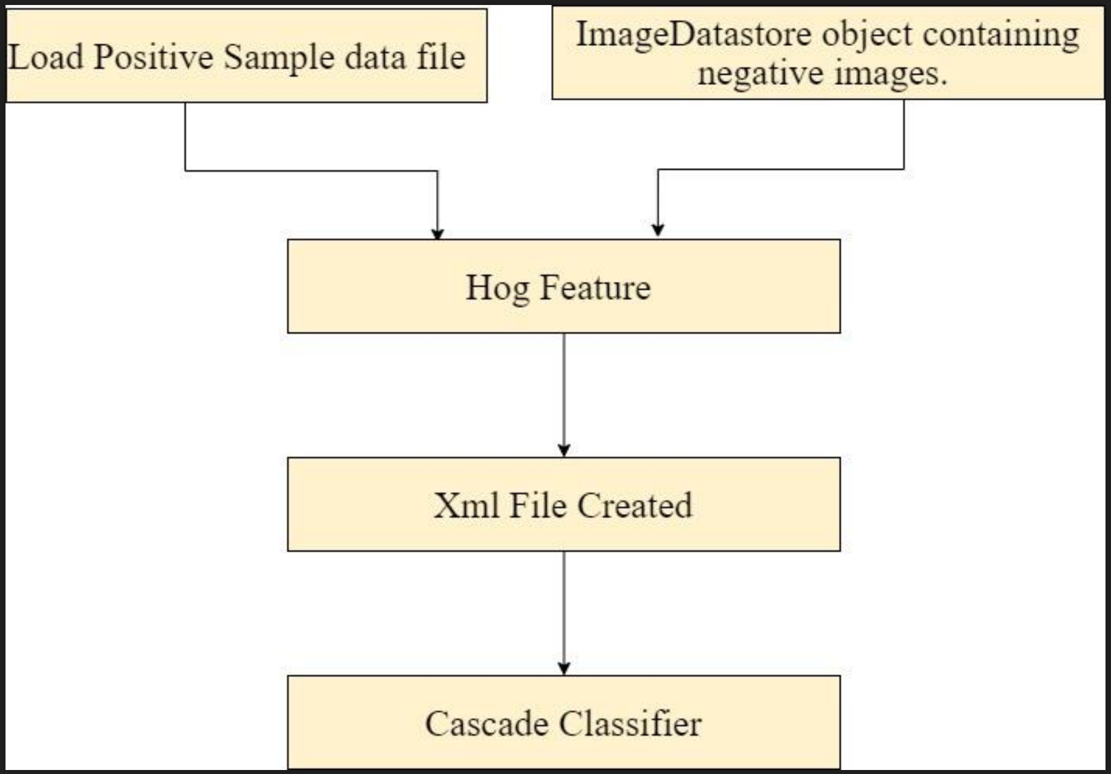


XML Creation

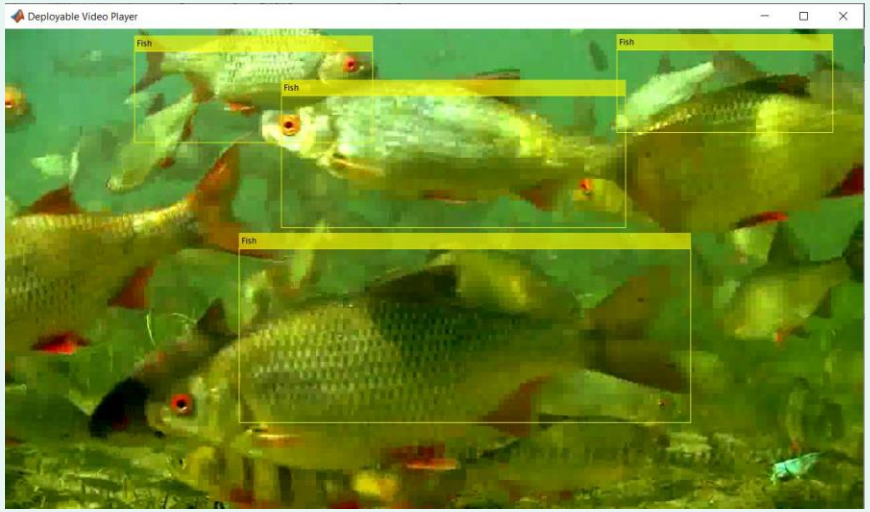
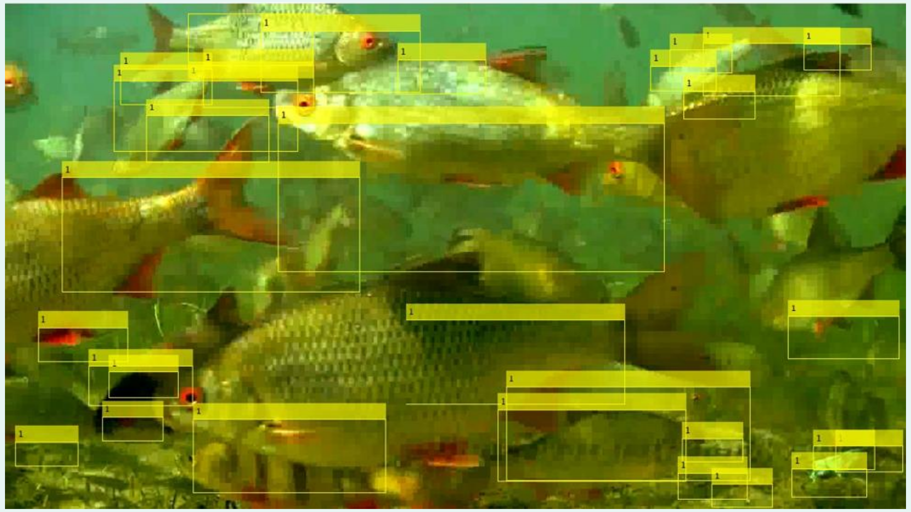
- Load the positive samples data from file.
- Create an imageDatastore object containing negative images.
- Choose the feature that suits the type of object detection.

(The HOG features are often used to detect objects) .

- Then run trainCascadeObjectDetector to create xml file

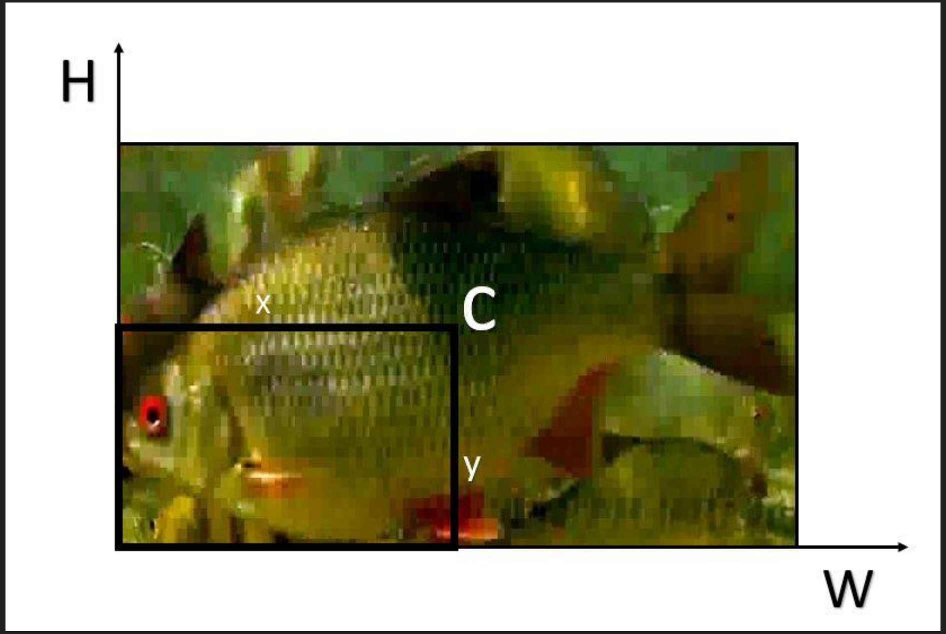


Compare

HOG Feature	Haar Feature
HOG features are often used to detect objects such as people and cars. They are useful for capturing the overall shape of an object.	Haar features are often used to detect faces because they work well for representing fine-scale textures.
	

Video Trajectory

Matrix Creation



matrix2

181x2 double

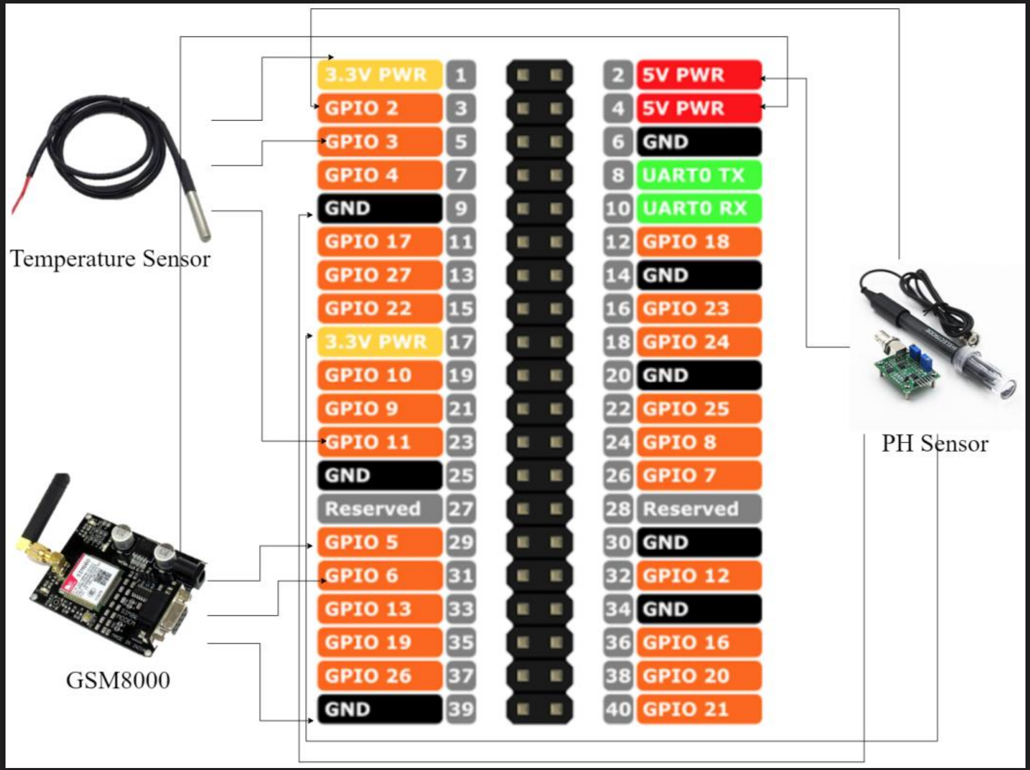
	1	2	3
1	372	102.5000	
2	1.0745e+03	94	
3	670	199	
4	687	458	
5	458.5000	104.5000	
6	682.5000	197	
7	578	526	
8	288	331.5000	
9	428.5000	105.5000	
10	279	138.5000	
11	719	191	

Tracking-Scenario

- Time corresponding to arrival at each waypoint in seconds, The first elements of Time Of Arrival must be 0.
- Waypoints: calculate the position for each bonding box in the matrix
- Velocity in the navigation coordinate system at each way point in meters per second, specified as an N-by-3 matrix

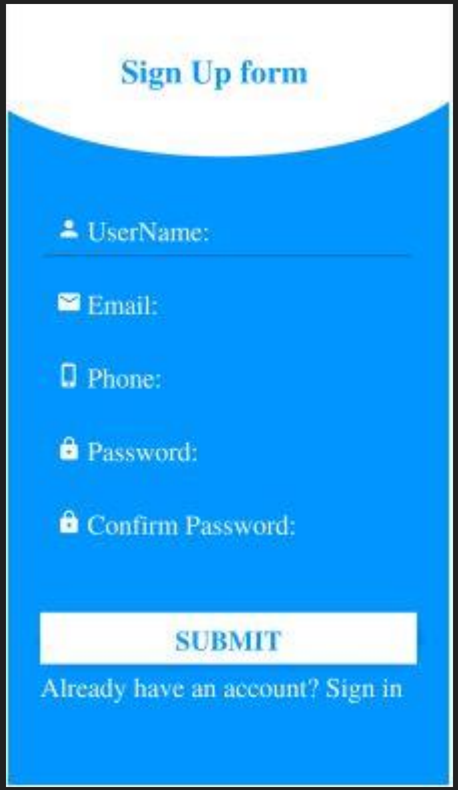
1 Time	2 Position			3 Velocity		
0	-10.0800	202.6500	33.9418	-11.2800	8.8100	-0.0588
1.9608	-21.1200	211.2700	33.8846	-11.2500	8.7900	-0.0577
2.9412	-32.1300	219.8800	33.8287	-11.2100	8.7800	-0.0565
3.9216	-43.1000	228.4900	33.7738	-11.1600	8.7800	-0.0554
4.9020	-54	237.1000	33.7201	-11.0900	8.7900	-0.0543
5.8824	-64.8500	245.7300	33.6674	-11.0300	8.8100	-0.0531
6.8627	-75.6200	254.3800	33.6158	-10.9500	8.8400	-0.0520
7.8431	-86.3100	263.0700	33.5654	-10.8600	8.8800	-0.0510
8.8235	-96.9100	271.8000	33.5159	-10.7600	8.9300	-0.0499
9.8039	-107.4100	280.5800	33.4676	-10.6600	8.9900	-0.0488
10.7843	-117.8100	289.4300	33.4202	-10.5500	9.0500	-0.0478
11.7647	-128.0900	298.3400	33.3739	-10.4200	9.1300	-0.0467
12.7451	-138.2500	307.3200	33.3286	-10.2900	9.2100	-0.0457
13.7255	-148.2700	316.3900	33.2843	-10.1500	9.2900	-0.0447
14.7059	-158.1500	325.5400	33.2410	-10	9.3800	-0.0437
15.6863	-167.8800	334.7900	33.1987	-9.8400	9.4800	-0.0427
16.6667	-177.4500	344.1400	33.1574	-9.6700	9.5900	-0.0417
17.6471	-186.8500	353.5900	33.1170	-9.5000	9.6900	-0.0407
18.6275	-196.0600	363.1500	33.0776	-9.3100	9.8100	-0.0397
19.6078	-205.1000	372.8200	33.0391	-9.1100	9.9200	-0.0388
20.5882	-213.9300	382.6100	33.0016	-8.9100	10.0400	-0.0378
21.5686	-222.5500	392.5100	32.9649	-8.6900	10.1600	-0.0369
22.5490	-230.9600	402.5400	32.9292	-8.4600	10.2800	-0.0360
23.5294	-239.1400	412.6800	32.8944	-8.2300	10.4100	-0.0351
24.5098	-247.0900	422.9400	32.8604	-7.9800	10.5300	-0.0342

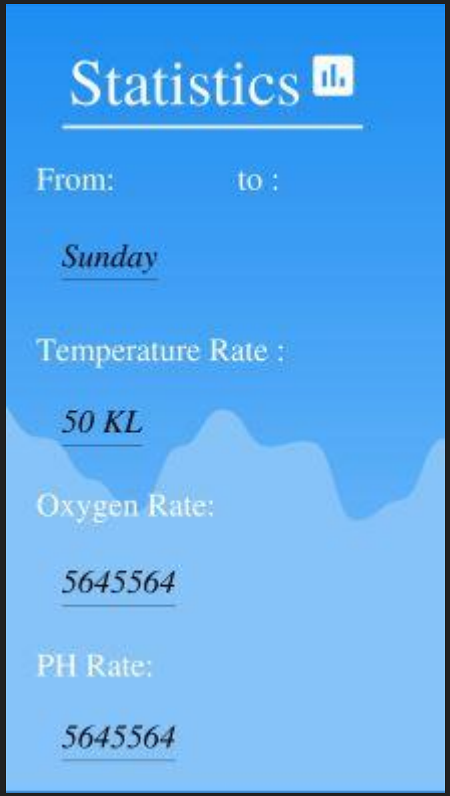
Hardware



Temperature Sensor
GSM8000
PH Sensor
3.3V PWR 1
GPIO 2 3
GPIO 3 5
GPIO 4 7
GND 9
GPIO 17 11
GPIO 27 13
GPIO 22 15
3.3V PWR 17
GPIO 10 19
GPIO 9 21
GPIO 11 23
GND 25
Reserved 27
GPIO 5 29
GPIO 6 31
GPIO 13 33
GPIO 19 35
GPIO 26 37
GND 39
2 5V PWR
4 5V PWR
6 GND
8 UART0 TX
10 UART0 RX
12 GPIO 18
14 GND
16 GPIO 23
18 GPIO 24
20 GND
22 GPIO 25
24 GPIO 8
26 GPIO 7
28 Reserved
30 GND
32 GPIO 12
34 GND
36 GPIO 16
38 GPIO 20
40 GPIO 21

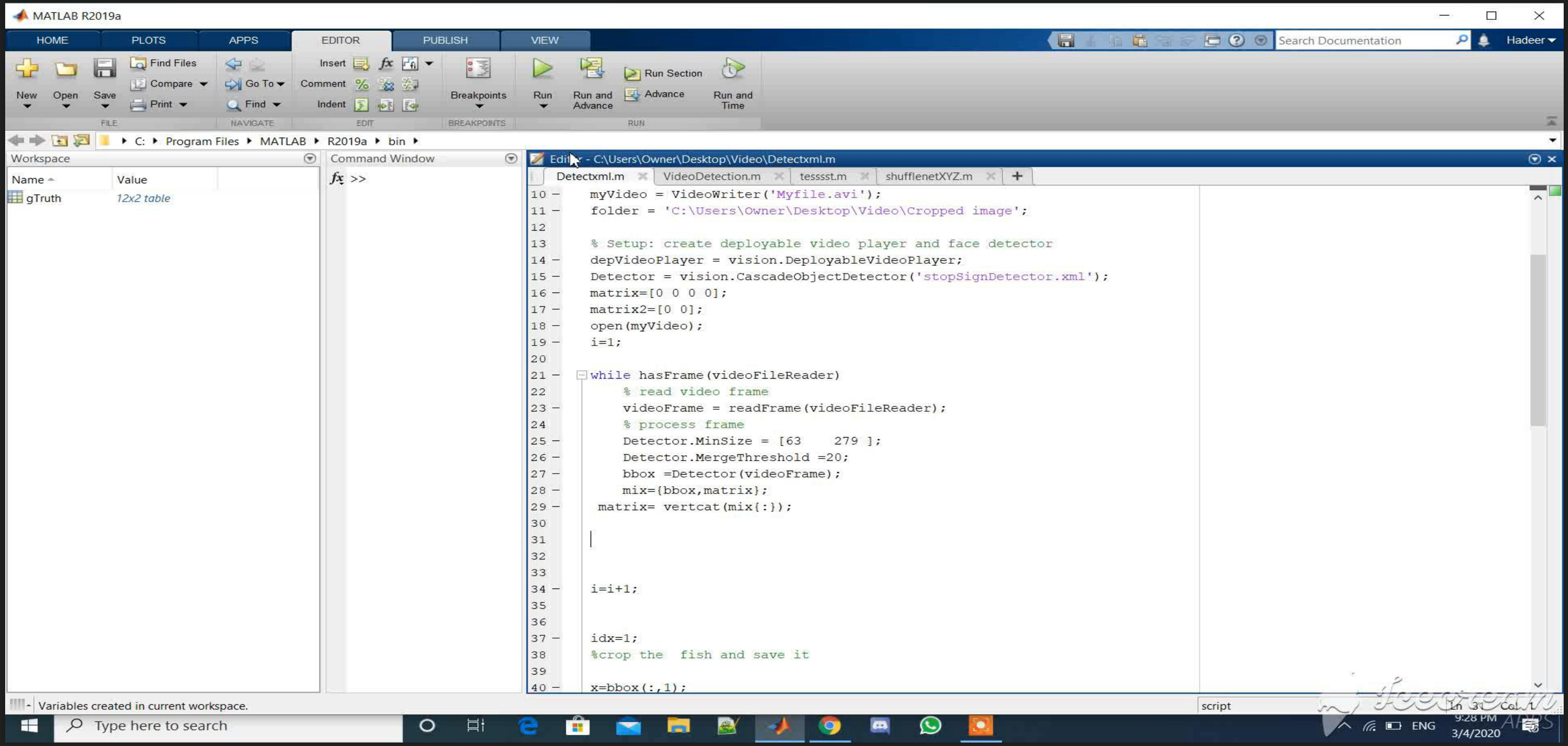
Human Interface Design





Demo

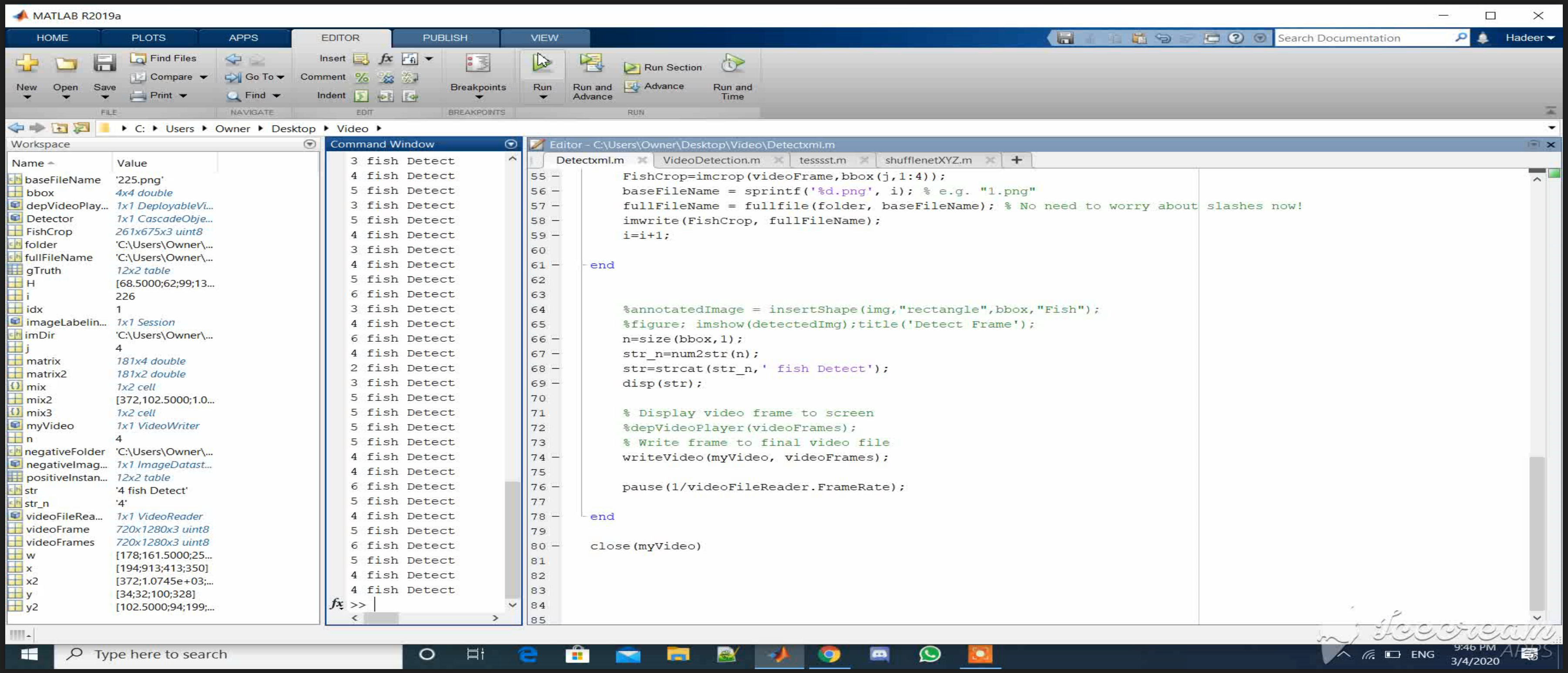
Detect and cropped Fish



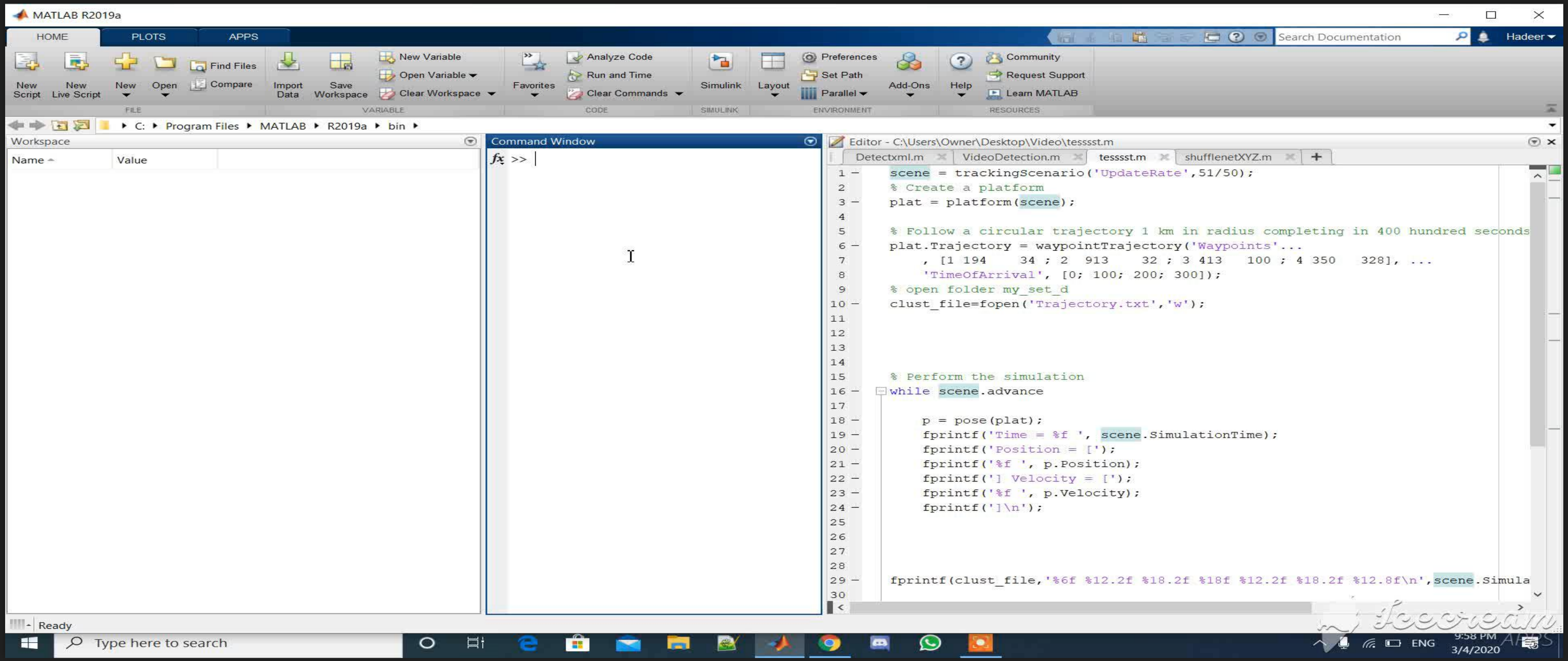
Sample of crop Images



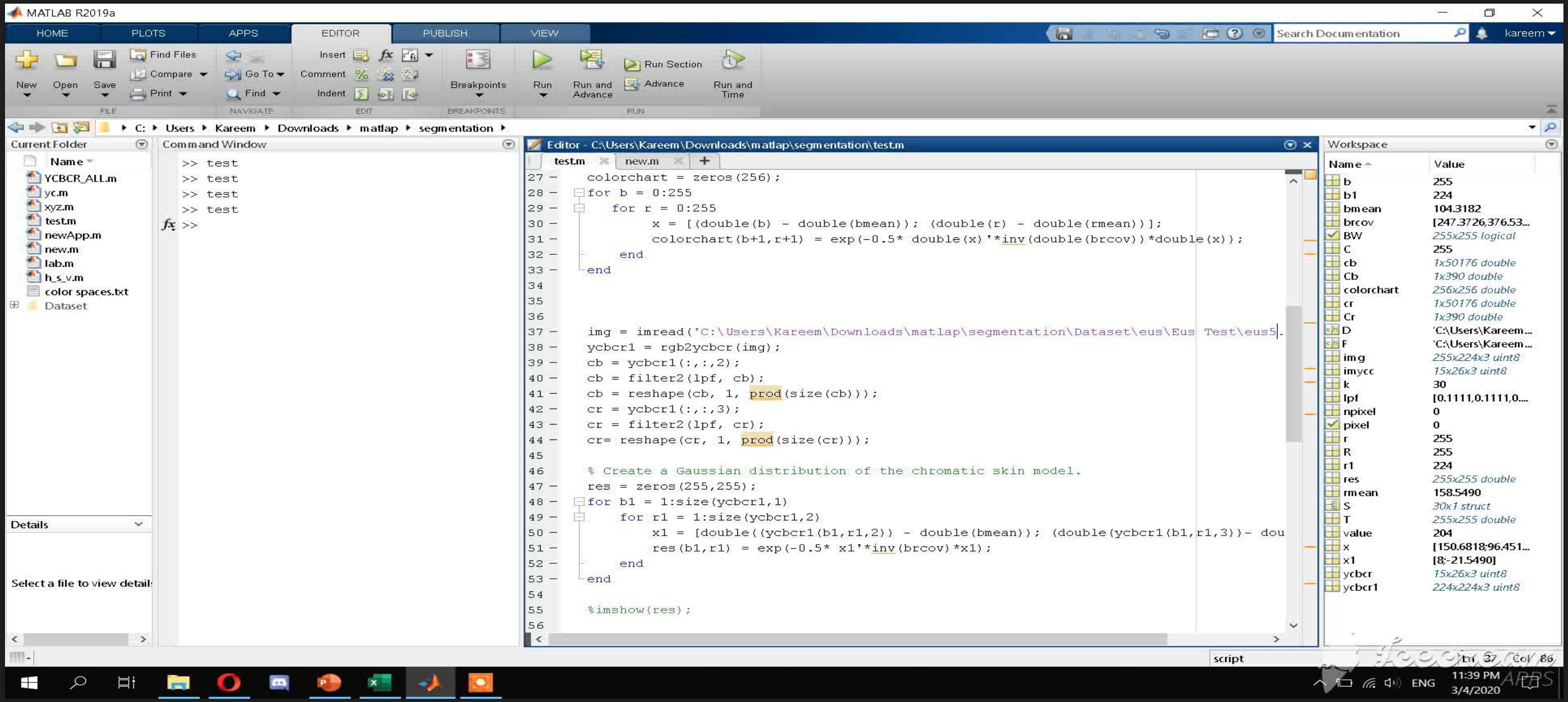

Matrix Calculation



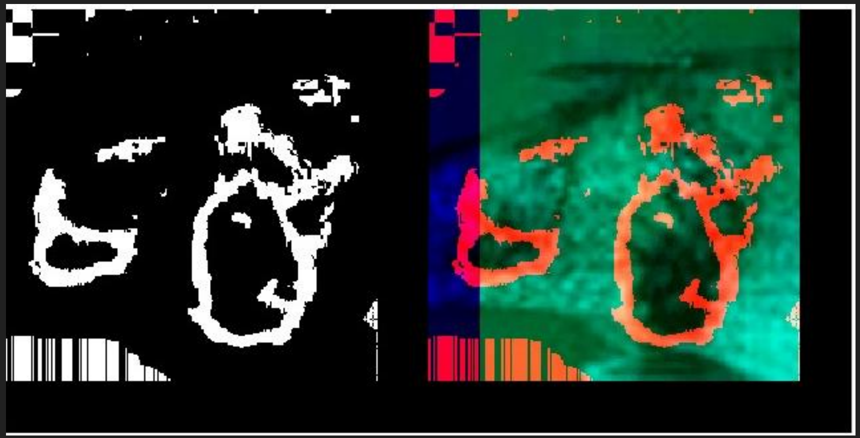
Tracking-Scenario

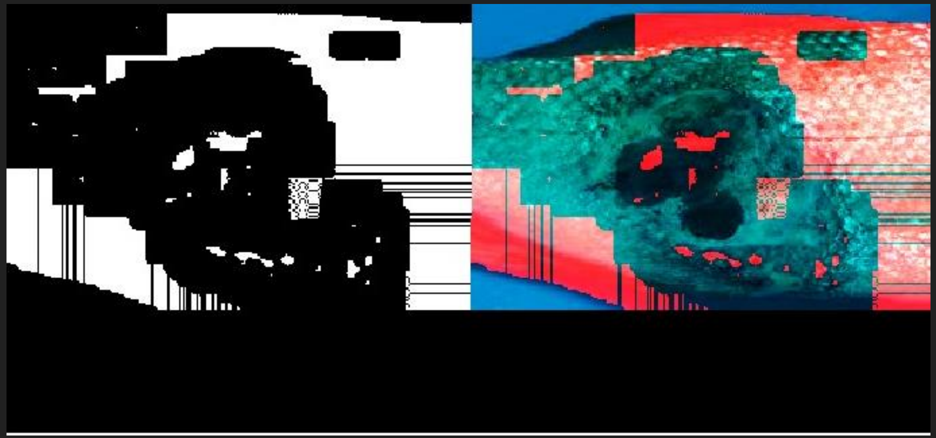


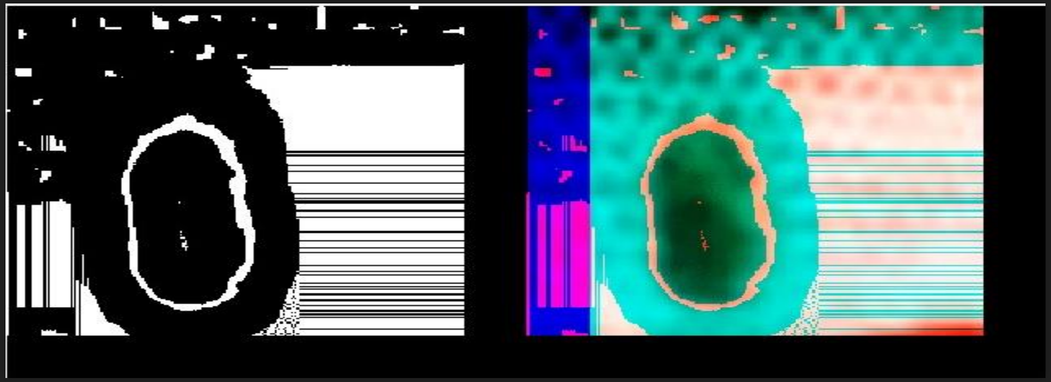
Segmentation

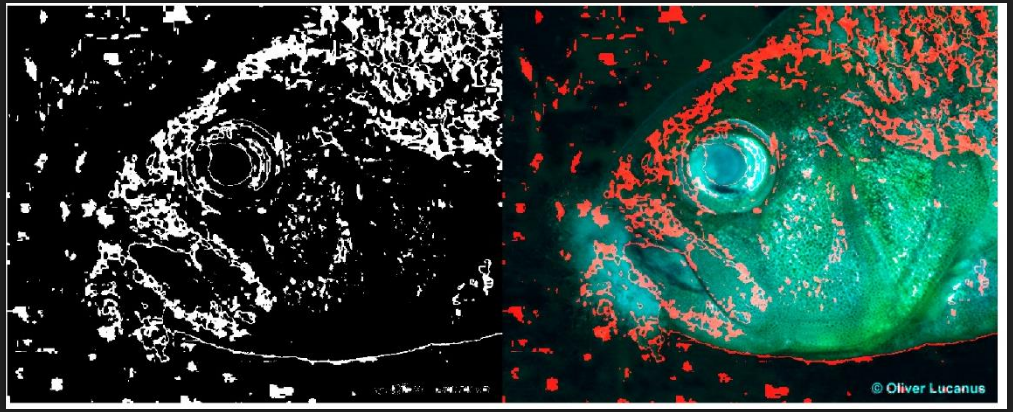
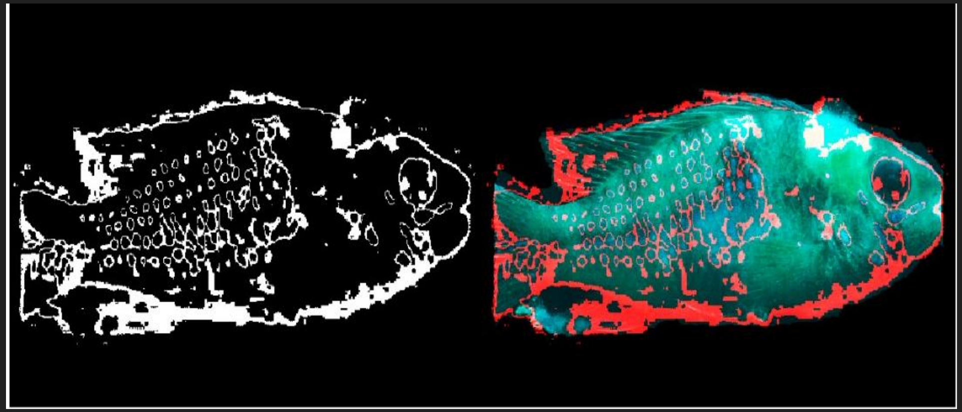
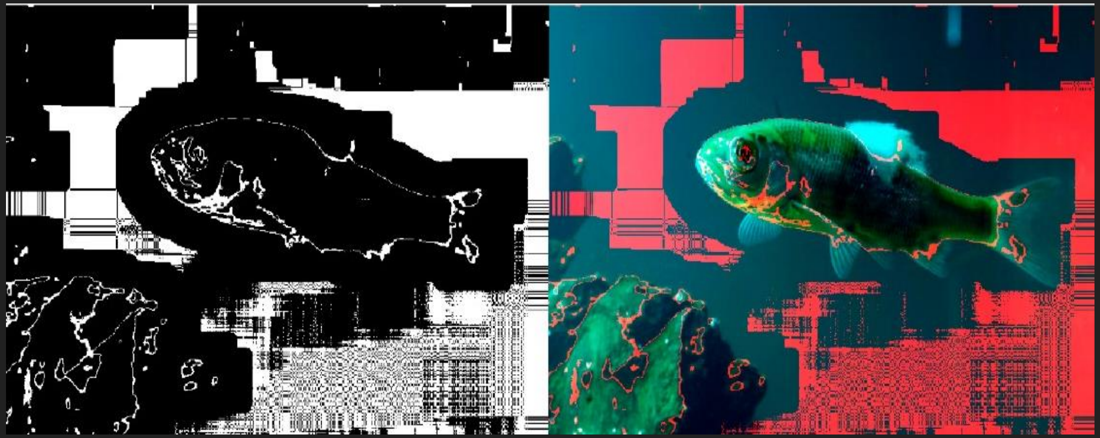


Sample for Segmentation

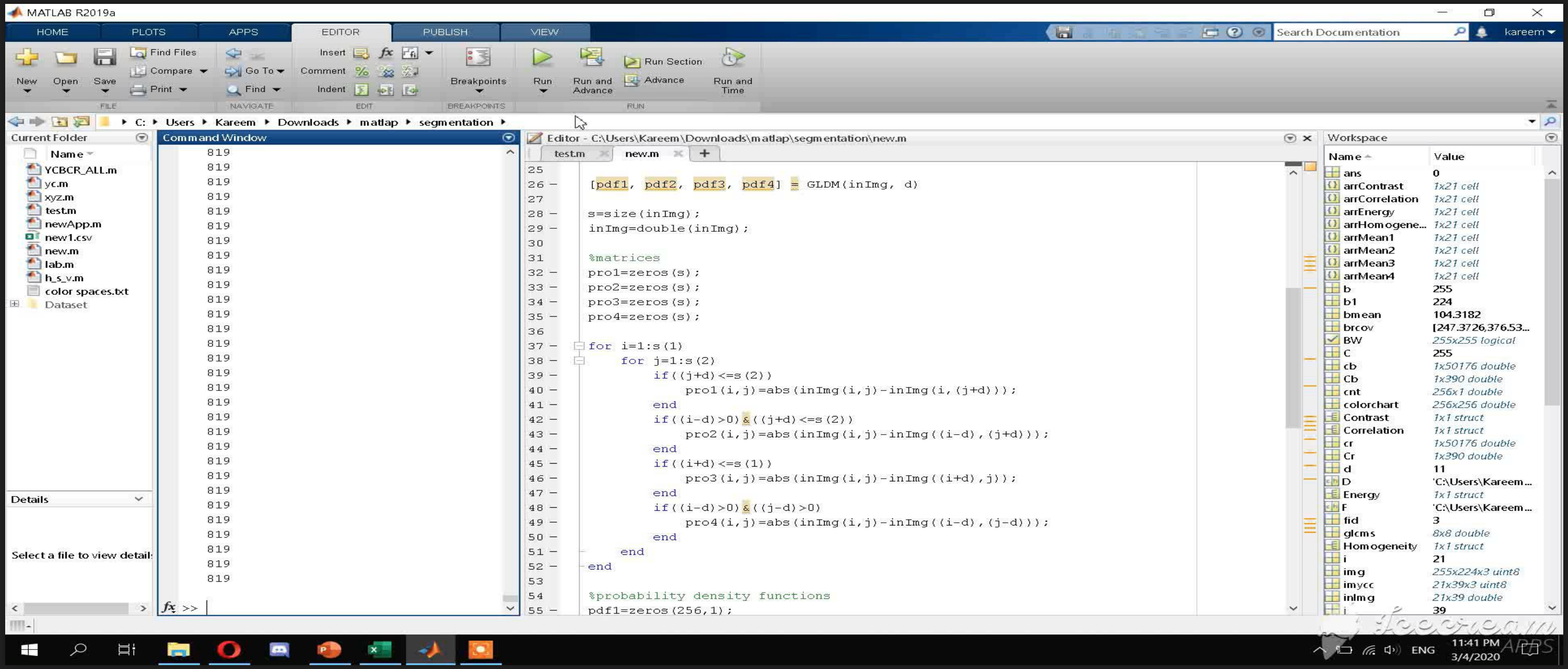







© Oliver Lucanus



Texture Feature



Thanks!