

Table of Contents

Contents

[B - Activity 1: Color Space Conversions 1](#_Toc435569744)

[B.1 - Program Description 1](#_Toc435569745)

[B.2 - Computer Vision Function Descriptions 2](#_Toc435569746)

[B.3 - Output Video Results 3](#_Toc435569747)

[C - Activity 2: Binary Video Vision Processing 4](#_Toc435569748)

[C.1 - CV Function Descriptions 4](#_Toc435569749)

[C.2 - Output Video Results 6](#_Toc435569750)

[D - Activity 3: Thresholding Fundamentals 6](#_Toc435569751)

[D.1 - CV Function Descriptions 7](#_Toc435569752)

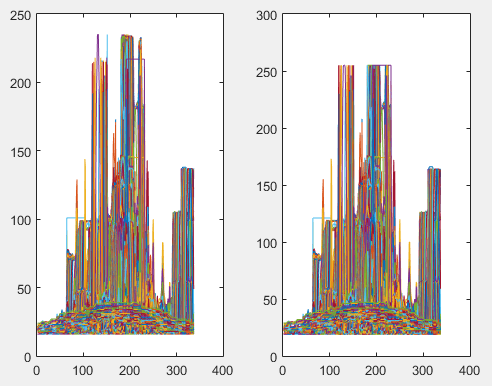
[D.2 - Output Video Results 8](#_Toc435569753)

# Activity 1: Color Space Conversions

## Program Description

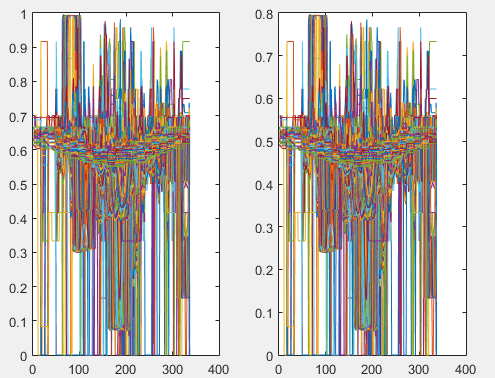
Activity 1 demonstrates the ability of the University of Alabama (UA) EcoCAR 3 Team to manipulate Color Space properties in MATLAB. This program increases the Y component in the YCbCr color space by 20% for Activity 1a and decreases the H component in the HSV color space by 20% for Activity 1b. It also places a green box over the bottom quarter of the video with “activity1a\_The University of Alabama” in the middle of that box.

The Activity 1 program was developed as a MATLAB program and can be broken down into three phases for each part of the activity: 1) Initialize Video structures 2) Process Video one frame at a time 3) Construct the Output Video. The program constructs video structures in order to store the “raw frames”, and to write the “processed frames” later. The program then processes each frame at a time. Part A of Activity 1 takes each frame from the raw video file, then increases the “Y” component of the video by 20%. Part B of Activity 1 takes each frame from the raw video file, then decreases the “H” component of the video by 20%. Finally the “processed frames” structure is written to the appropriate video file.

The first output video produced by the program is validated by inspecting a plot of the “Y” component of the image (Figure 1). Upon close inspection, some of the values on the right are now maxed out at 255. This validates that the Y component of the frame has been increased by 20%.

*Figure 1*. Activity 1a YCBCR colormap (left – before processing image, right – after processing image).

The second output video is validated by inspecting a plot of the Hue component of the image (Figure 2). Upon close inspection, some of the plot values on the right have been decreased to “.8”. This validates that the H component of the frame has been decreased by 20%.



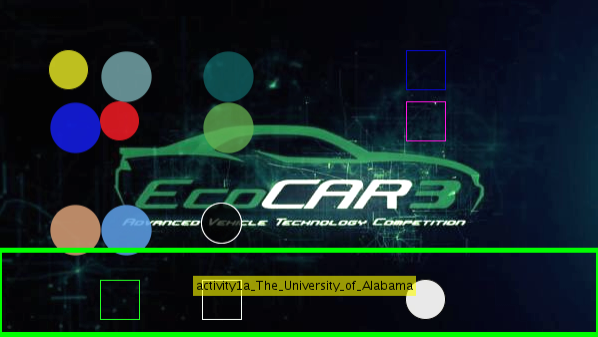
*Figure 2*. Activity 1a HSV colormap (left – before processing image, right – after processing image).

## Computer Vision Function Descriptions

* read(obj, index) – Reads a video frame at index (index) from the video object (obj). The program takes the frame returned from this function then performs the necessary color space conversion and manipulation to create the necessary processed image.
* rgb2ycbcr(Image) – Converts RGB color values to YCbCr color space. This allows manipulation of the YCbCr components of the video. The YCbCr color space can then be split into its various components (Luma, blue-difference, and red-difference), then either increased or decreased to the desired amount.
* rgb2hsv(Image) – Converts RGB colormap to HSV colormap. This allows manipulation of the HSV components of the video. The HSV color space can then be split into its various components (Hue, Saturation, and Value), then either increased or decreased to the desired amount.
* ycbcr2rgb(Image) – Converts YCbCr color values to RGB color space. This allows a manipulated image to be converted to the RGB color space (Red, Green and Blue) and then to be added to a frame structure, which will later be written to the output video file.
* hsv2rgb(Image) – Converts HSV colormap to RGB colormap. This allows to the addition of the manipulated image in the HSV color space (Hue, Saturation and Value) to a frame structure in the RGB colorspace (Red, Green, and Blue), which will later be written to the output video file.
* mat = frameImage( : , : , [1,2 or 3] ) - Returns a specific component from the colormap of the image (frameImage). This allows manipulation of the values in that component. For activity 1a this consisted of increasing the Y component of the video by 20%. In activity 1b this consisted of decreasing the H component of the video by 20%.
* cat(dim, A1, A2, A3) – Concatenates an array along a specified dimension. This allows the “merging” of components in the YCbCr (Luma, blue-difference, and red-difference) and HSV (Hue, Saturation, and Value) color spaces for a full image.
* insertShape(I, shape, position) – Inserts a shape (shape) into an image (I) at a specific position (position). This draws a bounding box at the lower 1/4 of the frame.
* insertText(I, position, textstring) – Inserts text (textstring) into an image (I) at a specific position (position). This allows insertion of text into the previously drawn shape at the bottom of the frame. In this program we inserted "activity1a\_The\_University\_of\_Alabama” and “activity1b\_The\_University\_of\_Alabama” into the image that would later be written to a frame.
* Im2frame(X) – Converts an image (X) into a movie frame. This takes the processed image to be later be converted to ta frame, and later constructed into the output video file.

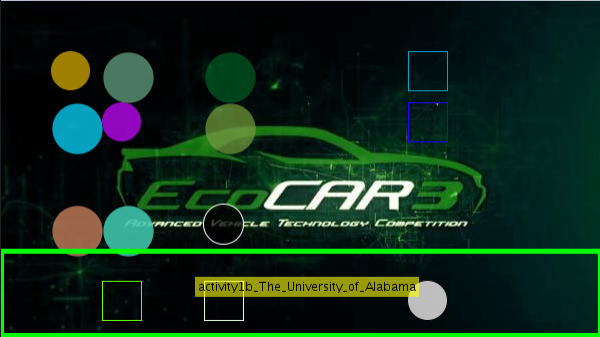
## Output Video Results

This screenshot (Figure 3.) at T (Time) = 2 seconds, demonstrates that a brighter image (specifically the “EcoCar3” logo and shapes) in this frame and confirms the “Y” component has been increased by 20%.



*Figure 3*. Activity 1a Output Video Screenshot At T (Time) = 2 Seconds

This screenshot (Figure 4.) at T (Time) = 2 seconds, demonstrates a frame that appears greener after reducing the “H” component by 20%.



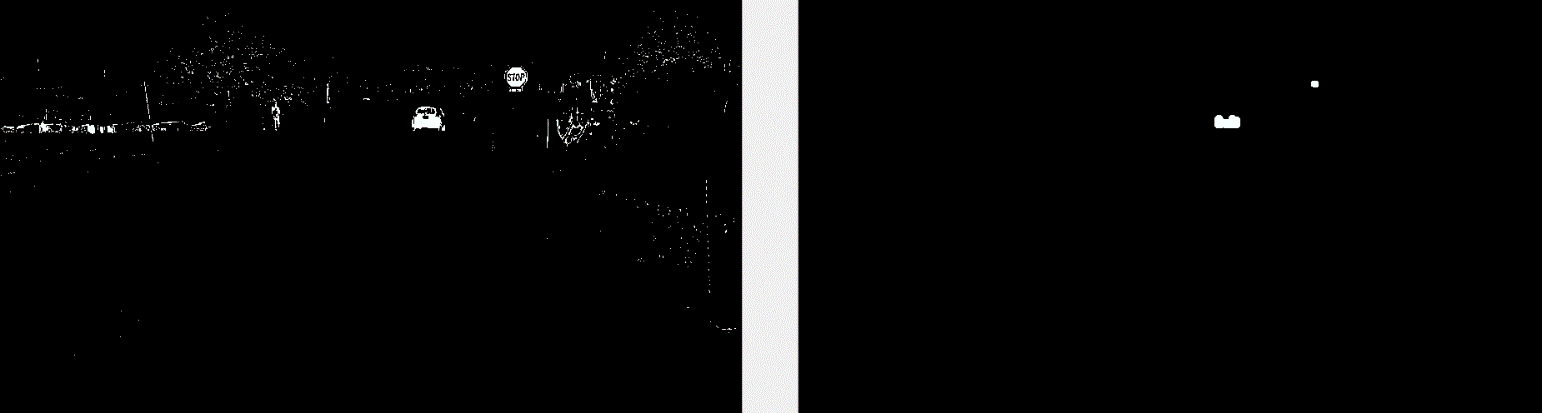
*Figure 4.* ACTIVITY 1b OUTPUT VIDEO SCREENSHOT AT T (Time) =2 seconds

# Activity 2: Binary Video Vision Processing

The Activity 2 program was developed as a MATLAB program and can be broken down into three phases: 1) Initialize Video structures, disk structure, and vision objects 2) Process Video one frame at a time 3) Construct the Output Video. The program constructs video structures in order to store the “raw frames”, and to write the “processed frames” later. The program then processes each frame at a time by reducing the noise in the image, then dilating the remaining blobs, then draws a bounding-box around the blobs stating their height, width and location with respect to the center of the frame. Finally the “processed frames” structure is written to the appropriate video file.

## CV Function Descriptions

* read(obj, index) – Reads a video frame at index (index) from the video object (obj). This program takes the frame return from this function, removes noise from the image, dilates the remaining blobs, analyzes the blobs and draws a bounding box around the blobs stating their height, width and location with respect to the center of the frame.
* rgb2gray – Converts RGB image or colormap to grayscale. This function is used to obtain a grayscale image that could later be manipulated with imdilate().
* strel(shape, parameters) – Creates morphological structuring element based on the parameter shape. A disk shape is implemented because image dilation tends to create a disk shape from the input image.
* imdilate(IM,SE) – Dilates the grayscale image (IM) based on the structuring element (SE) that is previously defined with the strel(shape, parameters) function. This function is used to dilate the car blobs in the video and enables bounding boxes to enclose the entire car.
* bwareaopen(BW, P) – Removes small objects from a binary image that have fewer than (P) pixels in the input image (BW). This removes noise from the video frame leaving only the large blobs that can later be processed. The results are shown below in Figure 5.

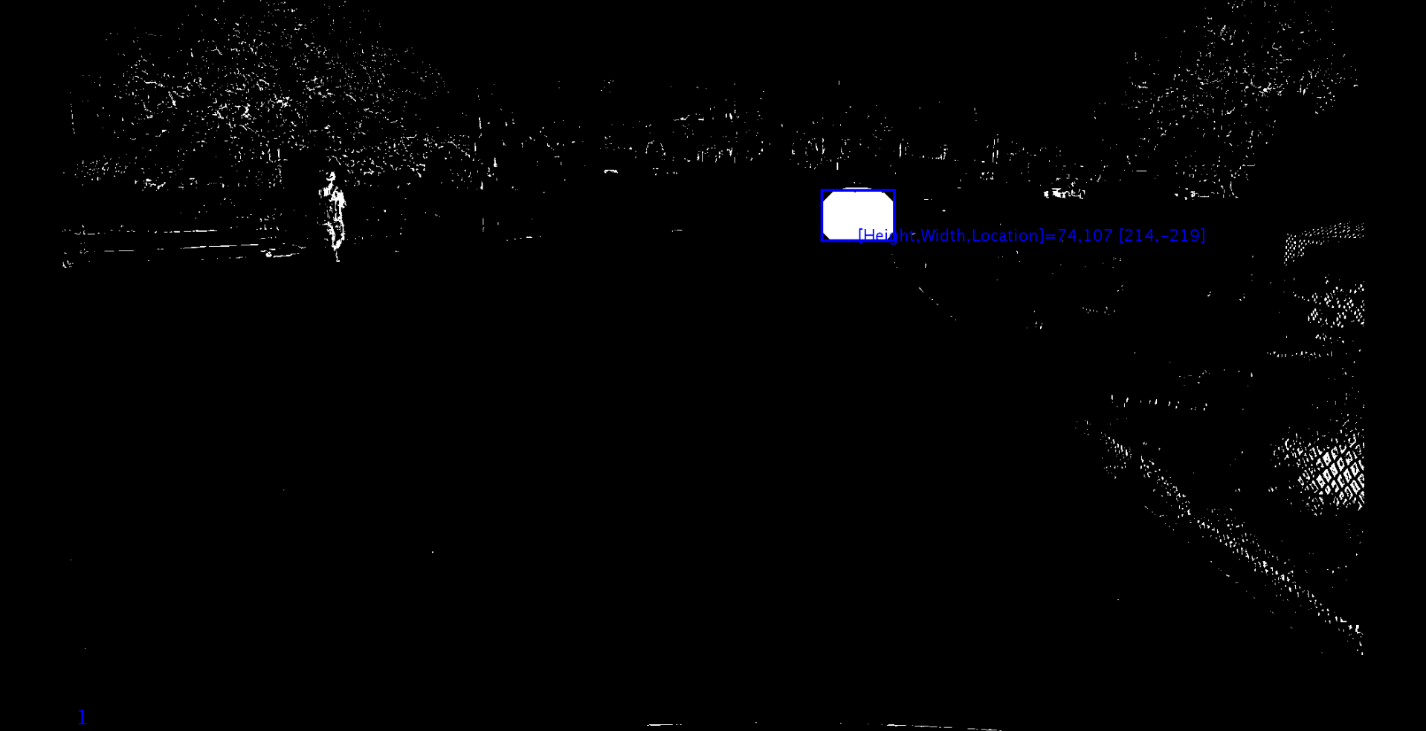


*Figure 5.* bwareaopen function (Left - Before, Right – after)

* hBlob = Vision.BlobAnalysis(Name, Value) – Returns a blob analysis object. This object can be used later on functions to determine where the car blobs are.
* [area, centroid, bbox] = step(H,BW) – Returns the area, centroid, and the bounding box of the blobs in the binary image (BW) based on the thresholds determined by the blob analysis object (H) that is previously defined.
* insertShape(I, shape, position) – Inserts a shape (shape) into an image (I) at a specific position (position). This draws a bounding box around the blobs returned from the step(H,BW) function previously defined.
* Im2frame(X) – Converts an image (X) into a movie frame. This program takes the “processed image” structure and write it to the output video file.

## Output Video Results

This screenshot (Figure 6.) at T (Time) = 2 seconds, confirms that a blue bounding box has been drawn around the distinguishable vehicle in this frame. The bounding box also has a description of its height, width and location with respect to the center of the frame.



*Figure 6.* ACTIVITY 2 OUTPUT VIDEO SCREENSHOT AT T=2

# Activity 3: Thresholding Fundamentals

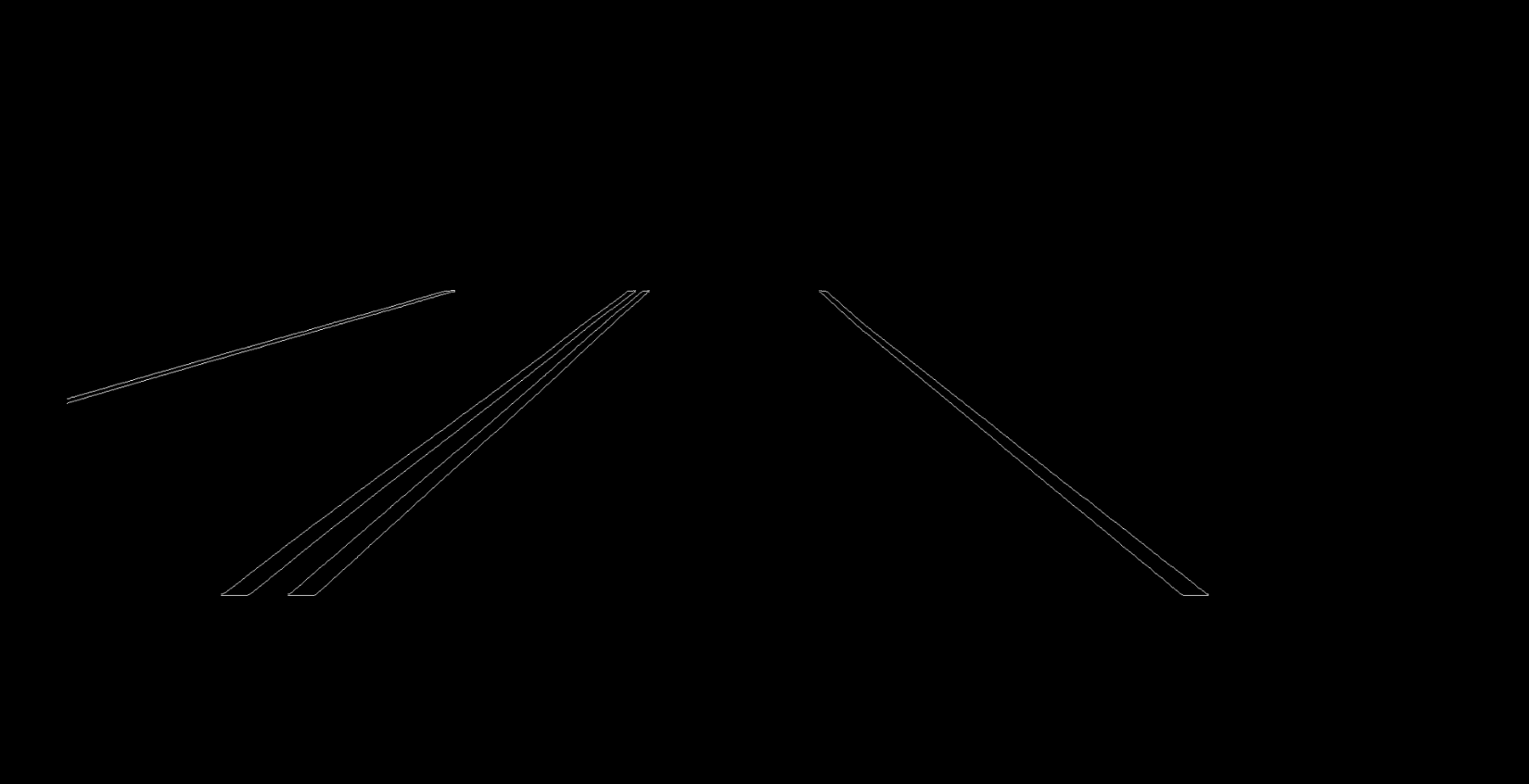
The Activity 3 program was developed as a MATLAB program and can be broken down into three phases: 1) Initialize Video structures, thresholds values, and mask values 2) Process Video one frame at a time 3) Construct the Output Video. The program constructs video structures in order to store the “raw frames”, and to write the “processed frames” later. The program then processes each frame at a time by reducing the noise in the image, converting the image to the HSV (Hue, Saturation, and Value) color space, then threshold’s the image in order to eliminate more objects that are not the road lanes, masks the hood and horizon and then detects the lanes via the prewitt edge detection method. Finally the “processed frames” structure is written to the appropriate video file.

## CV Function Descriptions

* read(obj, index) – Reads a video frame at index (index) from the video object (obj).
* false(sz, arraytype) – Creates an array size (sz) of logical 0’s. This is used to create a mesh to ignore the hood of the car and the sky.
* [x ,y] = Meshgrid(xgv ,ygv) – Creates a rectangular grid with the size of the vectors (xgv ,ygv). This is used to specify where the hood and sky mask should cover.
* hsv2rgb(Image) – Converts HSV colormap to RGB colormap. This allows the creation of masks based off of the HSV components. These masks will later be used to determine where the road lines exist.
* mat = frameImage( : , : , [1,2 or 3] ) – Returns a specific component from the colormap of the image (frameImage). This allows manipulation of the values in that component and creates a mask with those values.
* whitevalueMask = (vImage >= whitevalueThresholdLow) & (vImage <= whitevalueThresholdHigh) – Creates a mask based on the threshold values which are set respectively at .575 and 1. This took significant tuning because of the light gravel and light marks on the asphalt. The tuning technique for the “whitevalueThresholdLow” was similar to a binary search. Starting at .5, then the threshold was adjusted to the next “half”, .75. Depending on the result, the threshold was moved to the next “half”, whether higher or lower, and repeated until the desired result was achieved.
* bwareaopen(BW, P) – Removes small objects from a binary image that have fewer than (P) pixels in the input image (BW). This removes noise from the video which helps to identify the road lines.
* BW = edge(I, method) – Detects edges in image (I), based off of the method (method). The prewitt method was chosen after tuning the video and determining it was the most effective edge detection method for our program.
* repmat(uint8(moreEdit2).\*255, [1 1 3]) – Reconstructs the frame after having detected lines in the image. uint8(moreEdit2) converts the “moreEdit2” image to consist of 8-bit unsigned integers that can be written into the output video.
* im2frame(X) – Converts an image (X) into a movie frame. This is later written into the output video file.

## Output Video Results

This screenshot (Figure 7.) at T (Time) = 2 seconds, shows a fully processed frame of the video at T (Time) =2 where all noise has been eliminated and only the road lanes are shown as white pixels.



*Figure 7.* ACTIVITY 3 OUTPUT VIDEO SCREENSHOT AT T (Time) =2 seconds