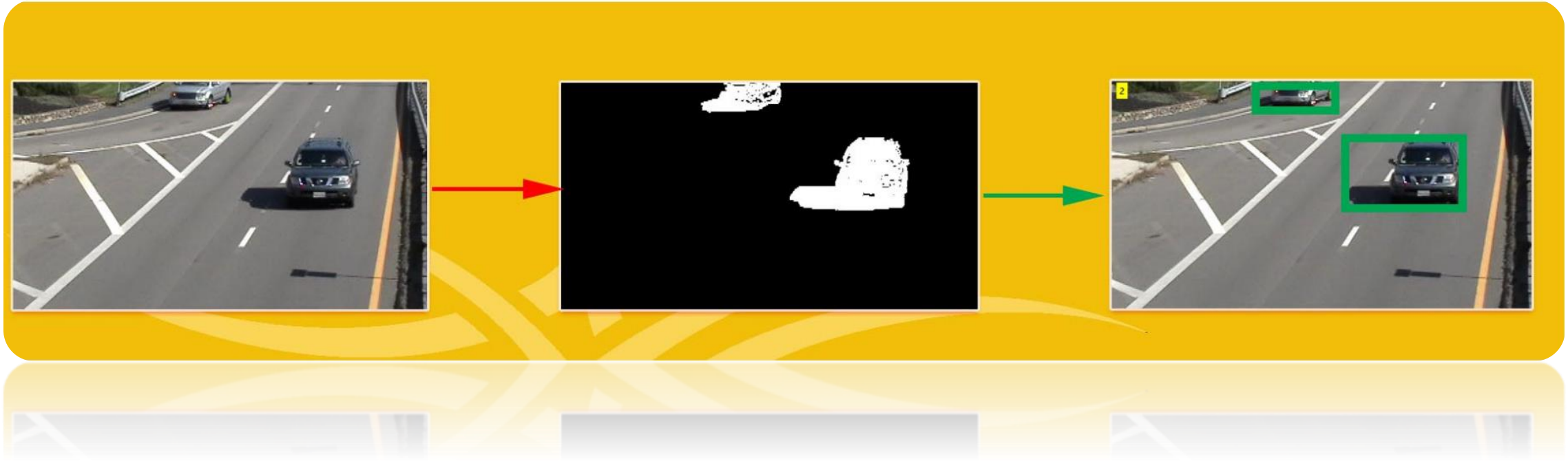
Term Project

Mat 116e – 2020-21 Spring

090150502 Berrin Göçer

090180328 Umut Hasan Ağbaba

Traffic control system with Image Processing method



Aims and Goals:

To predict and make accurate predictions by analyzing the intensity of traffic jams Traffic control system with Image Processing method

Step 1 – Importing the Video File

The first step is to import the video from where we are going to detect the vehicles. We can do it using the 'VideoReader()' function of MATLAB. This function requires the location and name of the video with the video extension to function properly. If everything is okay, then this function will read the video from the source. The following code shows how to use the 'VideoReader()' function.

Here, the name of the video file is 'traffic.avi'. This file is located it 'video' folder. That means the complete path to the video is 'video/traffic.avi'. The complete path to the video has been used as the input argument of the 'VideoReader()' function. The function returns the video which has been stored in 'the_Video' variable.

```
the_Video = VideoReader('video/traffic.avi');
```

Step 2 – Object Detection

The video processing is a time consuming process. If we try to process the entire video and then detect the vehicle, it will require significant amount of time. Moreover, it is not applicable for live traffic camera when the videos are continuous stream of frames. That is why the better approach is to start with some initial video frames where the vehicles are present. Then we can segment the vehicles from the frame and use them to train the machine learning model to recognize the vehicle. In this approach, we need to employ an unsupervised learning approach. In this lesson, we are going to use Gaussian Mixture Model (GMM) as machine learning model.

We can do all of these tasks using 'vision.ForegroundDetector(Name,Value)'. It is the simplest possible way to detect object (foreground) from an image. We can control the behavior of this object detector by using different "name, value" pair. We are going to use the Gaussian Mixture Model (GMM) with three background nodes. That is why, we will use "'NumGaussians', 3," as input argument of the object detector.

There are thousands of frames in a video. We should not use all of them for the training. It will make the process slower. In order to limit the number of frames we want to use in training, we can pass "'NumTrainingFrames', value" pair in the object detector. Here the 'NumTrainingFrames' stands for "number of training frames". The value represents the number of frames we are interested in. It must have to be an integer. The following code shows how to use the 'vision.ForegroundDetector(Name,Value)' with optimized "name, value" pair for vehicle detection.

Here, we are using 50 frames to train the GMM with 3 background nodes.

```
Object_Detector = vision.ForegroundDetector('NumGaussians', 3, 'NumTrainingFrames', 50);
```

Step 3 – Training the Model

The next task is to read the frames from the video file and pass it to the object detector. In order to read consecutive frames, we need to use a ‘for loop’. Inside the ‘for loop’, we can pass the frames to object detector one by one using ‘step function. The following block of code demonstrates the process:

```
for i = 1:150 frame = readFrame(the_Video); the_Object = step(Object_Detector, frame); end
```

In every iteration of the ‘for loop’ the frame contains the vehicle are passed to the object detector. The object detector learns to detect the vehicle though Gaussian Mixture Model (GMM) algorithm. We can visualize the frame using the following line of code

```
figure; imshow(frame); title('The Video Frame');
```

The code above shows the video frame where the vehicles are available. The object detector returns the detected object. To see the detected objects, we can use the following code:

```
figure; imshow(the_Object); title('Foreground');
```

Step 4 – Morphological Operation to Remove Noise

The accuracy of GMM model is satisfactory. However, sometimes noises are there in the background with the detected object. When a vehicle is far away from the camera, the size of the noises and the size of the vehicle becomes similar. As a result, the detector may generates undesirable result. In order to avoid such shortcomings, we can use morphological opening to remove the noise and fill the gaps in the detected objects. The morphological operation starts by defining the structuring element.

In MATLAB, we define the structure using 'strel()' function. This function requires two arguments. The first argument is the type of the structuring element. The second argument defines the width of the pixel. In our system, we are going to use a 'square' structure and the width is going to be 3 pixels. The following code defines the structuring element:

```
Structure = strel('square', 3);
```

After defining the structure, we can apply it to the nosy image using 'imopen()' function. The first argument of this function is the image where we want to perform the morphological operation. He second argument is the structure. In our code, the target image is the 'the_Object' and the structure is the 'Structure'. The following code applies the morphological opening to the image:

```
Noise_Free_Object = imopen(the_Object, Structure);
```

After applying morphological operation, the 'imopen()' function returns an image with less noise. We have stored this image in a variable named 'Noise_Free_Object'. We can visualize the 'Noise_Free_Object' using the following code:

```
figure; imshow(Noise_Free_Object); title('Object After Removing Noise');
```

Step 5 – Perfecting and Locating the Object

Now it is time to draw a rectangle around the detected object. We can do it using the 'vision.BlobAnalysis()' object. However, we have to define appropriate parameters to draw the rectangle. The 'vision.BlobAnalysis()' object accepts multiple parameters. What we need is a bounding box around the detected object. And it is defined by the 'BoundingBoxOutputPort' parameter. We need to specify the value of this parameter. It can take two values – 'true' or 'false'. As we want the bounding box, we have to use 'true'. There are other parameters such as 'AreaOutputPort' to get the area of the enclosed object, 'CentroidOutputPort' to get the center of the object. For the time being, we do not need them. Set the values of these parameter to false. The 'vision.BlobAnalysis()' object requires uniform object to draw the bounding box. However, in our image, there are many blobs which are not connected. However, we can use the 'MinimumBlogArea' parameter and set the value to 150 to force the 'vision.BlobAnalysis()' to ignore blobs larger than 150 pixels. All of these have been done using the following code:

```
Bounding_Box = vision.BlobAnalysis('BoundingBoxOutputPort', true, ...
    'AreaOutputPort', false, 'CentroidOutputPort', false, ...
    'MinimumBlobArea', 150);
```

Step 6 – Inserting the Box

In this step, we are going to insert the bounding box we created in the previous step using ‘vision.BlobAnalysis()’ object. We can do it using ‘step()’ function. This function requires two argument. The first argument is the bounding box and the second argument is the image where we want to place the box. Here is the code to use ‘step()’ function:

```
the_Box = step(Bounding_Box, Noise_Free_Object);
```

Step 7 – Drawing the Rectangle

In the previous step, we have represented the detected objects using bounding box. That means now we can draw rectangle around the boxes and place the rectangle on the frame. After doing it, we will see the rectangles encoding the vehicles. And this is exactly what the vehicle detection is. We can insert the rectangle around the box using the 'insertShape()' function. The first argument of this function will be the image where we want to insert the shape. In our program, we want to insert the shape on 'frame'. Second argument defines the type of the shape. We want to insert 'Rectangle'. The third argument is the coordinate of the rectangle. We have stored the coordinate of the box which represent the detected object in 'the_Box' variable. Simply putting it as the 'coordinate' will do the work. Then use 'Color' argument. We can use any value of this argument. Let's use 'blue' to create a blue box around the object. We can also define the 'LineWidth' of the rectangle. We are going to set to '5' to have a thick line around the detected vehicle. The following code is doing all of these things:

```
Detected_Car = insertShape(frame, 'Rectangle', the_Box, 'Color', 'green');
```

Step 8 – Counting the Number of Cars

Now every vehicle will be enclosed by a rectangle. If we count the number of rectangle, we are actually counting the number vehicles. And doing it is a piece of cake. We can count the number of rectangle using the 'size()' function. We need to input arguments to use the 'size()' function here. The first one is the coordinate of the rectangle, which is stored in the 'the_Box' variable. The second argument is the dimension. We want a number from the size function, which is a one dimensional. That is why the second argument is going to be '1'. The following code counts the number of cars and store it the variable named 'Number_of_Cars'.

```
Number_of_Cars = size(the_Box, 1);
```

Step 9 – Inserting the Number of Car

It is time to show the number of cars and the detected vehicle on the image. There is a function in MATLAB named 'insertText()'. We can use this function to insert text in the image frame. The first argument of this function is the image where we want to insert the text. Then we have to declare the size of the text box we are going to place on the image. We have to define the size as [10 10]. It means the size of the box is 10 by 10 pixels. Then we have to write the text we want to show on the image. In our code, the number of car is stored in a variable named 'Number_of_Cars' variable. That is why we have to put this variable as the argument of the function. We can control the opacity of the textbox using the 'BoxOpacity' property. We are creating a fully non-transparent box by setting the value of the 'BoxOpacity' to 1. We can change the size of the font using 'FontSize' property. Taking the font size '14' is standard.

The 'insertText()' function will return the frame with everything we wanted to include in the frame. We can replace the content on 'Detected_Car' variable by content returned from 'insertText()'. And we can display the final result using 'imshow()' function. All of these has been implemented using the following block of code:

```
Detected_Car = insertText(Detected_Car, [10 10], Number_of_Cars, 'BoxOpacity', 1, ...
    'FontSize', 14);
figure; imshow(Detected_Car); title('Detected Cars')
```

If we run the code now, we will get the detected cars enclosed by rectangles along with the number of cars on the frame.

Step 10 – Applying the Concept in Video Sequence

It's time to detect the tool from the video sequence. In fact, we've already done everything we need to do to detect the vehicle from video. Video is nothing but a stream of still pictures called frames. Since we have already developed a method to detect vehicle from frames, we can use exactly the same method in a loop and apply it to each frame and detect vehicle.

But before we start the loop, we need to define the video player. We use the 'vision.VideoPlayer()' object to define the video player in MATLAB. Next we need to specify the name of the video player. We can use any name. However, "Perceived Cars" sounds an apt name. Next we need to set the aspect ratio. We can do this using 'Position (3:4)'. This means the aspect ratio is 3:4. Then the size of the video frames is set to 650×400 pixels.

```
videoPlayer = vision.VideoPlayer('Name', 'Detected Cars');
videoPlayer.Position(3:4) = [650,400];
```

The video player is ready. It's time to create a loop so we can follow the car through the video frames. A while loop is created. This loop will continue to run as long as there are frames in the video. Then followed steps:

1. Read frame from 'the_Video': frame = readFrame(the_Video);
2. Then the frame is passed to the object detector: the_Object = step(Object_Detector, frame);
3. Removes the noise from the frame: Noise_Free_Object = imopen(the_Object, Structure);
4. Get the coordinate of the bounding box: the_Box = step(Bounding_Box, Noise_Free_Object);
5. A rectangle is placed around the car using the coordinate obtained in the previous section: Detected_Car = insertShape(frame, 'Rectangle', the_Box, 'Color', 'green');
6. Count the number of cars: Number_of_Cars = size(the_Box, 1);
7. Enter the car number in the frame: Detected_Car = insertText(Detected_Car, [10 10], Number_of_Cars, 'BoxOpacity', 1, 'FontSize', 14);
8. Finally, the video player shows the last frame: step(videoPlayer, Detected_Car);

The 'while loop' will repeat these 8 steps at each iteration. This means the vehicle will be tracked in every frame.

The MATLAB Code

```
clc;clear all


the_Video = VideoReader('traffic.avi');
Object_Detector = vision.ForegroundDetector('NumGaussians', 3, 'NumTrainingFrames', 50);
for i = 1:150
frame = readFrame(the_Video);
the_Object = step(Object_Detector, frame);
end
figure; imshow(frame); title('Video Frame');
figure; imshow(the_Object); title('The Object');
Structure = strel('square', 3);
Noise_Free_Object = imopen(the_Object, Structure);
figure; imshow(Noise_Free_Object); title('Object After Removing Noise');
Bounding_Box = vision.BlobAnalysis('BoundingBoxOutputPort', true,...
'AreaOutputPort', false, 'CentroidOutputPort', false,...
'MinimumBlobArea', 150);
```

Continuation of the code:

```
the_Box = step(Bounding_Box, Noise_Free_Object);
Detected_Car = insertShape(frame, 'Rectangle', the_Box, 'Color', 'green');
Number_of_Cars = size(the_Box, 1);
Detected_Car = insertText(Detected_Car, [10 10], Number_of_Cars, 'BoxOpacity', 1,'FontSize', 14);
figure; imshow(Detected_Car); title('Detected Cars');
videoPlayer = vision.VideoPlayer('Name', 'Detected Cars');
videoPlayer.Position(3:4) = [650,400];

while hasFrame(the_Video)
frame = readFrame(the_Video); the_Object = step(Object_Detector, frame); Noise_Free_Object = imopen(the_Object,
Structure); the_Box = step(Bounding_Box, Noise_Free_Object); Detected_Car = insertShape(frame, 'Rectangle', the_Box,
'Color', 'green'); Number_of_Cars = size(the_Box, 1); Detected_Car = insertText(Detected_Car, [10 10], Number_of_Cars,
'BoxOpacity', 1, 'FontSize', 14); step(videoPlayer, Detected_Car);
end
```

Figure 1

File Edit View Insert Tools Desktop Window Help

Video Frame

Figure 2

File Edit View Insert Tools Desktop Window Help

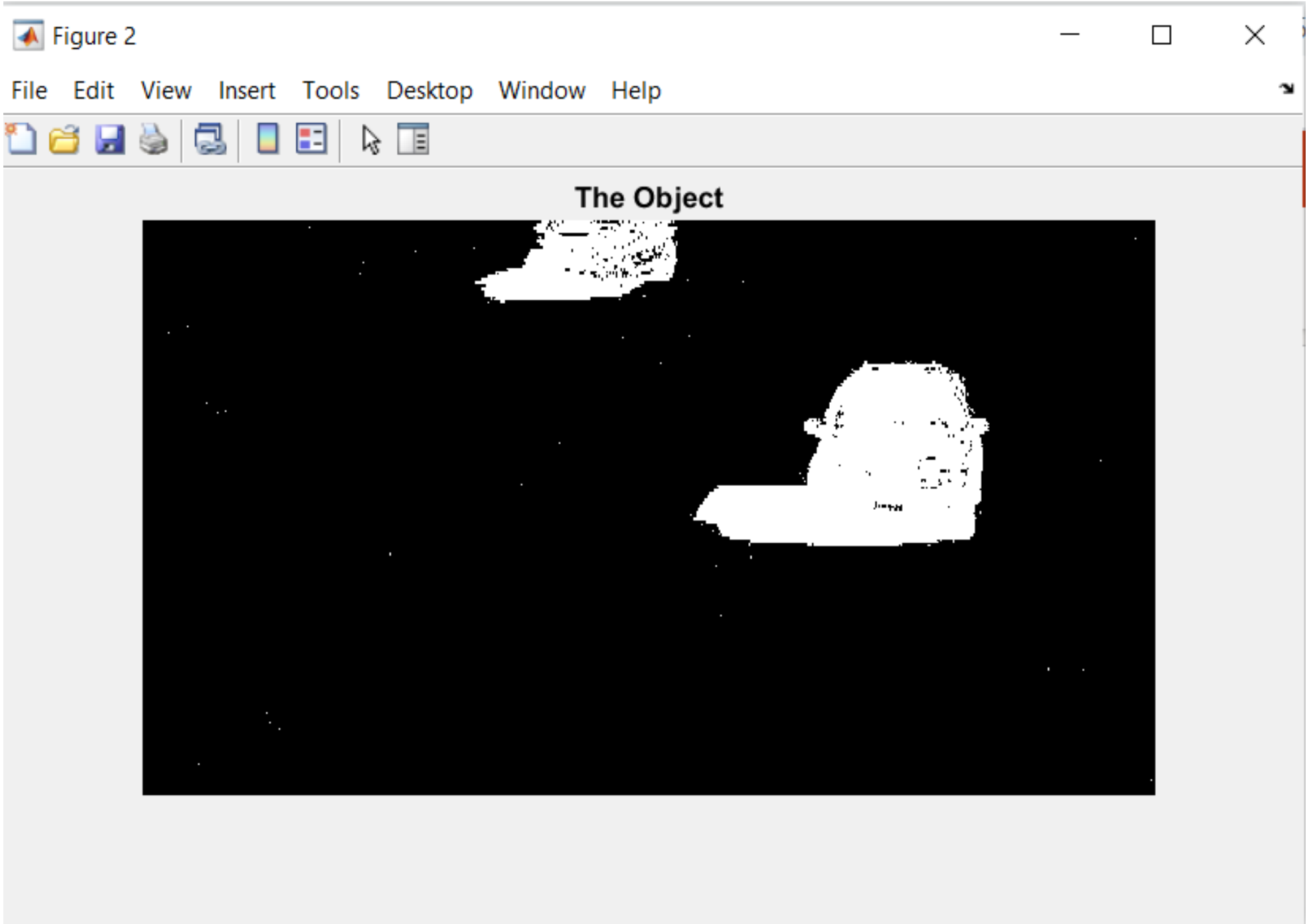


Figure 3

File Edit View Insert Tools Desktop Window Help

Object After Removing Noise

Figure 4

File Edit View Insert Tools Desktop Window Help