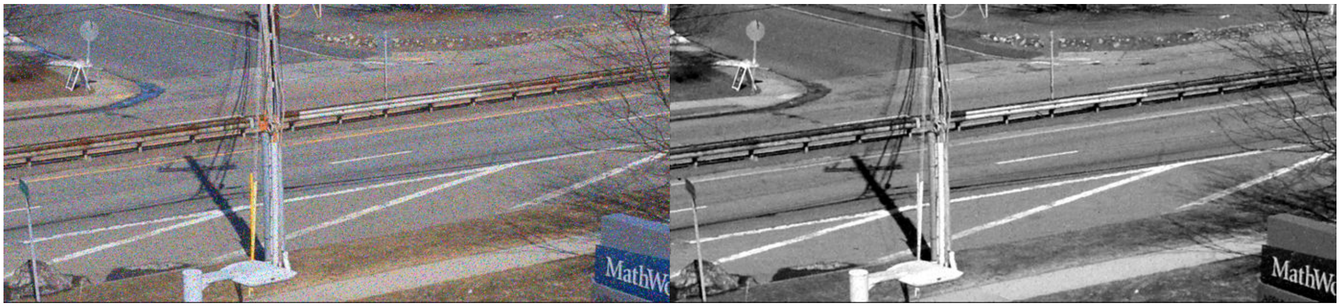
Cars Detection Project

Table of Contents

Section 1: Enhancing the video

Enhance the video by removing the noise, convert the video to grayscale, then save the result as a new video file.

For an example frame, go from the noisy image (left) to the grayscale image (right).



```
% Read the input video
v = VideoReader('RoadTraffic.mp4');

% Create a VideoWriter object for the output video
vOut = VideoWriter('CleanRoadTraffic.mp4', 'MPEG-4');
vOut.FrameRate = v.FrameRate;
open(vOut);

% Loop through each frame of the input video
while hasFrame(v)
    % Read a frame from the input video
    frame = readFrame(v);

    % Enhance the frame using the enhanceImage() function
    enhancedFrame = enhanceImage(frame);

    % Write the enhanced frame to the output video
    writeVideo(vOut, enhancedFrame);
end
% Close the VideoWriter object
close(vOut);
```

Section 2.1: Isolating the cars with background subtraction

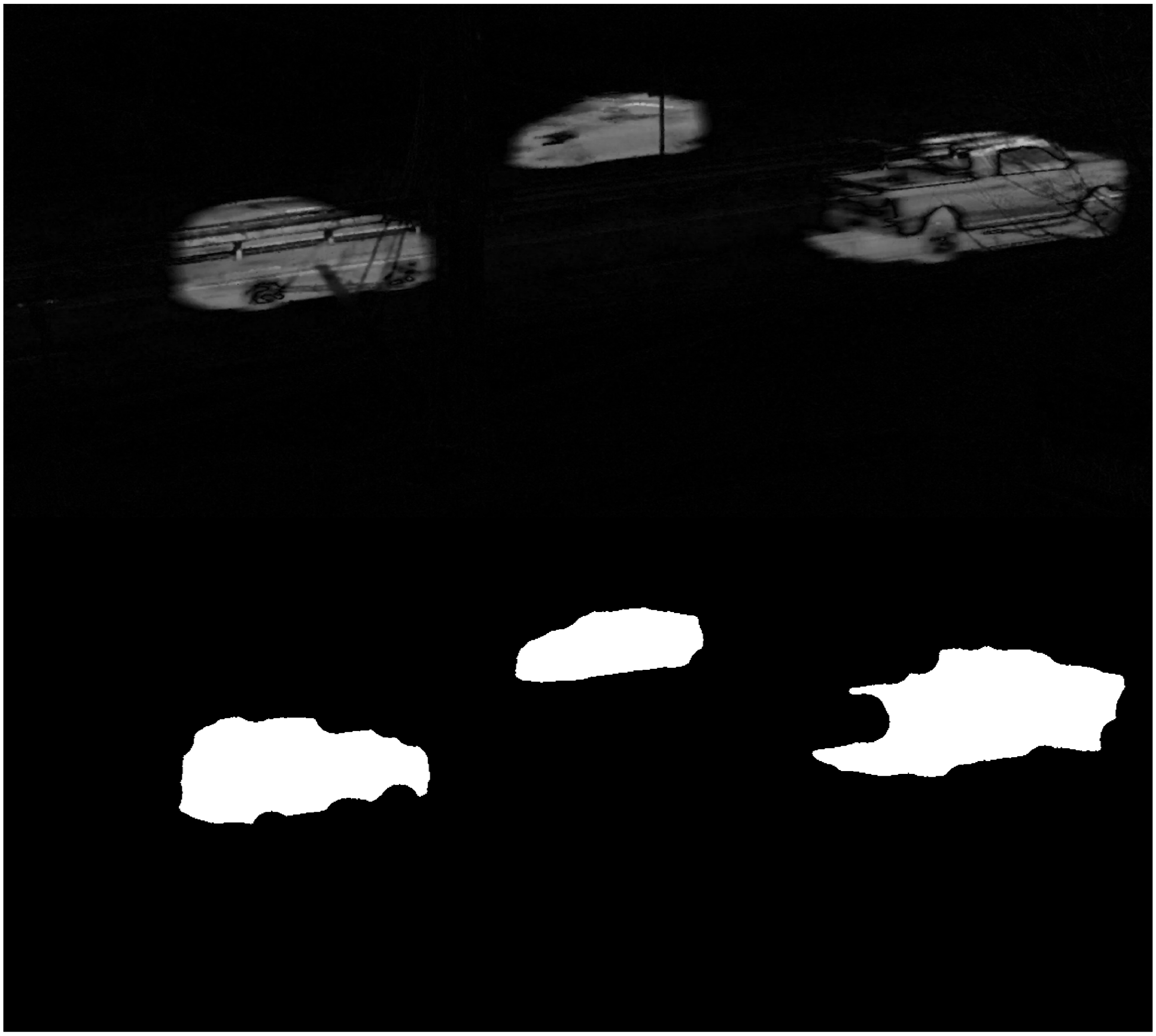
Isolate the cars using background subtraction.

The end goal of section 2 for an example frame, is to go from the grayscale image (left) to the BW mask image (right).

Background subtraction preparation

First, create a background image with no cars from the first frame.

```
v= VideoReader("CleanRoadTraffic.mp4");
frameSum = im2double(read(v,1));
for idx = 2:v.NumFrames
    frameSum = frameSum + im2double(read(v,idx));
end
aveFrame = frameSum/ v.NumFrames;
% Test Frame Segmentation Algorithm
frame =read(v,153);
frame = im2double(frame);
background = aveFrame;
frameDiff = abs(frame - background);
[CarsBW,CarsMasked] = segmentedCars2(frameDiff);
montage({frameDiff,CarsBW})
```



```
CarsProps = regionprops("table",CarsBW,"all");
CarsProps
```

```
CarsProps = 3×30 table
```

...

	Area	Centroid	
1	19711	290.0482	253.4751
2	10112	613.5195	129.4139
3	25123	987.5015	197.1641

```
totalTruePixels = sum(CarsBW(:));
```

Sections 2.2 & 3: Segmenting cars and Calculating region properties

Segment the cars and create a table that contains a row for each frame of the video and a column for the following three properties: number of regions, mean region size, and total region size.

```
% Initialize the video reader object.
v= VideoReader("CleanRoadTraffic.mp4");
frameSum = im2double(read(v,1));
for idx = 2:v.NumFrames
    frameSum = frameSum + im2double(read(v,idx));
end
aveFrame = frameSum/ v.NumFrames;
imshow(aveFrame)

% Get the total number of frames in the video
numFrames = v.NumFrames;

% Create a table to store the region information
regionTable = table();

% Initialize variables for average region size calculation
totalRegionSize = 0;
numFramesWithCar = 0;

% Process each frame
for frameNumber = 1:numFrames
    % Read the current frame
    frame = read(v, frameNumber);

    % Convert the frame to double precision
    frame = im2double(frame);

    % Calculate frame difference
    frameDiff = abs(frame - aveFrame);

    % Segment the cars using the segmentedCars2() function
    [CarsBW, ~] = segmentedCars2(frameDiff);

    % Calculate region properties
    CarsProps = regionprops(CarsBW, 'Area');

    % Check if at least one car is detected in the frame
    if ~isempty(CarsProps)
        % Update region size and frame count
        totalRegionSize = totalRegionSize + sum([CarsProps.Area]);
        numFramesWithCar = numFramesWithCar + 1;
    end

    % Create a row for the current frame in the region table
    frameRow = table(frameNumber, size(CarsProps, 1), mean([CarsProps.Area]),
sum([CarsProps.Area]));
```

```
    % Append the frame row to the region table
    regionTable = [regionTable; frameRow];
end

% Calculate the average region size in terms of pixels
averageRegionSize = totalRegionSize / numFramesWithCar;

% Display the region table
disp(regionTable);
```

```
    frameNumber    Var2     Var3     Var4 
    ___________    ____    ______    _____

          1         0         NaN        0
          2         0         NaN        0
          3         0         NaN        0
          4         0         NaN        0
          5         1          73       73
          6         2        3132     6264
          7         2       10186    20372
          8         2       15974    31949
          9         2       15258    30515
         10         2       13942    27883
         11         2       13329    26658
         12         2       14908    29817
         13         2       14368    28735
         14         2       13739    27478
         15         4        6493    25972
         16         3       12458    37373
         17         3       16533    49599
         18         2       23313    46626
         19         2       19228    38455
         20         2       15364    30727
         21         1       24430    24430
         22         1       23205    23205
         23         1       22251    22251
         24         1       20466    20466
         25         1       17513    17513
         26         1       13400    13400
         27         1        5144     5144
         28         1          94       94
         29         0         NaN        0
         30         0         NaN        0
         31         0         NaN        0
         32         0         NaN        0
         33         0         NaN        0
         34         0         NaN        0
         35         0         NaN        0
         36         0         NaN        0
         37         0         NaN        0
         38         0         NaN        0
         39         0         NaN        0
         40         0         NaN        0
         41         0         NaN        0
         42         0         NaN        0
         43         0         NaN        0
         44         0         NaN        0
         45         0         NaN        0
         46         0         NaN        0
         47         0         NaN        0
         48         0         NaN        0
         49         0         NaN        0
```

```
 50         0         NaN        0
 51         0         NaN        0
 52         0         NaN        0
 53         0         NaN        0
 54         0         NaN        0
 55         0         NaN        0
 56         0         NaN        0
 57         0         NaN        0
 58         0         NaN        0
 59         0         NaN        0
 60         1          42       42
 61         1        2549     2549
 62         1        6460     6460
 63         1        6747     6747
 64         1        8166     8166
 65         1        8625     8625
 66         1        8809     8809
 67         1        9150     9150
 68         1        8320     8320
 69         2      4076.5     8153
 70         1        9888     9888
 71         1       10826    10826
 72         2        3581     7162
 73         3      124.67      374
 74         1       11786    11786
 75         1       24235    24235
 76         1       20671    20671
 77         1       17609    17609
 78         1       18076    18076
 79         1       18087    18087
 80         1       17111    17111
 81         1       16512    16512
 82         1       15258    15258
 83         1       14483    14483
 84         1       11439    11439
 85         1        4394     4394
 86         0         NaN        0
 87         0         NaN        0
 88         0         NaN        0
 89         0         NaN        0
 90         0         NaN        0
 91         0         NaN        0
 92         0         NaN        0
 93         1         398      398
 94         1       10991    10991
 95         1       20734    20734
 96         1       18899    18899
 97         1       16052    16052
 98         2      6772.5    13545
 99         1       17413    17413
100         1       17004    17004
101         1       15798    15798
102         1       13568    13568
103         1       10744    10744
104         1        6292     6292
105         2       280.5      561
106         0         NaN        0
107         1        5384     5384
108         1       15863    15863
109         1       14658    14658
110         3        2682     8046
111         2        3976     7952
112         1       11724    11724
113         1       10511    10511
```

```
114         1        9160     9160
115         1        6624     6624
116         2      2371.5     4743
117         1        3622     3622
118         1         459      459
119         0         NaN        0
120         0         NaN        0
121         0         NaN        0
122         0         NaN        0
123         0         NaN        0
124         0         NaN        0
125         0         NaN        0
126         0         NaN        0
127         0         NaN        0
128         0         NaN        0
129         0         NaN        0
130         0         NaN        0
131         0         NaN        0
132         0         NaN        0
133         0         NaN        0
134         0         NaN        0
135         0         NaN        0
136         0         NaN        0
137         0         NaN        0
138         0         NaN        0
139         0         NaN        0
140         0         NaN        0
141         0         NaN        0
142         1         213      213
143         1        2353     2353
144         1       16687    16687
145         1       32419    32419
146         1       37656    37656
147         2       17920    35840
148         2       18154    36308
149         4      8997.2    35989
150         4      9968.5    39874
151         2       27638    55276
152         3       19037    57112
153         3       18315    54946
154         2       22521    45042
155         3       11203    33609
156         3       10128    30385
157         2       14450    28899
158         2       14054    28109
159         2        9518    19036
160         1       13343    13343
161         1       10368    10368
162         1        3997     3997
163         0         NaN        0
164         1         654      654
165         1        3877     3877
166         1        5646     5646
167         1        7455     7455
168         1        7944     7944
169         1        8400     8400
170         1        8382     8382
171         2        3263     6526
172         1        6890     6890
173         1        7407     7407
174         1        9406     9406
175         1        3429     3429
176         1        3770     3770
177         1       17199    17199
```

```
178         1       32386    32386
179         1       30324    30324
180         1       26825    26825
181         1       25078    25078
182         1       26369    26369
183         1       25784    25784
184         1       24275    24275
185         1       22814    22814
186         2       13459    26918
187         2       17796    35592
188         3       12355    37066
189         2       14948    29897
190         1       26761    26761
191         1       23893    23893
192         1       24704    24704
193         1       24522    24522
194         1       23742    23742
195         1       22381    22381
196         1       20716    20716
197         1       18098    18098
198         1       10664    10664
199         1        3576     3576
200         0         NaN        0
201         0         NaN        0
202         1        8100     8100
203         1       21272    21272
204         1       17952    17952
205         1       14747    14747
206         1       15199    15199
207         1       16465    16465
208         1       16057    16057
209         1       14545    14545
210         1       13047    13047
211         1        8172     8172
212         1        3205     3205
213         0         NaN        0
214         0         NaN        0
215         0         NaN        0
216         0         NaN        0
217         0         NaN        0
218         0         NaN        0
219         1         630      630
220         3      3083.7     9251
221         2       15617    31234
222         2       15599    31198
223         2       14644    29288
224         3      8502.3    25507
225         2       15004    30008
226         3       14600    43799
227         3       15888    47665
228         3       14897    44692
229         3       11782    35347
230         4      6594.8    26379
231         2       10886    21772
232         1       17766    17766
233         1       16639    16639
234         1       15411    15411
235         1       11377    11377
236         1        4279     4279
237         0         NaN        0
238         0         NaN        0
239         0         NaN        0
240         0         NaN        0
```

```
% Display the average region size
disp("Average region size (pixels): " + averageRegionSize);
```

```
Average region size (pixels): 18500.0621
```

```
%% Moving Car Detection Project
% Task is to finding moving cars, and insert a bounding box around it

v = VideoReader("NewRoadTraffic.mp4");
v.FrameRate;
firstFrame = read(v,1);
firstFrame = im2double(firstFrame);

v2 = VideoWriter("car_bound2.mp4", "MPEG-4");
v2.FrameRate = v.FrameRate;
open(v2)

for idx = 1:v.NumFrames
    frame = read(v,idx);
    frame = im2double(frame);

    frameDiffer = abs (firstFrame - frame);

    bw = segmentedCars2(frameDiffer);

    props = regionprops("table",bw,"BoundingBox");

    carboxed = insertShape(frame,"rectangle",props.BoundingBox,...
        "LineWidth",3,"Color","red");
    writeVideo(v2,carboxed)
end
close(v2);
```

```
% Create VideoReader objects for both videos
video1Reader = VideoReader("RoadTraffic.mp4");
video2Reader = VideoReader("car_bound2.mp4");

% Get video properties
frameRate = video1Reader.FrameRate;
frameWidth = video1Reader.Width;
frameHeight = video1Reader.Height;

% Create a new VideoWriter object for the output montage video
outputVideo = VideoWriter("ComparisonRoadTraffic.mp4", 'MPEG-4');
outputVideo.FrameRate = frameRate;
open(outputVideo);

% Read and process each frame from both videos
```

```
while hasFrame(video1Reader) && hasFrame(video2Reader)
    % Read frames from both videos
    frame1 = readFrame(video1Reader);
    frame2 = readFrame(video2Reader);

    % Resize frame 2 to match frame 1's height
    frame2 = imresize(frame2, [frameHeight, NaN]);

    % Create a new frame with both videos side by side
    montageFrame = [frame1, frame2];

    % Write the frame into the output video
    writeVideo(outputVideo, montageFrame);
end

% Close the output video file
close(outputVideo);
```

Segmentation function

```
function [BW,maskedImage] = segmentedCars2(X)
%segmentImage Segment image using auto-generated code from Image Segmenter app
%  [BW,MASKEDIMAGE] = segmentImage(X) segments image X using auto-generated
%  code from the Image Segmenter app. The final segmentation is returned in
%  BW, and a masked image is returned in MASKEDIMAGE.

% Auto-generated by imageSegmenter app on 18-Jul-2023
%----------------------------------------------------

% Threshold image with manual threshold
BW = im2gray(X) > 2.000000e-01;

% Open mask with disk
radius = 3;
decomposition = 0;
se = strel('disk', radius, decomposition);
BW = imopen(BW, se);

% Close mask with disk
radius = 20;
decomposition = 0;
se = strel('disk', radius, decomposition);
BW = imclose(BW, se);

% Create masked image.
maskedImage = X;
maskedImage(~BW) = 0;

end
```