VIDEO SIGNAL PROCESSING

UEC626

LAB FILE
EXPERIMENTS 6,7,8

SUBMITTED BY

PRANAV YOGESH MAHAJAN

101906108

3EC5

Experiment 6

Software used: MATLAB R2021a

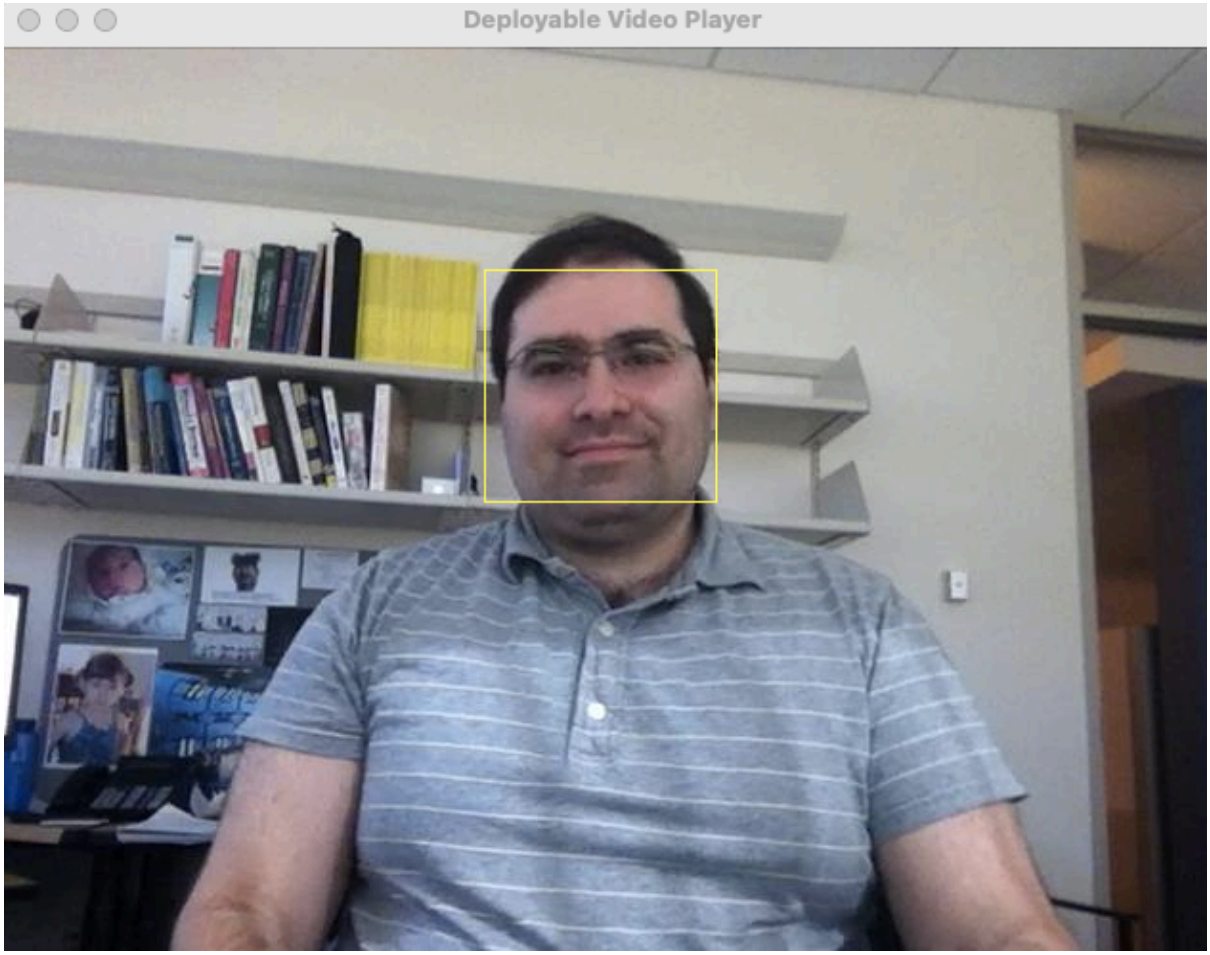
Read, write and display a video using computer vision toolbox. Also detect

Face of a person using, vision.CascadeObjectDetector.

Code:

```
clc;
clear all;
close all hidden;

%Reading file using computer vision toolbox
videoFReader = vision.VideoFileReader('tilted_face.avi');

%Creating a new file to write in
myVideo = VideoWriter('yourFile.avi');
open(myVideo);

%Creating objects for vision.CascadeObjectDetector and
vision.DeployableVideoPlayer
faceDetector = vision.CascadeObjectDetector();
depVideoPlayer = vision.DeployableVideoPlayer;

%Looping for all frames in the video
while ~isDone(videoFReader)
    %Steping one frame each iteration of loop
    videoFrame = step(videoFReader);

    %detecting faces in each frame and marking them
    bbox = faceDetector(videoFrame);
    videoFrame = insertShape(videoFrame, 'Rectangle', bbox);

    %displaying each frame
    depVideoPlayer(videoFrame);

    %writing the final frame
    writeVideo(myVideo, videoFrame);
end
close(myVideo)
```

Output:



Experiment 7

Software used: MATLAB R2021a

Video scene change detection using computer vision toolbox

a) Compute edges of each macro-block using canny edge detector

b) Detect scene changes by computing difference of means of each edge block

c) Display the sequence of identified scene changes along with the edge information

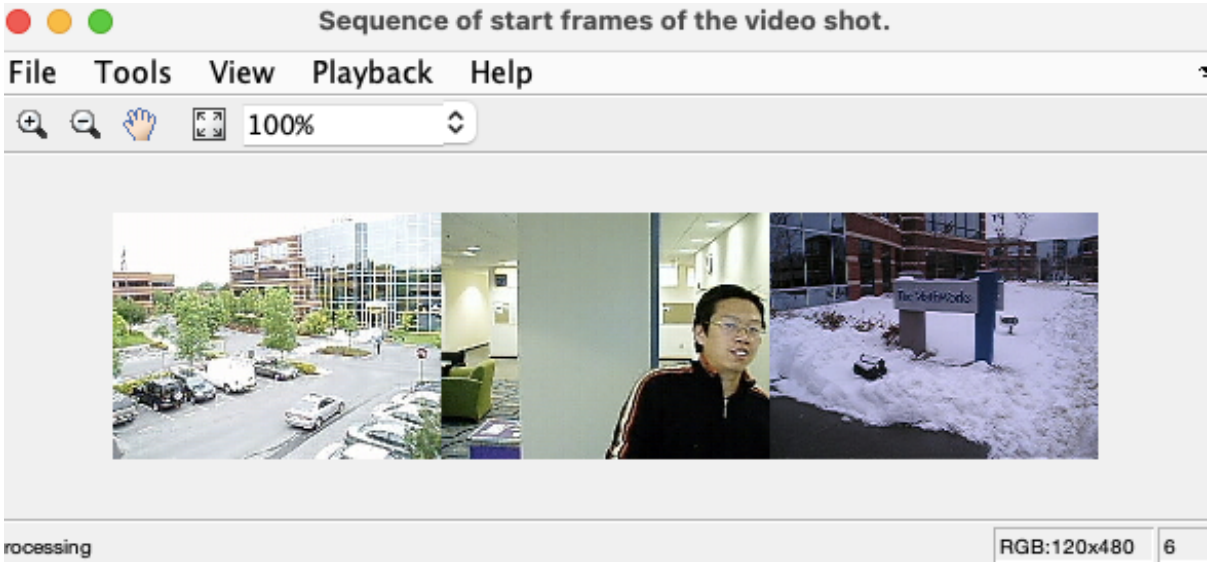
Code:

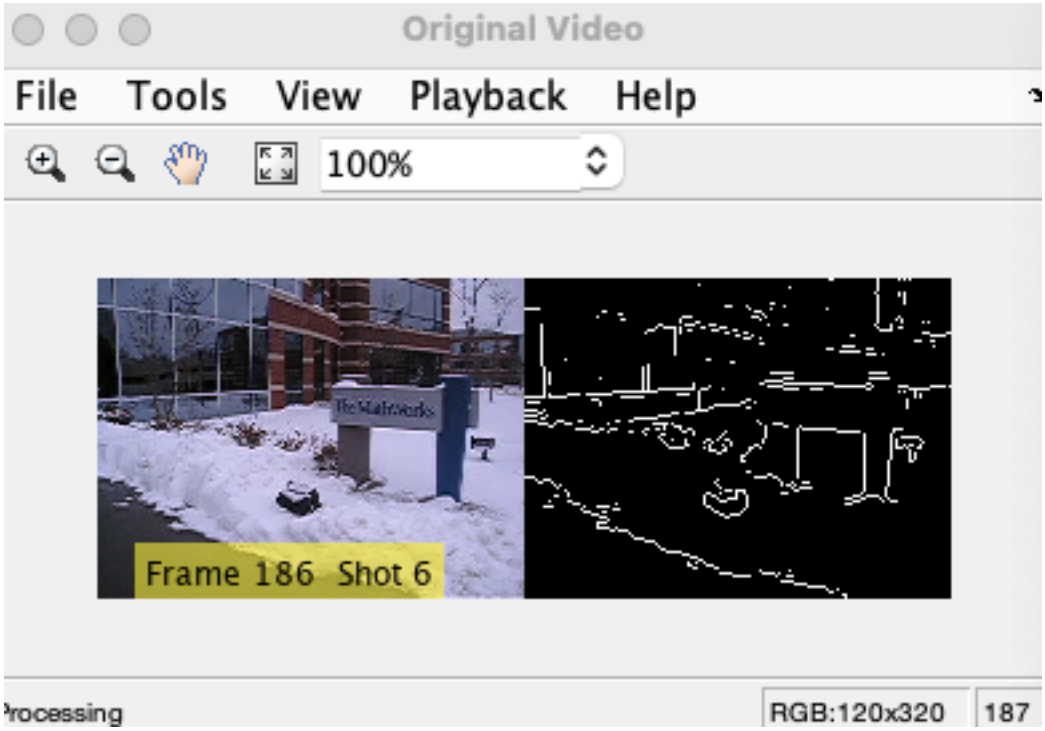
```
clc;
clear all;
close all hidden;

NumTimes = 2;    % Number of times the stream processing loop should
run
%%
% Create a VideoFileReader System object to read video from a file.
hmfr = vision.VideoFileReader( ...
    'Filename', 'vipscenevideoclip.avi', ...
    'PlayCount',  NumTimes);

% Get the dimensions of each frame.
Info = info(hmfr);
rows = Info.VideoSize(2);  % Height in pixels
cols = Info.VideoSize(1);  % Width in pixels
blk_size = 32;  % Block size

% Create ROI rectangle indices for each block in image.
blk_rows = (1:blk_size:rows-blk_size+1);
blk_cols = (1:blk_size:cols-blk_size+1);
[X, Y] = meshgrid(blk_rows, blk_cols);
block_roi = [X(:)'; Y(:)'];
block_roi(3:4, :) = blk_size;
block_roi = block_roi([2 1 4 3], :)';

%%
```

```
% Create an EdgeDetector System object to find out the edges in each
frame.
% hedge = edge( hmfr,'canny');

%%
% Create a Mean System object to calculate the mean of every block
of the
% edge detected image.
hmean = vision.Mean;
hmean.ROIProcessing = true;

%%
% Create VideoPlayer System objects to display the original video
and the
% scene change detected video.
hVideo1 = vision.VideoPlayer;
hVideo1.Name  = 'Original Video';
% Video window position
hVideo1.Position(1) = round(0.4*hVideo1.Position(1));
hVideo1.Position(2) = round(1.5*(hVideo1.Position(2)));
hVideo1.Position([3 4]) = [400 200]; % video window size

hVideo2 = vision.VideoPlayer;
hVideo2.Name  = 'Sequence of start frames of the video shot.';
% Video window position
hVideo2.Position(1) = hVideo1.Position(1) + 410;
hVideo2.Position(2) = round(1.5* hVideo2.Position(2));
hVideo2.Position([3 4]) = [600 200];  % video window size

%% Stream Processing Loop
% Create a processing loop to perform scene change detection in the
input
% video. This loop uses the System objects you instantiated above.

% Initialize variables.
mean_blks_prev = zeros([numel(X), 1], 'single');

scene_out      = zeros([rows, 3*cols, 3], 'single');
count          = 1;
frameCount     = 0;
shotCount      = 0;

while count <= NumTimes
    I = step(hmfr);              % Read input video

    % Calculate the edge-detected image for one video component.
    I_edge = edge( I(:,:,3),'sobel'); %step(hedge, I(:,:,3));

    % Compute mean of every block of the edge image.
    mean_blks = step(hmean, single(I_edge), block_roi);

    % Compare the absolute difference of means between two
consecutive
    % frames against a threshold to detect a scene change.
    edge_diff = abs(mean_blks - mean_blks_prev);
```

```
    edge_diff_b = edge_diff > 0.08;
    num_changed_blocks = sum(edge_diff_b(:));
    % It is a scene change if there is more than one changed block.
    scene_chg = num_changed_blocks > 0.5;

    % Display the sequence of identified scene changes along with
the edges
    % information. Only the start frames of the scene changes are
    % displayed.
    I_out = cat(2, I, repmat(I_edge, [1,1,3]));

    % Display the number of frames and the number of scene changes
detected
    if scene_chg
        shotCount = shotCount + 1;
    end
    txt = sprintf('Frame %3d  Shot %d', frameCount, shotCount);
    I_out = insertText(I_out, [15 100], txt);

    % Generate sequence of scene changes detected
    if scene_chg
        % Shift old shots to left and add new video shot
        scene_out(:, 1:2*cols, :) = scene_out(:, cols+1:end, :);
        scene_out(:, 2*cols+1:end, :) = I;
        step(hVideo2, scene_out); % Display the sequence of scene
changes
    end

    step(hVideo1, I_out);          % Display the Original Video.
    mean_blks_prev = mean_blks;       % Save block mean matrix

    if isDone(hmfr)
        count = count+1;
    end

    frameCount = frameCount + 1;
end

%% Release
% Here you call the release method on the System objects to close
any open
% files and devices.
release(hmfr);

%% Summary
% In the Original Video window you can see the original video and
the edge
% detected version of it. In the window titled 'Sequence of start
frames of
% the video shot' you can see the sequence of the start frames
whenever any
% scene change occurs. The number of shots (scene changes) and the
number
% of frames are displayed in the Original Video window.
displayEndOfDemoMessage(mfilename)
```

Output:


Sequence of start frames of the video shot.
File Tools View Playback Help
100%
RGB:120x480
6


Original Video
File Tools View Playback Help
100%
Frame 186 Shot 6
RGB:120x320
187

Experiment 8

Software used: MATLAB R2021a

Moving object detection in a video using computer vision toolbox

a) First use region-based threshold to remove unwanted objects

b) Use morphological processing to remove small objects from a binary image

c) Get the area and centroid of each remaining object in the frame. Create a copy of the original frame and tag the object by changing the centroid pixel value to red.

Code:

```
clc;
clear all;
close all hidden;

%reading the video
trafficVid = VideoReader('traffic.mj2');
get(trafficVid);
darkCarValue = 50;
darkCar = rgb2gray(read(trafficVid,71));
noDarkCar = imextendedmax(darkCar, darkCarValue);
imshow(darkCar)
figure
imshow(noDarkCar)
sedisk = strel('disk',2);
noSmallStructures = imopen(noDarkCar, sedisk);
imshow(noSmallStructures)
nframes = trafficVid.NumFrames;
I = read(trafficVid, 1);
taggedCars = zeros([size(I,1) size(I,2) 3 nframes], class(I));

for k = 1 : nframes
    singleFrame = read(trafficVid, k);

    % Convert to grayscale to do morphological processing.
    I = rgb2gray(singleFrame);
```

```
    % Remove dark cars.
    noDarkCars = imextendedmax(I, darkCarValue);

    % Remove lane markings and other non-disk shaped structures.
    noSmallStructures = imopen(noDarkCars, sedisk);

    % Remove small structures.
    noSmallStructures = bwareaopen(noSmallStructures, 150);

    % Get the area and centroid of each remaining object in the
frame. The
    % object with the largest area is the light-colored car.  Create
a copy
    % of the original frame and tag the car by changing the centroid
pixel
    % value to red.
    taggedCars(:,:,:,k) = singleFrame;

    stats = regionprops(noSmallStructures, {'Centroid','Area'});
    if ~isempty([stats.Area])
        areaArray = [stats.Area];
        [junk,idx] = max(areaArray);
        c = stats(idx).Centroid;
        c = floor(fliplr(c));
        width = 2;
        row = c(1)-width:c(1)+width;
        col = c(2)-width:c(2)+width;
        taggedCars(row,col,1,k) = 255;
        taggedCars(row,col,2,k) = 0;
        taggedCars(row,col,3,k) = 0;
    end
end

frameRate = trafficVid.FrameRate;
implay(taggedCars,frameRate);
implay('traffic.mj2')
```

Output: