**APPENDIX**

**Codes used video watermarking**

vrd

**Reading of host video normally on Matlab**

%% Read a video into MATLAB

videoFileReader = VideoReader('video 1.mp4');

depVideoPlayer = vision.DeployableVideoPlayer;

%Read frame by frame

while hasFrame(videoFileReader)

videoFrame = readFrame(videoFileReader);

%Display video

depVideoPlayer(videoFrame);

pause(1/videoFileReader.FrameRate);

end

* **Reading and displaying watermark on Black and white, Greyscale and colour Images**

v = 'logo 2.jpg';

originalImage = imread(v);

grayImage = rgb2gray(originalImage);

subplot(1, 2, 1);

imshow(originalImage);

title('Original Image');

subplot(1, 2, 2);

imshow(grayImage);

title('Grayscale Image');

threshold = 128; % Adjust this value as needed

binaryImage = grayImage > threshold;

figure;

imshow(binaryImage);

title('Binary Image');

imwrite(binaryImage, 'binary\_image.jpg');

* **The code for changing videos to sequence of Images in Matlab.**

%%To read the video in matlab, we need to give it a name.

a= VideoReader('video 3.mp4');

get (a)

%% To start the reading of the video frame one by one and labelling it accordingly

a= VideoReader('video 3.mp4');

for img = 1:a.NumberOfFrames;

filename = strcat('frame',num2str(img),'.jpg');

b =read(a,img);

imwrite(b,filename);

end

v2i

Converts video to image data. Gives dimenssion of the video row, column, frame

%%Reading each frame one by one to the last one

VidObj = VideoReader('video 2.mp4');

%Num\_Frames = Vptr.NumberOfFrames;

NFrames = round(VidObj.FrameRate\*VidObj.Duration);

%Find the height and weight of the frame

Nrows = VidObj.height;

Ncols = VidObj.width;

%Preallocate the matrix

Img\_s = zeros([Nrows,Ncols,NFrames]);

for i = 1:NFrames

%Read each frame

Img = readFrame(VidObj);

%To display all the frames

figure,imshow(Img);

end

%Save the matrix as .mat file

Save Video\_Images.mat Img\_s;

Note: The images are extracted into frame numbers from the video and they are sorted numerically.

* **The changing of sequences of images in a folder into video in matlab**

%Make the Below path as the Current Folder

cd('C:\Program Files\Matlab\bin\My Images watermark\videos folder.jpg');

%Obtain all the JPEG format files in the current folder

Files = dir('\*.jpg');

%Find the total number of JPEG files in the Current Folder

NumFiles= size(Files,1);

%Preallocate a 4-D matrix to store the Image Sequence

%Matrix Format : [Height Width 3 Number\_Of\_Images]

Uphold\_Images = uint8(zeros([294 419 3 NumFiles\*2]));

%To write Video File

VideoObj = VideoWriter('React.mp4');

%Number of Frames per Second

VideoObj.FrameRate = 2;

%Define the Video Quality [ 0 to 100 ]

VideoObj.Quality = 60;

count=1;

for i = 1 : NumFiles

%Read the Images in the Current Folder one by one using For Loop

I = imread(Files(i).name);

%The Size of the Images are made same

ResizeImg = imresize(I,[294 419]);

%Each Image is copied 2 times so that in a second 1 image can be viewed

for j = 1 : 2

Uphold\_Images(:,:,:,count)=ResizeImg;

count = count + 1;

end

end

%Open the File 'React.mp4'

open(VideoObj);

%Write the Images into the File 'React.mp4'

writeVideo(VideoObj,Uphold\_Images );

%Close the file 'React.mp4'

close(VideoObj);

* **Adding noise for imperceptible and Capacity for video**

The % Read the video

videoFile = 'video 1.mp4';

videoReader = VideoReader(videoFile);

% Read the watermark image

watermarkFile = 'Logo 2.jpg';

watermark = imread(watermarkFile);

watermark = rgb2gray(watermark); % Convert to grayscale if necessary

watermark = imresize(watermark, [videoReader.Height, videoReader.Width]); % Resize to fit video frame

% Parameters

alpha = 0.5; % Transparency level for watermark (adjust for imperceptibility)

embeddingRate = 0.1; % Adjust according to desired capacity

% Embed watermark into video frames

watermarkedFrames = cell(1, videoReader.NumFrames);

for i = 1:videoReader.NumFrames

frame = read(videoReader, i);

% Embed watermark imperceptibly

watermarkedFrame = (1 - alpha) \* frame + alpha \* watermark \* embeddingRate;

watermarkedFrames{i} = watermarkedFrame;

end

% Display original and watermarked frames

for i = 1:videoReader.NumFrames

figure;

subplot(1, 2, 1);

imshow(read(videoReader, i)); % Display original frame

title('Original Frame');

subplot(1, 2, 2);

imshow(watermarkedFrames{i}); % Display watermarked frame

title('Watermarked Frame');

end

% Simulate an attack by adding noise to the watermarked frames

noisyWatermarkedFrames = cellfun(imnoise(x, 'gaussian', 0, 0.01), watermarkedFrames, 'UniformOutput', false);

% Evaluate imperceptibility and capacity

psnrValues = zeros(1, videoReader.NumFrames);

for i = 1:videoReader.NumFrames

psnrValues(i) = psnr(watermarkedFrames{i}, noisyWatermarkedFrames{i});

end

% Plot PSNR values

figure;

plot(psnrValues);

xlabel('Frame Index');

ylabel('PSNR (dB)');

title('Imperceptibility Analysis');

* **The patchwork Algorithm code for video with graph**

% Specify file paths for the original video, watermark image, and output video

originalVideoPath = 'video\_1.mp4';

watermarkImagePath = 'Logo 1.jpg';

outputVideoPath = 'Wendy.mp4';

% Read the original video and watermark image

originalVideo = VideoReader(originalVideoPath);

watermarkImage = imread(watermarkImagePath);

% Get video properties

frameRate = originalVideo.FrameRate;

frameWidth = originalVideo.Width;

frameHeight = originalVideo.Height;

numFrames = floor(originalVideo.Duration \* frameRate);

% Create a VideoWriter object for the output video

outputVideo = VideoWriter(outputVideoPath, 'MPEG-4');

outputVideo.FrameRate = frameRate;

open(outputVideo);

% Initialize a figure for displaying the graph and videos

figure;

for frameIdx = 1:numFrames

% Read a frame from the original video

originalFrame = readFrame(originalVideo);

% Resize the watermark image to match the frame dimensions

watermarkResized = imresize(watermarkImage, [frameHeight, frameWidth]);

% Create the watermarked frame by overlaying the watermark image

alpha = 0.5; % Adjust the transparency of the watermark

watermarkedFrame = originalFrame;

for c = 1:3

watermarkedFrame(:, :, c) = ...

(1 - alpha) \* double(originalFrame(:, :, c)) + ...

alpha \* double(watermarkResized(:, :, c));

end

watermarkedFrame = uint8(watermarkedFrame);

% Display the graph and videos

subplot(1, 2, 1);

plot(1:frameIdx, 'r'); % Modify this line to plot your own data

title('Graph');

subplot(1, 2, 2);

imshowpair(originalFrame, watermarkedFrame, 'montage');

title('Original vs Watermarked');

% Write the watermarked frame to the output video

writeVideo(outputVideo, watermarkedFrame);

% Pause to simulate real-time display

pause(0.1); % Adjust the duration to control the display speed

end

% Close the output video file

close(outputVideo);

disp('Watermarked video has been created.');

% Optionally, play the output video

outputVideo = VideoReader(outputVideoPath);

while hasFrame(outputVideo)

frame = readFrame(outputVideo);

imshow(frame);

end

* **Getting the Metric Value for Video watermark**

% Load original and watermarked videos

originalVideo = VideoReader('video 2.mp4');

watermarkedVideo = VideoReader('marked.mp4');

% Initialize arrays to store metric values

numFrames = min(originalVideo.NumFrames, watermarkedVideo.NumFrames);

psnrValues = zeros(1, numFrames);

ssimValues = zeros(1, numFrames);

nccValues = zeros(1, numFrames);

scopValues = zeros(1, numFrames);

% Calculate metric values frame by frame

for i = 1:numFrames

% Read frames from original and watermarked videos

frameOriginal = readFrame(originalVideo);

frameWatermarked = readFrame(watermarkedVideo);

% Calculate PSNR

psnrValues(i) = psnr(frameOriginal, frameWatermarked);

% Calculate SSIM

ssimValues(i) = ssim(frameOriginal, frameWatermarked);

% Calculate NCC

nccValues(i) = max(max(normxcorr2(frameOriginal(:,:,1), frameWatermarked(:,:,1))));

% Calculate SCoP (Custom implementation required)

% scopValues(i) = calculateSCoP(frameOriginal, frameWatermarked);

end

% Plotting metric results

figure;

subplot(3,2,1);

plot(psnrValues);

title('PSNR');

xlabel('Frame');

ylabel('PSNR (dB)');

subplot(3,2,2);

plot(ssimValues);

title('SSIM');

xlabel('Frame');

ylabel('SSIM');

subplot(3,2,3);

plot(nccValues);

title('NCC');

xlabel('Frame');

ylabel('NCC');

subplot(3,2,5);

% Plot SCoP results if available

% Display average metric values

fprintf('Average PSNR: %.2f dB\n', mean(psnrValues));

fprintf('Average SSIM: %.4f\n', mean(ssimValues));

fprintf('Average NCC: %.4f\n', mean(nccValues));

fprintf('Average SCoP: %.4f\n', mean(scopValues));

* **How to watermark a video using logo images (Visible watermarking)**

% Read in the video file

vidObj = VideoReader('video 2.mp4');

% Read in the watermark image

watermark = imread('Logo 1.jpg');

watermark\_alpha = 0.5; % alpha value for watermark transparency (adjust as needed)

% Define the location of the watermark in the video frame

watermark\_pos = [50, 50];

% Create a new video writer object

outObj = VideoWriter('marked.mp4', 'MPEG-4');

open(outObj);

% Loop over each frame in the video and add the watermark

while hasFrame(vidObj)

frame = readFrame(vidObj);

% Add the watermark to the frame

watermarked\_frame = frame;

watermarked\_frame(watermark\_pos(1):watermark\_pos(1)+size(watermark, 1)-1, ...

watermark\_pos(2):watermark\_pos(2)+size(watermark, 2)-1, :) = ...

(1-watermark\_alpha)\*watermarked\_frame(watermark\_pos(1):watermark\_pos(1)+size(watermark, 1)-1, ...

watermark\_pos(2):watermark\_pos(2)+size(watermark, 2)-1, :) + ...

watermark\_alpha\*watermark;

% Write the watermarked frame to the new video

writeVideo(outObj, watermarked\_frame);

end

Note: Once you paste the code in your editor, just wait a little, to find the output object, Then implay it, if it throws error relax and redo it or give it another name, it will give the watermarked video with logo.

implay('marked.mp4')

* **File Insertion for Video watermarking**

% Load the original MP4 video file

video = VideoReader('video 2.mp4');

% Determine the number of frames and frame dimensions

numFrames = video.NumberOfFrames;

frameHeight = video.Height;

frameWidth = video.Width;

% Create a new MP4 video file with the inserted file

outputFilename = 'Rested\_day.mp4';

writerObj = VideoWriter(outputFilename, 'MPEG-4');

open(writerObj);

% Loop through each frame of the original video and write it to the new video

for k = 1:numFrames

% Read the current frame

frame = read(video, k);

% Insert the file into the frame (for example, a JPG image file)

if mod(k, 80) == 0 % insert the file every 80 frames

file = imread('Logo 2.jpg');

file = imresize(file, [frameHeight/4, frameWidth/4]); % resize the file to fit a quarter of the frame

[fileHeight, fileWidth, ~] = size(file);

rowStart = randi(frameHeight - fileHeight);

colStart = randi(frameWidth - fileWidth);

frame(rowStart:rowStart+fileHeight-1, colStart:colStart+fileWidth-1, :) = file;

end

% Write the modified frame to the new video

writeVideo(writerObj, frame);

end

% Close the new video file

close(writerObj);

* **The code for audio sound removal using Patchwork Techniques.**

% Read the input video file

videoObj = VideoReader('video 1.mp4');

% Define the patch size and overlap between patches

patchSize = [100, 100];

overlapSize = [25, 25];

% Determine the number of rows and columns of patches

nRows = floor((videoObj.Height - overlapSize(1)) / (patchSize(1) - overlapSize(1)));

nCols = floor((videoObj.Width - overlapSize(2)) / (patchSize(2) - overlapSize(2)));

% Initialize the output video file

outputVideo = VideoWriter('court.avi');

open(outputVideo);

% Iterate through all frames of the video

while hasFrame(videoObj)

% Read the current frame

currentFrame = readFrame(videoObj);

% Initialize the output frame

outputFrame = zeros(videoObj.Height, videoObj.Width, 3, 'uint8');

% Iterate through all patches in the current frame

for iRow = 1:nRows

for iCol = 1:nCols

% Compute the coordinates of the current patch

patchTop = (iRow - 1) \* (patchSize(1) - overlapSize(1)) + 1;

patchBottom = patchTop + patchSize(1) - 1;

patchLeft = (iCol - 1) \* (patchSize(2) - overlapSize(2)) + 1;

patchRight = patchLeft + patchSize(2) - 1;

% Extract the current patch from the input frame

patch = currentFrame(patchTop:patchBottom, patchLeft:patchRight, :);

% Insert the current patch into the output frame

outputFrame(patchTop:patchBottom, patchLeft:patchRight, :) = patch;

end

end

% Write the output frame to the output video file

writeVideo(outputVideo, outputFrame);

end

implay(outputVideo)

% Close the output video file and release the input video object

close(outputVideo);

delete(video**Obj);**

* **Reading the input video file using Stack decomposition filter**

% Read the input video file

videoObj = VideoReader('video 1.mp4');

% Define the number of temporal layers

numLayers = 3;

% Define the stack decomposition filters

filters = cell(numLayers, 1);

for i = 1:numLayers

filters{i} = fspecial('gaussian', [i\*2+1 i\*2+1], i);

end

% Initialize the output video file

outputVideo = VideoWriter('mine.avi');

open(outputVideo);

% Iterate through all frames of the video

while hasFrame(videoObj)

% Read the current frame

currentFrame = readFrame(videoObj);

% Initialize the temporal layers

layers = cell(numLayers, 1);

for i = 1:numLayers

layers{i} = zeros(size(currentFrame));

end

% Apply the stack decomposition filters to the current frame

for i = 1:numLayers

filtered = imfilter(currentFrame, filters{i}, 'replicate', 'same', 'conv');

layers{i} = currentFrame - filtered;

end

% Write the temporal layers to the output video file

for i = 1:numLayers

writeVideo(outputVideo, layers{i});

end

end

disp(outputVideo)

% Close the output video file and release the input video object

close(outputVideo);

delete(videoObj);

* **Reading and extracting of audio and input of video watermark in Matlab**

close all;

clear;

%% Extracting Video & Audio

[videoFilename, videoPath]=uigetfile({'\*.mp4';'\*.avi'},'Video Selector'); %appears GUI to select the video

videoFReader = VideoReader([videoPath '\' videoFilename]); %Extracts the video from the provided path

[audio,fsAudio] = audioread([videoPath '\' videoFilename]); %Extracts the audio from the selected video

fsAudioNew = 2\*fsAudio; %Doubling the sampling rate for audio write process in the next section

%% Watermarking and writing the result to a new video file

%Creating a 'output.avi' video file in the current directory to write the

%frames back to a video after the processing

videoFWriter = vision.VideoFileWriter('output.avi','AudioInputPort',true,'FrameRate',videoFReader.FrameRate);

samplesPerFrame = fsAudioNew/videoFReader.FrameRate; %Calculating number of samples of the audio per frame

%%%%%%% Watermarking Specifications the video and writing the audio for each frame into the

%'output.avi' file %%%%%%%%%%%%%%%%%%%%%%%

watermark = imread('Logo 1.jpg');

watermark\_alpha = 0.5; % alpha value for watermark transparency (adjust as needed)

% Define the location of the watermark in the video frame

watermark\_pos = [50, 50];

frameToWatermark = 10;

% Create a new video writer object

updatedFVideo = vision.VideoFileWriter('water\_marked\_image.avi','AudioInputPort',true,'FrameRate',videoFReader.FrameRate);

for framei=1:videoFReader.NumFrames/2

videoFrame = readFrame(videoFReader); %Extracting the video frame for each frame

% %%%%%%%%%% Watermarking Procedure %%%%%%%%%%%%%%%%%%%%%%%%%%%%

% % Loop over each frame in the video and add the watermark

% while hasFrame(videoFReader)

if (framei == frameToWatermark)

% frame = readFrame(videoFReader);

% Add the watermark to the frame

watermarked\_frame = videoFrame;

watermarked\_frame(watermark\_pos(1):watermark\_pos(1)+size(watermark, 1)-1, ...

watermark\_pos(2):watermark\_pos(2)+size(watermark, 2)-1, :) = ...

(1-watermark\_alpha)\*watermarked\_frame(watermark\_pos(1):watermark\_pos(1)+size(watermark, 1)-1, ...

watermark\_pos(2):watermark\_pos(2)+size(watermark, 2)-1, :) + ...

watermark\_alpha\*watermark;

% Write the watermarked frame to the new video

else

watermarked\_frame = videoFrame;

end

audioPerFrame = audio((framei-1)\*samplesPerFrame+1:samplesPerFrame\*framei,:); %Extracting audio samples per frame

step(updatedFVideo, watermarked\_frame,audioPerFrame); %Writing video, and audio for each frame into the 'output.avi' file

end

release(updatedFVideo); %Releases videoFWriter that creates the 'output.avi' file

**Note**: It displays the watermark video and it shows the logo if it is slowly played but it works better in a higher version of Matlab like 2020 to 2022 version.

* **Reading of RGB Colour to Grayscale video**

%%Changing RGB Colour to Grayscale video in Matlab

filename = 'video 6.mp4';

vid = VideoReader(filename);

newVid = VideoWriter('The real video.avi');

open(newVid);

numFrames = vid.NumberOfFrames;

for frame = 1:numFrames

%Extract the frame from the movie structure

thisFrame = read(vid, frame);

%convert each frame to black and white

gray = rgb2gray(thisFrame);

writeVideo(newVid,gray);

end

close(newVid)

implay('The real video.avi');

* **Changing of RGB video to Red, Blue and Green Channel in a video in Matlab**

% Open the video file

videoFile = 'video 3.mp4';

videoObj = VideoReader(videoFile);

% Create folders to store the RGB channels

outputFolderRed = 'Red\_channel';

outputFolderGreen = 'Green\_channel';

outputFolderBlue = 'Blue\_channel';

if ~exist(outputFolderRed, 'dir')

mkdir(outputFolderRed);

end

if ~exist(outputFolderGreen, 'dir')

mkdir(outputFolderGreen);

end

if ~exist(outputFolderBlue, 'dir')

mkdir(outputFolderBlue);

end

% Read and save each frame's RGB channels

frameIndex = 1;

while hasFrame(videoObj)

% Read the frame

frame = readFrame(videoObj);

% Split the frame into red, green, and blue channels

redChannel = frame(:, :, 1);

greenChannel = frame(:, :, 2);

blueChannel = frame(:, :, 3);

% Save the channels as separate images

outputFilenameRed = fullfile(outputFolderRed, sprintf('frame\_%04d.jpg', frameIndex));

outputFilenameGreen = fullfile(outputFolderGreen, sprintf('frame\_%04d.jpg', frameIndex));

outputFilenameBlue = fullfile(outputFolderBlue, sprintf('frame\_%04d.jpg', frameIndex));

imwrite(redChannel, outputFilenameRed);

imwrite(greenChannel, outputFilenameGreen);

imwrite(blueChannel, outputFilenameBlue);

% Display the original frame and channels

subplot(2, 2, 1);

imshow(frame);

title(sprintf('Frame %d (Original)', frameIndex));

subplot(2, 2, 2);

imshow(redChannel);

title('Red Channel');

subplot(2, 2, 3);

imshow(greenChannel);

title('Green Channel');

subplot(2, 2, 4);

imshow(blueChannel);

title('Blue Channel');

drawnow;

% Increment the frame index

frameIndex = frameIndex + 1;

end

**Common attacks**

* **The code for Adding Gaussian noise to a mp4 video in matlab**

%% Load the input MP4 video

v = VideoReader('video 1.mp4');

% Create a VideoWriter object for the output video

vw = VideoWriter('output.mp4', 'MPEG-4');

vw.FrameRate = v.FrameRate;

% Open the VideoWriter object

open(vw);

% Loop through each frame of the video and apply a Gaussian noise distortion

while hasFrame(v)

% Read the current frame

frame = readFrame(v);

% Apply Gaussian noise distortion to the frame

% For example, add noise with mean 0 and standard deviation 10

noisyFrame = imnoise(frame, 'gaussian', 0, 10^0.001);

% Write the modified frame to the output video

writeVideo(vw, noisyFrame);

end

% Close the VideoWriter object

close(vw);

imshow(noisyFrame)

* **The Code showing denoise frame attack in mp4 video file**

% Load the input MP4 video

v = VideoReader('video 1.mp4');

% Create a VideoWriter object for the output video

vw = VideoWriter('output.mp4', 'MPEG-4');

vw.FrameRate = v.FrameRate;

% Open the VideoWriter object

open(vw);

% Loop through each frame of the video and apply a Gaussian denoising attack

while hasFrame(v)

% Read the current frame

frame = readFrame(v);

% Apply Gaussian denoising to the frame

% For example, remove noise with standard deviation 10

denoisedFrame = imgaussfilt(frame, 10);

% Write the modified frame to the output video

writeVideo(vw, denoisedFrame);

end

% Close the VideoWriter object

close(vw);

imshow(denoisedFrame)

1. **The Quantization attack in mp4 video using matlab**

% Load the input MP4 video

v = VideoReader('video 1.mp4');

% Create a VideoWriter object for the output video

vw = VideoWriter('output.mp4', 'MPEG-4');

vw.FrameRate = v.FrameRate;

% Open the VideoWriter object

open(vw);

% Loop through each frame of the video and apply a color quantization attack

while hasFrame(v)

% Read the current frame

frame = readFrame(v);

% Apply color quantization to the frame

% For example, reduce the color depth to 4 bits

quantizedFrame = imquantize(frame, linspace(0, 1, 2^4));

% Convert the quantized frame to uint8 format for saving to MP4

quantizedFrame = uint8(quantizedFrame \* (256 / 2^4 - 1));

% Write the modified frame to the output video

writeVideo(vw, quantizedFrame);

end

% Close the VideoWriter object

close(vw);

imshow(quantizedFrame)

* **The Compressed Frame in mp4 video in matlab**

% Load the input MP4 video

v = VideoReader('video 1.mp4');

% Create a VideoWriter object for the output video

vw = VideoWriter('output.mp4', 'MPEG-4');

vw.FrameRate = v.FrameRate;

% Open the VideoWriter object

open(vw);

% Set the target resolution for the output video

targetWidth = 640;

targetHeight = 360;

% Loop through each frame of the video and apply a compression attack

while hasFrame(v)

% Read the current frame

frame = readFrame(v);

% Resize the frame to the target resolution

resizedFrame = imresize(frame, [targetHeight, targetWidth]);

% Apply lossy compression to the resized frame

% For example, compress the frame with a quality factor of 50

compressedFrame = imresize(imresize(resizedFrame, 0.5), 2, 'nearest');

% Write the modified frame to the output video

writeVideo(vw, compressedFrame);

end

% Close the VideoWriter object

close(vw);

imshow(compressedFrame)

* **The Geometric attack Frame in mp4 video in matlab**

% Load the input MP4 video

v = VideoReader('video 1.mp4');

% Create a VideoWriter object for the output video

vw = VideoWriter('output.mp4', 'MPEG-4');

vw.FrameRate = v.FrameRate;

% Open the VideoWriter object

open(vw);

% Loop through each frame of the video and apply a geometric attack

while hasFrame(v)

% Read the current frame

frame = readFrame(v);

% Apply a geometric attack to the frame

% For example, scale the frame by a factor of 1.5

scaledFrame = imresize(frame, 1.5);

% Write the modified frame to the output video

writeVideo(vw, scaledFrame);

end

% Close the VideoWriter object

close(vw);

imshow(scaledFrame)

* **Code for extracting video watermark in matlab**

clc

% Specify the input video file

originalVideoFile = 'video 6.mp4';

% Create a VideoReader object for the original video

videoReader = VideoReader(originalVideoFile)

% Specify the frame or time intervals where the watermark videos or images are present

watermarkFrames = [10, 20, 30]; % Example: watermark frames at indices 10, 20, and 30

% Create a directory to save the extracted watermark videos or images

outputDir = 'watermark\_output';

if ~exist(outputDir, 'dir')

mkdir(outputDir);

end

% Extract watermark videos or images

for i = 1:length(watermarkFrames)

% Read the desired frame

frameIndex = watermarkFrames(i);

frame = read(videoReader, frameIndex);

% Generate a unique filename for the watermark video or image

filename = sprintf('watermark\_%d.jpg', frameIndex); % Example: watermark\_10.jpg, watermark\_20.jpg, ...

outputFilePath = fullfile(outputDir, filename);

% Save the frame as a watermark video or image

imwrite(frame, outputFilePath);

end

disp('Watermark videos or images extracted successfully.');

* **Code for Frame Averaging**

%% Frame averaging of video using Matlab

%% Specify the frame window size for averaging

frameWindow = 5; % Number of frames to average

%% Read the video in MATLAB

video = VideoReader('video 1.mp4');

%% Initialize variables

frameCount = 0;

frameSum = zeros(video.Height, video.Width, 3, 'double');

%% Iterate over the frames with frame averaging

while hasFrame(video)

frameCount = frameCount + 1;

% Read the current frame

frame = readFrame(video);

% Accumulate frames within the window

frameSum = frameSum + double(frame);

% Check if the frame window is complete

if frameCount >= frameWindow

% Compute the average frame

averagedFrame = uint8(frameSum / frameWindow);

% Perform your desired operations on the averaged frame here

% Save the averaged frame to an image file

filename = strcat('averaged\_frame', num2str(frameCount), '.jpg');

imwrite(averagedFrame, filename);

% Reset the frame window variables

frameCount = 0;

frameSum = zeros(video.Height, video.Width, 3, 'double');

end

end

% Handle the remaining frames if the total number of frames is not divisible by the window size

if frameCount > 0

averagedFrame = uint8(frameSum / frameCount);

filename = strcat('averaged\_frame', num2str(frameCount), '.jpg');

imwrite(averagedFrame, filename);

end

* **Code for Frame dropping**

%%Frame dropping of videos (images) in matlab

%% Specify the frame interval to drop

frameInterval = 2; % Drop every 2nd frame

%% Read the video in MATLAB

video = VideoReader('video 3.mp4');

%% Iterate over the frames with frame dropping

frameCount = 0;

while hasFrame(video)

frameCount = frameCount + 1;

% Check if the current frame should be dropped

if mod(frameCount, frameInterval) == 0

continue; % Skip this frame

end

% Read and process the frame

frame = readFrame(video);

% Perform your desired operations on the frame here

% Save the frame to an image file

filename = strcat('frame', num2str(frameCount), '.jpg');

imwrite(frame, filename);

end

* **Metric video using Matlab**

% Load the video

videoFile = 'video\_1.mp4';

videoReader = VideoReader(videoFile);

% Load the watermark image

watermarkFile = 'Logo 3.png';

watermark = imread(watermarkFile);

% Create a VideoWriter object for the output video

outputFile = 'Metric videos.mp4';

outputVideo = VideoWriter(outputFile, 'MPEG-4');

open(outputVideo);

% Loop through each frame of the input video

while hasFrame(videoReader)

frame = readFrame(videoReader);

% Resize watermark to match the frame size

watermarkResized = imresize(watermark, [size(frame, 1) size(frame, 2)]);

% Add watermark to the frame

alpha = 0.5; % Adjust watermark transparency

watermarkedFrame = frame;

watermarkedFrame(1:size(watermarkResized, 1), 1:size(watermarkResized, 2), :) = ...

(1 - alpha) \* watermarkedFrame(1:size(watermarkResized, 1), 1:size(watermarkResized, 2), :) + ...

alpha \* watermarkResized;

% Write the watermarked frame to the output video

writeVideo(outputVideo, watermarkedFrame);

end

% Close the output video file

close(outputVideo);

% Calculate and display a metric value (e.g., frame count)

numFrames = videoReader.NumFrames;

disp(['Number of frames in the video: ', num2str(numFrames)]);

* Using video to bit index

% Define input and output file names

inputFile = 'video 2.mp4';

outputFile = 'output.bit';

% Create VideoReader object for input file

videoReader = VideoReader(inputFile);

% Create VideoWriter object for output file

videoWriter = VideoWriter(outputFile);

% Open VideoWriter object

open(videoWriter);

try

% Read frames from input video and write them to output video

while hasFrame(videoReader)

frame = readFrame(videoReader);

writeVideo(videoWriter, frame);

end

% Close VideoWriter object

close(videoWriter);

disp('Video conversion successful!');