```
%{
PROJECT 1 MAIN CODE
%}

clc
clear all
close all

imageType = 1;

if (imageType == 1)
    firstImageNumber = 0;
    lastImageNumber = 484;
    imageFolder = 'EnterExitCrossingPaths2cor';
    imagePrefix = 'EnterExitCrossingPaths2cor';
elseif (imageType == 2)
    firstImageNumber = 1;
    lastImageNumber = 1070;
    imageFolder = 'Office';
    imagePrefix = 'img01_';
elseif (imageType == 3)
    firstImageNumber = 2;
    lastImageNumber = 354;
    imageFolder = 'RedChair';
    imagePrefix = 'advbgst1_21_';
end

imageSuffix = '.jpg';

imgList=[];
for i = firstImageNumber:lastImageNumber
    fullFileName = strcat(imageFolder, '/', imagePrefix,
 sprintf('%04d', i), imageSuffix);
    img = imread(fullFileName);

    red = img(:,:,1); % Red channel
    green = img(:,:,2); % Green channel
    blue = img(:,:,3); % Blue channel
    grayscale = (red + green + blue);

    imgList(:, :, i+1) = grayscale;
end

temporal1DGaussianFilteredThresholds = [110, 127; 150, 177];
temporal1DGaussianFilteredList1 = temporal1DGaussianFilter(imgList,
 1.0, temporal1DGaussianFilteredThresholds);
temporal1DGaussianFilteredList2 = temporal1DGaussianFilter(imgList,
 1.2, temporal1DGaussianFilteredThresholds);
temporal1DGaussianFilteredList3 = temporal1DGaussianFilter(imgList,
 1.5, temporal1DGaussianFilteredThresholds);
temporal1DGaussianFilteredList4 = temporal1DGaussianFilter(imgList,
 2.0, temporal1DGaussianFilteredThresholds);
```

```
simpleFilteredList1 = temporalSimpleFilter(imgList, 10);
simpleFilteredList2 = temporalSimpleFilter(imgList, 15);
simpleFilteredList3 = temporalSimpleFilter(imgList, 20);
boxFilter3 = boxFilter(imgList, 3);
boxFilter5 = boxFilter(imgList, 5);
spacial2DGaussianFilter1 = spacial2DGaussianFilter(imgList, 1.0);
spacial2DGaussianFilter2 = spacial2DGaussianFilter(imgList, 1.2);
spacial2DGaussianFilter3 = spacial2DGaussianFilter(imgList, 1.5);
spacial2DGaussianFilter4 = spacial2DGaussianFilter(imgList, 2);

makeVideo(grayToRgb(temporal1DGaussianFilteredList1), '1D Gaussian
 Temporal STD = 1.0');
makeVideo(grayToRgb(temporal1DGaussianFilteredList2), '1D Gaussian
 Temporal STD = 1.2');
makeVideo(grayToRgb(temporal1DGaussianFilteredList3), '1D Gaussian
 Temporal STD = 1.5');
makeVideo(grayToRgb(temporal1DGaussianFilteredList4), '1D Gaussian
 Temporal STD = 2.0');
makeVideo(grayToRgb(simpleFilteredList1), 'Simple Temporal Threshold =
 10');
makeVideo(grayToRgb(simpleFilteredList2), 'Simple Temporal Threshold =
 15');
makeVideo(grayToRgb(simpleFilteredList3), 'Simple Temporal Threshold =
 20');
makeVideo(grayToRgb(temporalSimpleFilter(spacial2DGaussianFilter1,
 15)), '2D Gaussian STD = 1.0'); % the best video
makeVideo(grayToRgb(temporalSimpleFilter(spacial2DGaussianFilter2,
 15)), '2D Gaussian STD = 1.2');
makeVideo(grayToRgb(temporalSimpleFilter(spacial2DGaussianFilter3,
 15)), '2D Gaussian STD = 1.5');
makeVideo(grayToRgb(temporalSimpleFilter(spacial2DGaussianFilter4,
 15)), '2D Gaussian STD = 2');
makeVideo(grayToRgb(temporalSimpleFilter(boxFilter3, 15)), 'Box Filter
 3');
makeVideo(grayToRgb(temporalSimpleFilter(boxFilter5, 15)), 'Box Filter
 5');
makeVideo(grayToRgb(temporal1DGaussianFilter(spacial2DGaussianFilter1,
 1.2, temporal1DGaussianFilteredThresholds)), '2D Gaussian STD = 1.0
 with 1D Gaussian Temporal STD = 1.2');
makeVideo(grayToRgb(imgList), 'Grayscale Original Video')
```

Published with MATLAB® R2017a

```
%{
TEMPORAL SIMPLE FILTER
%}

function [thresholdDerivativeList] = temporalSimpleFilter(imgList,
 threshold)
    numImages = size(imgList);
    numImages = numImages(3) - 1;
    derivativeList=zeros(size(imgList));
    for i = 2:numImages
        derivativeList(:, :, i) = (imgList(:, :, i+1) - imgList(:, :,
 i-1)) / 2;
    end

    absDerivativeList = abs(derivativeList);
    thresholdDerivativeList = zeros(size(absDerivativeList));
    thresholdDerivativeList(absDerivativeList > threshold) = 1;
end
```

Published with MATLAB® R2017a

```
%{
TEMPORAL 1D GAUSSIAN FILTER
%}

function [thresholdGaussianFilteredList] =
 temporal1DGaussianFilter(imgList, tsigma, thresholds)
    imgListSize = size(imgList);
    numImages = imgListSize(3);
    gaussianFilteredList=zeros(size(imgList));
    filterLength = ceil(((tsigma * 5) + 1)/2)*2 - 1;

    startImage = ceil(filterLength/2);
    endImage = numImages - floor(filterLength/2);

    Gx = @(x) 1 / sqrt(2 * pi * (tsigma^2)) * exp(-1 * x^2 / (2 *
 (tsigma^2)));

    gaussianFilter = zeros(1, filterLength);
    for i = 1:filterLength
        x = i - ceil(filterLength/2);
        gaussianFilter(i) = Gx(x);
    end
    gaussianFilterSum = sum(gaussianFilter);

    for i = startImage:endImage
        gaussianFilteredFrame = zeros(imgListSize(1), imgListSize(2));
        for j = 1:filterLength
            currentFilterFrame = (i + j - (ceil(filterLength/2)));
            gaussianFilteredFrame = gaussianFilteredFrame +
 (gaussianFilter(j) * imgList(:,:,currentFilterFrame));
        end
        gaussianFilteredList(:,:,i) = gaussianFilteredFrame /
 gaussianFilterSum;
    end

    thresholdGaussianFilteredList = zeros(size(gaussianFilteredList));
    for i = 1:length(thresholds)
        thresholdGaussianFilteredList(gaussianFilteredList >
 thresholds(i, 1) & gaussianFilteredList < thresholds(i, 2)) = 255;
    end

    figure
    histogram(gaussianFilteredList(:,:,:))
end
```

Published with MATLAB® R2017a

```
%{
BOX FILTER
%}

function [boxFilterList] = boxFilter(imgList,boxSize)
    imgListSize = size(imgList);
    xDir = imgListSize(2);
    yDir = imgListSize(1);
    zDir = imgListSize(3);
    startPixel = boxSize - floor(boxSize/2);
    endPixelX = xDir - floor(boxSize/2);
    endPixelY = yDir - floor(boxSize/2);
    boxFilterList = imgList;

    boxFilterX = (1/boxSize) * ones(1,boxSize);
    boxFilterY = transpose(boxFilterX);
    for k = 1:zDir
        boxFilterList(startPixel:endPixelY, startPixel:endPixelX,
 k) = conv2(conv2(imgList(:,:,k), boxFilterX, 'valid'),
 boxFilterY, 'valid');
    end
end
```

Published with MATLAB® R2017a

```
%{
SPACIAL 2D GAUSSIAN FILTER
%}

function [gaussianFilterList] = spacial2DGaussianFilter(imgList,
 ssigma)
    Gx = @(x) 1 / sqrt(2 * pi * (ssigma^2)) * exp(-1 * x^2 / (2 *
 (ssigma^2)));
    Gy = @(y) 1 / sqrt(2 * pi * (ssigma^2)) * exp(-1 * y^2 / (2 *
 (ssigma^2)));
    imgListSize = size(imgList);
    xDir = imgListSize(2);
    yDir = imgListSize(1);
    zDir = imgListSize(3);
    boxSize = ceil(((ssigma * 5) + 1)/2)*2 - 1;
    startPixel = boxSize - floor(boxSize/2);
    endPixelX = xDir - floor(boxSize/2);
    endPixelY = yDir - floor(boxSize/2);
    gaussianFilterList = imgList;

    gaussianFilterHorizontal = zeros(1, boxSize);
    for i = 1:boxSize
        x = i - ceil(boxSize/2);
        gaussianFilterHorizontal(i) = Gx(x);
    end
    gaussianFilterVertical = transpose(gaussianFilterHorizontal);
    gaussianFilterSumX = sum(gaussianFilterHorizontal);
    gaussianFilterSumY = sum(gaussianFilterVertical);

    gaussianFilterSum = gaussianFilterSumX * gaussianFilterSumY;

    for k = 1:zDir
        gaussianFilterList(startPixel:endPixelY,
 startPixel:endPixelX, k) = (1/gaussianFilterSum) *
 (conv2(conv2(imgList(:,:,k), gaussianFilterHorizontal, 'valid'),
 gaussianFilterVertical, 'valid'));
    end
end
```

Published with MATLAB® R2017a

```
%{
GRAY TO RGB
%}

function [rgbImages] = grayToRgb(grayImages)
    % normalize the gray images from 255 -> 1, if necessary
    if (max(grayImages(:)) > 1 && max(grayImages(:)) <= 255)
        grayImages = grayImages / 255;
    end

    grayImagesSize = size(grayImages);
    numGrayImages = grayImagesSize(3);
    for i = 1:numGrayImages
        rgbImages(:, :, :, i) = cat(3, grayImages(:, :, i),
 grayImages(:, :, i), grayImages(:, :, i));
    end
end
```

Published with MATLAB® R2017a

```
%{
MAKE VIDEO
%}

function [] = makeVideo( images, name )
    % create the video writer with 30fps
    writerObj = VideoWriter(strcat(name,'.avi'));
    writerObj.FrameRate = 30;
    % open the video writer
    open(writerObj);
    % write the frames to the video
    for i=1:length(images)
        % convert the image to a frame
        frame = im2frame(images(:,:,:,i));
        writeVideo(writerObj, frame);
    end
    % close the writer object
    close(writerObj);
end
```

Published with MATLAB® R2017a