```
clc
clear all
close all

warning off; % warnings were spawning from the linsolve(A,B) calls

tic

imageType = 4;
gaussianSmoothingSigma = 1.4;
CMatrixWindowSize = 3;

% calculates the flow vector between imageNum and imageNum+1
% so, imageNum of 1 calculates the flow vector between images 1 and 2
imageNum = 1;

if (imageType == 1)
    folderName = 'Images/toy1/';
    fileNamePrefix = 'toys';
    fileNameSuffix = '.gif';
    fileNumberMin = 1;
    fileNumberMax = 3;
elseif (imageType == 2)
    folderName = 'Images/toy2/';
    fileNamePrefix = 'toys2';
    fileNameSuffix = '.gif';
    fileNumberMin = 1;
    fileNumberMax = 3;
elseif (imageType == 3)
    folderName = 'Images/LKTest1/';
    fileNamePrefix = 'LKTest1im';
    fileNameSuffix = '.pgm';
    fileNumberMin = 1;
    fileNumberMax = 2;
elseif (imageType == 4)
    folderName = 'Images/LKTest2/';
    fileNamePrefix = 'LKTest2im';
    fileNameSuffix = '.pgm';
    fileNumberMin = 1;
    fileNumberMax = 2;
elseif (imageType == 5)
    folderName = 'Images/LKTest3/';
    fileNamePrefix = 'LKTest3im';
    fileNameSuffix = '.pgm';
    fileNumberMin = 1;
    fileNumberMax = 2;
end

grayImgList = [];

numImages = 0;
for i = fileNumberMin:fileNumberMax
```

```
    fullFileName = strcat(folderName, fileNamePrefix, sprintf('%d',
 i), fileNameSuffix);
    img = imread(fullFileName);

    numImages = numImages + 1;
    grayImgList(:, :, numImages) = img;
end

gaussianFilterList = spacial2DGaussianFilter(grayImgList,
 gaussianSmoothingSigma);

[Iy, Ix] = prewittFilter(gaussianFilterList(:,:,imageNum+1));

It = gaussianFilterList(:,:,imageNum+1) -
 gaussianFilterList(:,:,imageNum);

[CMatrix, TMatrix] = CMatrix(Iy, Ix, It, CMatrixWindowSize);

[u, v] = flowVector(CMatrix, TMatrix);

% uncomment this line to overlay the flow vector on top of the first
 image
% imshow(grayImgList(:,:,imageNum)/255)

hold on
quiver(u,v)
hold off
```

Published with MATLAB® R2017a

```
function [gaussianFilterList] = spacial2DGaussianFilter(imgList,
 ssigma)
%UNTITLED5 Summary of this function goes here
%   Detailed explanation goes here
    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; % gets an odd box size

    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

```
function [verticalPrewitt, horizontalPrewitt] = prewittFilter(image)
%UNTITLED4 Summary of this function goes here
%   Detailed explanation goes here
verticalPrewittFilter = [-1 0 1;
                         -1 0 1;
                         -1 0 1];

horizontalPrewittFilter = [-1 -1 -1;
                           0 0 0;
                           1 1 1];

verticalPrewitt = imfilter(image, verticalPrewittFilter, 'replicate');
horizontalPrewitt = imfilter(image,
 horizontalPrewittFilter, 'replicate');

end
```

Published with MATLAB® R2017a

```
function [C, T] = CMatrix(verticalPrewitt, horizontalPrewitt, It,
 boxFilterSize)

verticalHorizontalPrewitt = verticalPrewitt .* horizontalPrewitt;
verticalPrewittSquared = verticalPrewitt .* verticalPrewitt;
horizontalPrewittSquared = horizontalPrewitt .* horizontalPrewitt;

verticalTemporal = verticalPrewitt .* It;
horizontalTemporal = horizontalPrewitt .* It;

boxFilter = ones(boxFilterSize, boxFilterSize)./(boxFilterSize *
 boxFilterSize);

boxFilterVerticalHorizontalPrewitt =
 imfilter(verticalHorizontalPrewitt, boxFilter);
boxFilterVerticalPrewitt = imfilter(verticalPrewittSquared,
 boxFilter);
boxFilterHorizontalPrewitt = imfilter(horizontalPrewittSquared,
 boxFilter);
boxFilterVerticalTemporal = imfilter(verticalTemporal, boxFilter);
boxFilterHorizontalTemporal = imfilter(horizontalTemporal, boxFilter);

sizeMatrix = size(verticalHorizontalPrewitt);
sizeX = sizeMatrix(1);
sizeY = sizeMatrix(2);

C = zeros(sizeX, sizeY, 2, 2);
T = zeros(sizeX, sizeY, 2, 1);
for i = 1:sizeX
    for j = 1:sizeY
        C(i,j,:,:) = [boxFilterHorizontalPrewitt(i, j)
 boxFilterVerticalHorizontalPrewitt(i, j);
            boxFilterVerticalHorizontalPrewitt(i, j)
 boxFilterVerticalPrewitt(i, j)];
       T(i,j,:,:) = [boxFilterHorizontalTemporal(i, j);
 boxFilterVerticalTemporal(i, j)];
    end
end

end
```

Published with MATLAB® R2017a

```
function [u, v] = flowVector(CMatrix, TMatrix)

% invert the T matrix for the calculation of u and v
TMatrix = -1 * TMatrix;

sizeMatrix = size(CMatrix);
sizeX = sizeMatrix(1);
sizeY = sizeMatrix(2);
u = zeros(sizeX, sizeY);
v = zeros(sizeX, sizeY);

for i = 1:sizeX
    for j = 1:sizeY
        %x = CMatrix(i,j,:,:)\TMatrix(i,j,:);
        C(:,:) = CMatrix(i,j,:,:);
        T(:,:) = TMatrix(i,j,:);
        %T = transpose(T);
        x = C\T;
%         x2 = inv(C'*C)*C'*T;
        u(i,j) = x(1);
        v(i,j) = x(2);
    end
end

end
```

Published with MATLAB® R2017a