```
clc
clear all
close all

imageType = 2;

if (imageType == 1)
    folderName = 'Images/cast/';
    fileNamePrefix = 'cast-';
    fileName1 = 'left';
    fileName2 = 'right';
    fileNameExtension = '.jpg';
    rThreshold = 50000000;
    nccThreshold = 1.04;
elseif (imageType == 2)
    folderName = 'Images/Cones/';
    fileNamePrefix = 'Cones_im';
    fileName1 = '2';
    fileName2 = '6';
    fileNameExtension = '.JPG';
    rThreshold = 100000000;
    nccThreshold = 1.04;
else
    error('Cannot select image with the given image type.')
end

fullFileName1 = strcat(folderName, fileNamePrefix, fileName1,
 fileNameExtension);
fullFileName2 = strcat(folderName, fileNamePrefix, fileName2,
 fileNameExtension);
image1 = imread(fullFileName1);
image2 = imread(fullFileName2);

% we need doubles so that we can perform necessary calculations later
rgbImgList(:, :, :, 1) = double(image1);
rgbImgList(:, :, :, 2) = double(image2);
grayImgList(:, :, 1) = double(rgb2gray(image1));
grayImgList(:, :, 2) = double(rgb2gray(image2));

% Harris Corner Detection, Non-Max Suppression and NCC
[corrX, corrY] = getCorners(grayImgList(:,:,1), grayImgList(:,:,2),
 rThreshold, nccThreshold);

plotCorrelations(rgbImgList(:,:,:,1), rgbImgList(:,:,:,2), corrX,
 corrY);

% Calculate the fundamental matrix
[matchedPoints1, matchedPoints2] = getMatchedPoints(corrX, corrY);
F = estimateFundamentalMatrix(matchedPoints1, matchedPoints2);

% Perform RANSAC to find the inliers
sizeImageX = size(grayImgList(:, :, 1), 1);
```

```
sizeImageY = size(grayImgList(:, :, 1), 2);
[inliersX, inliersY] = ransac(F, matchedPoints1, matchedPoints2,
 sizeImageX, sizeImageY);

figure
plotCorrelations(rgbImgList(:,:,:,1), rgbImgList(:,:,:,2), inliersX,
 inliersY);

grayImgList = uint8(grayImgList);
[disparityMapHorizontal, disparityMapVertical] =
 getDisparityMap(grayImgList(:,:,1), grayImgList(:,:,2));

figure
imshow(disparityMapHorizontal, [0 255]);
figure
imshow(disparityMapVertical, [0 255]);
```

Published with MATLAB® R2017a

```
function [corrX, corrY] = getCorners(image1, image2, rThreshold,
 nccThreshold)
    verticalPrewittFilter = [-1 0 1;
                             -1 0 1;
                             -1 0 1];

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

    % Harris Corner Detection
    verticalPrewitt1 = imfilter(image1,
 verticalPrewittFilter, 'replicate');
    horizontalPrewitt1 = imfilter(image1,
 horizontalPrewittFilter, 'replicate');

    verticalPrewitt2 = imfilter(image2,
 verticalPrewittFilter, 'replicate');
    horizontalPrewitt2 = imfilter(image2,
 horizontalPrewittFilter, 'replicate');

    R1 = CMatrix(verticalPrewitt1, horizontalPrewitt1);
    R2 = CMatrix(verticalPrewitt2, horizontalPrewitt2);

    magnitude1 = sqrt(horizontalPrewitt1.^2 + verticalPrewitt1.^2);
    orientation1 = atand(horizontalPrewitt1./verticalPrewitt1);
    magnitude2 = sqrt(horizontalPrewitt2.^2 + verticalPrewitt2.^2);
    orientation2 = atand(horizontalPrewitt2./verticalPrewitt2);

    R1(R1 < rThreshold) = 0;
    R1(R1 >= rThreshold) = 255;
    orientation1(isnan(orientation1)) = 90; % get rid of bad
 orientation values

    R2(R2 < rThreshold) = 0;
    R2(R2 >= rThreshold) = 255;
    orientation2(isnan(orientation2)) = 90; % get rid of bad
 orientation values

    % Non-Max Suppression
    nonMaxSuppress1 = nonMaxSuppression(orientation1, magnitude1, R1);
    nonMaxSuppress2 = nonMaxSuppression(orientation2, magnitude2, R2);

    % NCC
    [corrX, corrY] = normalizedCrossCorrelation(image1, image2,
 nonMaxSuppress1, nonMaxSuppress2, nccThreshold);
end
```

Published with MATLAB® R2017a

```
function [DetMatrix] = CMatrix(verticalPrewitt, horizontalPrewitt)
    verticalHorizontalPrewitt = verticalPrewitt .* horizontalPrewitt;

    verticalPrewittSquared = verticalPrewitt .* verticalPrewitt;

    horizontalPrewittSquared = horizontalPrewitt .* horizontalPrewitt;

    boxFilterSize = 7;
    boxFilter = ones(boxFilterSize, boxFilterSize)./(boxFilterSize *
 boxFilterSize);

    boxFilterVerticalHorizontalPrewitt =
 imfilter(verticalHorizontalPrewitt, boxFilter);
    boxFilterVerticalPrewitt = imfilter(verticalPrewittSquared,
 boxFilter);
    boxFilterHorizontalPrewitt = imfilter(horizontalPrewittSquared,
 boxFilter);

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

    DetMatrix = zeros(sizeX, sizeY);
    for i = 1:sizeX
        for j = 1:sizeY
            C = [boxFilterVerticalPrewitt(i, j)
 boxFilterVerticalHorizontalPrewitt(i, j);
                boxFilterVerticalHorizontalPrewitt(i, j)
 boxFilterHorizontalPrewitt(i, j)];
            D = eig(C);
            det = D(1) * D(2);
            trace = D(1) + D(2);
            DetMatrix(i,j) = det - (0.05 * (trace^2));
        end
    end
end
```

Published with MATLAB® R2017a

```
function [outputMatrix] = nonMaxSuppression(orientation, magnitude, R)
    totalSize = size(orientation);
    sizeX = totalSize(1);
    sizeY = totalSize(2);

    outputMatrix = R;

    for i = 2:sizeX-1
        for j = 2:sizeY-1
            angle = closestAngle(orientation(i,j));
            if (angle == 0)
                compareIndex1 = magnitude(i, j-1);
                compareIndex2 = magnitude(i, j+1);
            elseif (angle == 45)
                compareIndex1 = magnitude(i-1, j+1);
                compareIndex2 = magnitude(i+1, j-1);
            elseif (angle == 90)
                compareIndex1 = magnitude(i-1, j);
                compareIndex2 = magnitude(i+1, j);

            elseif (angle == 135)
                compareIndex1 = magnitude(i-1, j-1);
                compareIndex2 = magnitude(i+1, j+1);
            end
            if (or((magnitude(i,j) < compareIndex1),(magnitude(i,j) <
 compareIndex2)))
                outputMatrix(i,j) = 0;
            else
                outputMatrix(i,j) = R(i,j);
            end
        end
    end
end
```

Published with MATLAB® R2017a

```
function [angle] = closestAngle(orientation)
    compareArray = [0 45 90 135];
    if (abs(mod(orientation,180)) >= (180-22.5))
        angle = 0;
    else
        for i = 1:4
            if (abs(mod(orientation,180)-compareArray(i)) <= 22.5)
                angle = compareArray(i);
            end
        end
    end
end
```

Published with MATLAB® R2017a

```
function [corrX, corrY] = normalizedCrossCorrelation(grayImg1,
 grayImg2, nonMaxSuppress1, nonMaxSuppress2, nccThreshold)
    nccBoxFilterSize = 11;

    totalImageSize1 = size(grayImg1);
    imageHeight1 = totalImageSize1(1);
    imageWidth1 = totalImageSize1(2);
    totalImageSize2 = size(grayImg2);
    imageHeight2 = totalImageSize2(1);
    imageWidth2 = totalImageSize2(2);

    nccImg1 = zeros(size(grayImg1));
    nccImg2 = zeros(size(grayImg2));

    halfNccBoxFilterSizeFloored = floor(nccBoxFilterSize / 2);
    xMin = nccBoxFilterSize - halfNccBoxFilterSizeFloored;
    yMin = xMin;
    xMax1 = imageHeight1 - halfNccBoxFilterSizeFloored;
    yMax1 = imageWidth1 - halfNccBoxFilterSizeFloored;
    xMax2 = imageHeight2 - halfNccBoxFilterSizeFloored;
    yMax2 = imageWidth2 - halfNccBoxFilterSizeFloored;

    for x = xMin:xMax1
        for y = yMin:yMax1
            if (nonMaxSuppress1(x, y) == 255)
                xRange = x - halfNccBoxFilterSizeFloored : x +
 halfNccBoxFilterSizeFloored;
                yRange = y - halfNccBoxFilterSizeFloored : y +
 halfNccBoxFilterSizeFloored;
                grayImg1Snippet = grayImg1(xRange, yRange);
                nccImg1(x, y) = norm(grayImg1Snippet);
            end
        end
    end

    for x = xMin:xMax2
        for y = yMin:yMax2
            if (nonMaxSuppress2(x, y) == 255)
                xRange = x - halfNccBoxFilterSizeFloored : x +
 halfNccBoxFilterSizeFloored;
                yRange = y - halfNccBoxFilterSizeFloored : y +
 halfNccBoxFilterSizeFloored;
                grayImg2Snippet = grayImg2(xRange, yRange);
                nccImg2(x, y) = norm(grayImg2Snippet);
            end
        end
    end

    corrX = zeros(size(grayImg1));
    corrY = zeros(size(grayImg1));

    for x1 = xMin:xMax1
```

```
        for y1 = yMin:yMax1
            if (nonMaxSuppress1(x1,y1) == 255)
                x1Range = x1 - halfNccBoxFilterSizeFloored : x1 +
halfNccBoxFilterSizeFloored;
                y1Range = y1 - halfNccBoxFilterSizeFloored : y1 +
halfNccBoxFilterSizeFloored;
                grayImg1Snippet = grayImg1(x1Range, y1Range);

                c = zeros(size(grayImg2));

                for x2 = xMin:xMax2
                    for y2 = yMin:yMax2
                        if (nonMaxSuppress2(x2,y2) == 255)
                            x2Range = x2 -
halfNccBoxFilterSizeFloored : x2 + halfNccBoxFilterSizeFloored;
                            y2Range = y2 -
halfNccBoxFilterSizeFloored : y2 + halfNccBoxFilterSizeFloored;
                            grayImg2Snippet = grayImg2(x2Range,
y2Range);

                            ccPoint = grayImg1Snippet .*
grayImg2Snippet;
                            nccPoint = sum(sum(ccPoint)) /
(nccImg1(x1,y1) * nccImg2(x2,y2));
                            c(x2,y2) = nccPoint;
                        end
                    end
                end

                [cMaxVal, cMaxIndex] = max(c(:));
                if (cMaxVal > nccThreshold)
                    [cMaxX, cMaxY] = ind2sub(size(c), cMaxIndex);
                    corrX(x1,y1) = cMaxX;
                    corrY(x1,y1) = cMaxY;
                end
            end
        end
    end
end
```

Published with MATLAB® R2017a

```
function [] = plotCorrelations(grayImg1, grayImg2, corrX, corrY)

totalImageSize1 = size(grayImg1);
imageHeight1 = totalImageSize1(1);
imageWidth1 = totalImageSize1(2);

grayImgFull = horzcat(grayImg1, grayImg2);

totalImageSizeFull = size(grayImgFull);
imageHeightFull = totalImageSizeFull(1);
imageWidthFull = totalImageSizeFull(2);

imshow(grayImgFull/255)
hold on
for x = 1:imageHeight1
    for y = 1:imageWidth1
        if ((corrX(x,y) ~= 0) && (corrY(x,y) ~= 0))
            img2XCoordinate = imageWidth1 + corrY(x,y);
            line([y img2XCoordinate], [x
 corrX(x,y)], 'Color', 'yellow')
        end
    end
end

axis([1 imageWidthFull 1 imageHeightFull])
daspect([1 1 1])

hold off

end
```

Published with MATLAB® R2017a

```
function [ matchedPoints1, matchedPoints2 ] = getMatchedPoints(corrX,
 corrY)

numMatchedPoints = sum(sum(corrX > 0));
matchedPoints1 = zeros(numMatchedPoints, 2);
matchedPoints2 = zeros(numMatchedPoints, 2);

sizeX = size(corrX, 1);
sizeY = size(corrX, 2);

matchedPointsCount = 0;
for i = 1:sizeX
    for j = 1:sizeY
        if (corrX(i,j) > 0 && corrY(i,j) > 0)
            matchedPointsCount = matchedPointsCount + 1;
            matchedPoints1(matchedPointsCount, :) = [i j];
            matchedPoints2(matchedPointsCount, :) = [corrX(i,j)
 corrY(i,j)];
        end
    end
end

end
```

Published with MATLAB® R2017a

```
function [inliersX, inliersY] = ransac(F, matchedPoints1,
 matchedPoints2, imageSizeX, imageSizeY)

    [inliers1, inliers2, fValue] = calculateNumCorrectPoints(F,
 matchedPoints1, matchedPoints2);

    totalInliersSize = size(inliers1);
    inliersLength = totalInliersSize(1);

    inliersX = zeros(imageSizeX, imageSizeY);
    inliersY = zeros(imageSizeX, imageSizeY);

    for k = 1:inliersLength
        x = inliers1(k, 1);
        y = inliers1(k, 2);

        inliersX(x,y) = inliers2(k,1);
        inliersY(x,y) = inliers2(k,2);
    end
end
```

Published with MATLAB® R2017a

```
function [inliers1, inliers2, fValues] = calculateNumCorrectPoints(F,
 points1, points2)

numPoints = size(points1, 1);
fValues = zeros(numPoints, 1);

fThreshold = .002;

j = 0;
for i = 1:numPoints
    p1 = [points1(i, :) 1];
    p2 = [points2(i, :) 1];

    fVal = p2 * F * p1';
    fValues(i, :) = fVal;

    if (abs(fVal) < fThreshold)
        j = j + 1;
        inliers1(j, :) = points1(i, :);
        inliers2(j, :) = points2(i, :);
    end
end
```

Published with MATLAB® R2017a

```
function [disparityMapHorizontal, disparityMapVertical] =
 getDisparityMap(image1, image2)

    disparityMapHorizontal = disparity(image1,
 image2, 'Method', 'SemiGlobal');

    rotatedImage1 = imrotate(image1, 90);
    rotatedImage2 = imrotate(image2, 90);
    disparityMapVertical = disparity(rotatedImage1,
 rotatedImage2, 'Method', 'SemiGlobal');

    disparityMapVertical = imrotate(disparityMapVertical, -90);

    % normalize the disparity map between 0-255
    disparityMapHorizontal(disparityMapHorizontal < 0) = 0;
    disparityMapVertical(disparityMapVertical < 0) = 0;
    disparityMapHorizontal = normalizeMatrix(disparityMapHorizontal,
 0, 255);
    disparityMapVertical = normalizeMatrix(disparityMapVertical, 0,
 255);
end
```

Published with MATLAB® R2017a

```
function [ normalizedMatrix ] = normalizeMatrix(matrix, normMin,
 normMax)

    matrixMin = min(min(matrix));
    matrixMax = max(max(matrix));
    matrixRange = matrixMax - matrixMin;

    matrix0To1 = (matrix - matrixMin) / matrixRange;

    normRange = normMax - normMin;
    normalizedMatrix = (matrix0To1 * normRange) + normMin;

end
```

Published with MATLAB® R2017a