```
clc
clear all
close all

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

tic

imageType = 1;

imageScale = 1;

if (imageType == 1)
    folderName = 'Images/DanaHallway1/';
    fileNamePrefix = 'DSC_0';
    fileNameSuffix = '.JPG';
    fileNumberMin = 281;
    fileNumberMax = 283;
    rThreshold = 10000000;
    nccThreshold = 1.04;
elseif (imageType == 2)
    folderName = 'Images/DanaHallway2/';
    fileNamePrefix = 'DSC_0';
    fileNameSuffix = '.JPG';
    fileNumberMin = 285;
    fileNumberMax = 287;
    rThreshold = 10000000;
    nccThreshold = 1.03;
elseif (imageType == 3)
    folderName = 'Images/DanaOffice/';
    fileNamePrefix = 'DSC_0';
    fileNameSuffix = '.JPG';
    fileNumberMin = 308;
    fileNumberMax = 310;
    rThreshold = 100000000;
    nccThreshold = 1.07;
end

grayImgList = [];

numImages = 0;
for i = fileNumberMin:fileNumberMax
    fullFileName = strcat(folderName, fileNamePrefix, sprintf('%3d',
 i), fileNameSuffix);
    img = imread(fullFileName);

    numImages = numImages + 1;
    rgbImgList(:, :, :, numImages) = double(img); % we need doubles so
 that we can perform necessary calculations later
    grayImgList(:, :, numImages) = rgb2gray(img);
end
```

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

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

% RANSAC
[ roughH, roughInliersX, roughInliersY ] = ransac2(grayImgList(:,:,1),
 grayImgList(:,:,2), corrX, corrY);
[ H, inliersX, inliersY ] = ransac2(grayImgList(:,:,1),
 grayImgList(:,:,2), roughInliersX, roughInliersY);

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

% Image Warping
warpedImage = warpImage(rgbImgList(:,:,:,1), H);

figure;
imshow(warpedImage/255);

grayWarpedImage = round(rgb2gray(warpedImage/255)*255);

% Find the Corners of the Warped Image Matching with the Second Image
[corrXWarped, corrYWarped] = getCorners(grayWarpedImage,
 grayImgList(:,:,2), rThreshold, nccThreshold);
[ roughHWarped, roughInliersWarpedX, roughInliersWarpedY ]
 = ransac2(grayWarpedImage, grayImgList(:,:,2), corrXWarped,
 corrYWarped);
[ HWarped, inliersXWarped, inliersYWarped ] = ransac2(grayWarpedImage,
 grayImgList(:,:,2), roughInliersWarpedX, roughInliersWarpedY);
XOffset = round(HWarped(1, 3));
YOffset = round(HWarped(2, 3));

% Image Blending
blendedImage = blendImages(warpedImage, rgbImgList(:,:,:,2), XOffset,
 YOffset);
figure
imshow(blendedImage/255);

timeElapsed = toc
```

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 [ H, inliersX, inliersY ] = ransac2( imgList1, imgList2,
 corrX, corrY )
    totalImageSize1 = size(imgList1);
    imageHeight1 = totalImageSize1(1);
    imageWidth1 = totalImageSize1(2);

    j = 0;
    for x = 1:imageHeight1
        for y = 1:imageWidth1
            if ((corrX(x,y) ~= 0) && (corrY(x,y) ~= 0))
                j = j + 1;
                img1(j,:) = [x y];
                img2(j,:) = [corrX(x,y) corrY(x,y)];
            end
        end
    end

    bestH = zeros(3,3);
    bestHNumCorrect = -1;
    bestInliers1 = zeros(1,2);
    bestInliers2 = zeros(1,2);

    for i = 1:(j*10)
        rng('shuffle');
        loc1 = ceil(rand() * j);
        loc2 = ceil(rand() * j);
        loc3 = ceil(rand() * j);
        loc4 = ceil(rand() * j);

        x1 = img1(loc1, 1);
        y1 = img1(loc1, 2);
        xNew1 = img2(loc1, 1);
        yNew1 = img2(loc1, 2);
        x2 = img1(loc2, 1);
        y2 = img1(loc2, 2);
        xNew2 = img2(loc2, 1);
        yNew2 = img2(loc2, 2);
        x3 = img1(loc3, 1);
        y3 = img1(loc3, 2);
        xNew3 = img2(loc3, 1);
        yNew3 = img2(loc3, 2);
        x4 = img1(loc4, 1);
        y4 = img1(loc4, 2);
        xNew4 = img2(loc4, 1);
        yNew4 = img2(loc4, 2);

        potentialH = generateHFromPoints(x1, y1, xNew1, yNew1, x2, y2,
 xNew2, yNew2, x3, y3, xNew3, yNew3, x4, y4, xNew4, yNew4);
        [potentialHNumCorrect, potentialInliers1, potentialInliers2] =
 calculateNumCorrectPoints(img1, img2, potentialH);

        if (potentialHNumCorrect > bestHNumCorrect)
```

```
            bestH = potentialH;
            bestHNumCorrect = potentialHNumCorrect;
            bestInliers1 = potentialInliers1;
            bestInliers2 = potentialInliers2;
        end
    end

    H = bestH;
    inliers1 = bestInliers1;
    inliers2 = bestInliers2;

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

    inliersX = zeros(size(imgList1));
    inliersY = zeros(size(imgList2));

    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 [H] = generateHFromPoints(x1, y1, xNew1, yNew1, x2, y2,
 xNew2, yNew2, x3, y3, xNew3, yNew3, x4, y4, xNew4, yNew4)
    A1 = [x1 y1 1; x2 y2 1; x3 y3 1; x4 y4 1];
    B1 = [xNew1; xNew2; xNew3; xNew4];
    A2 = A1;
    B2 = [yNew1; yNew2; yNew3; yNew4];
    A3 = A2;
    B3 = [1; 1; 1; 1];

    H1 = linsolve(A1,B1);
    H2 = linsolve(A2,B2);
    H3 = linsolve(A3,B3);
    H = [transpose(H1); transpose(H2); transpose(H3)];
end
```

Published with MATLAB® R2017a

```
function [numCorrect, inliers1, inliers2] =
 calculateNumCorrectPoints(points1, points2, H)

totalSizePoints = size(points1);
numPoints = totalSizePoints(1);

numCorrect = 0;
j = 0;

inliers1 = zeros(1,2);
inliers2 = zeros(1,2);

accuracyThreshold = 1;

for i = 1:numPoints
    pointOld = [points1(i,1); points1(i,2); 1];
    pointNew = H * pointOld;
    if (abs(points2(i,1) - pointNew(1)) <= accuracyThreshold &&
 abs(points2(i,2) - pointNew(2)) <= accuracyThreshold)
        j = j + 1;
        inliers1(j,1) = points1(i,1);
        inliers1(j,2) = points1(i,2);
        inliers2(j,1) = points2(i,1);
        inliers2(j,2) = points2(i,2);
        numCorrect = numCorrect + 1;
    end
end
```

Published with MATLAB® R2017a

```
function [ warpedImage ] = warpImage( rgbImage, H )
    totalImageSize = size(rgbImage);
    imageHeight = totalImageSize(1);
    imageWidth = totalImageSize(2);

    dummyVal = -100000;
    minX = dummyVal;
    minY = dummyVal;
    maxX = dummyVal;
    maxY = dummyVal;

    for x = 1:imageHeight
        for y = 1:imageWidth
            newPoints = H * [x; y; 1];
            newX = round(newPoints(1));
            newY = round(newPoints(2));
            if (newX < minX || minX == dummyVal)
                minX = newX;
            end
            if (newY < minY || minY == dummyVal)
                minY = newY;
            end
            if (newX > maxX || maxX == dummyVal)
                maxX = newX;
            end
            if (newY > maxY || maxY == dummyVal)
                maxY = newY;
            end
        end
    end

    if (minX < 0)
       offsetX = minX * -1 + 1;
       offsetY = minY * -1 + 1;
    else
       offsetX = minX;
       offsetY = minY;
    end

    warpedImageSize = [(maxX - minX + 1) (maxY - minY + 1) 3];
    warpedImage = zeros(warpedImageSize);
    warpedImageHeight = warpedImageSize(1);
    warpedImageWidth = warpedImageSize(2);

    for x = 1:warpedImageHeight
        for y = 1:warpedImageWidth
            oldPoints = zeros(2, 1);
            if (minX < 0)
                oldPoints = H \ [x - offsetX; y - offsetY; 1];
            else
                oldPoints = H \ [x + offsetX; y + offsetY; 1];
            end
```

```
            oldX = round(oldPoints(1));
            oldY = round(oldPoints(2));

            if (oldX >= 1 && oldX <= imageHeight && oldY >=1 && oldY
 <= imageWidth)
                warpedImage(x, y, :) = rgbImage(oldX, oldY, :);
            end
        end
    end
end
```

Published with MATLAB® R2017a

```
function [newWarpedImage] = blendImages(image1, image2, XOffset,
 YOffset)
    sizeImage1 = size(image1);
    sizeImage1X = sizeImage1(1);
    sizeImage1Y = sizeImage1(2);
    sizeImage2 = size(image2);
    sizeImage2X = sizeImage2(1);
    sizeImage2Y = sizeImage2(2);

    image1XOffset = 0;
    image1YOffset = 0;
    image2XOffset = 0;
    image2YOffset = 0;

    if (XOffset < 0)
        image2XOffset = abs(XOffset);
    else
        image1XOffset = XOffset;
    end
    if (YOffset < 0)
        image2YOffset = abs(YOffset);
    else
        image1YOffset = YOffset;
    end

    newWarpedImageX = max(sizeImage1X, sizeImage2X) + abs(XOffset);
    newWarpedImageY = max(sizeImage1Y, sizeImage2Y) + abs(YOffset);
    newWarpedImage = ones(newWarpedImageX, newWarpedImageY, 3) * -1;

    if (XOffset < 0 && YOffset < 0)
        for i = 1:sizeImage1X
            for j = 1:sizeImage1Y
                newWarpedImage(i, j, :) = image1(i, j, :);
            end
        end

        for k = 1:sizeImage2X
            for l = 1:sizeImage2Y
                if (newWarpedImage(k + abs(image2XOffset), l +
 abs(image2YOffset), :) == -1)
                    newWarpedImage(k + abs(image2XOffset), l +
 abs(image2YOffset), :) = image2(k,l, :);
                else
                    pixelAverage = (image2(k,l, :) + newWarpedImage(k
 + abs(image2XOffset), l + abs(image2YOffset), :)) / 2;
                    newWarpedImage(k + abs(image2XOffset), l +
 abs(image2YOffset), :) = pixelAverage;
                end
            end
        end
    end
```

```
    if (XOffset > 0 && YOffset > 0)
        for i = 1:sizeImage2X
            for j = 1:sizeImage2Y
                newWarpedImage(i, j, :) = image2(i, j, :);
            end
        end

        for k = 1:sizeImage1X
            for l = 1:sizeImage1Y
                if (newWarpedImage(k + abs(image1XOffset), l +
 abs(image1YOffset), :) == -1)
                    newWarpedImage(k + abs(image1XOffset), l +
 abs(image1YOffset), :) = image1(k,l, :);
                else
                    pixelAverage = (image1(k,l, :) + newWarpedImage(k
 + abs(image1XOffset), l + abs(image1YOffset), :)) / 2;
                    newWarpedImage(k + abs(image1XOffset), l +
 abs(image1YOffset), :) = pixelAverage;
                end
            end
        end
    end

    newWarpedImage(newWarpedImage == -1) = 0;
end
```

Published with MATLAB® R2017a