Table of Contents

GrayScale Images : Panorama Stitching using Harris points computed using Laplacian of Gaussian (LOG) formulation

Written by : Sameer Sharma (ED13D007), Engg. design , IIT Madras

References:

1. Harris-Affine & Hessian Affine: K. Mikolajczyk and C. Schmid, Scale and Affine invariant interest point detectors. In IJC V 60(1):63-86, 2004.
2. Digital Video Processing (http://10.6.4.152/dvp/dvp.html), Computer Sc., Indian Institute of Technology, Madras.
3. VLFEAT SIFT Tool box (http://www.vlfeat.org/overview/sift.html)
4. RANSAC algorithm with example of finding homography : Edward Wiggin. MATLAB Central 2011.
5. A Comparison of Affine Region Detectors, K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, L. Van Gool. IJCV 2005.
6. Scale-Space Theory in Computer Vision, T. Lindeberg. Springer
7. http://en.wikipedia.org/wiki/Corner_detection
The_Harris_Stephens_Plessey_Shi_Tomasi_corner_detection_algorithm
8. Local Invariant Feature Detectors - A Survey, T. Tuytelaars and K. Mikolajczyk. FGV 2008

```
close all;
clc;
```

The information of the image name for e.g. the images are "rio-1.png", "rio-2.png", "rio-3.png" and so on. Total images are the number of images to be used for the panorama image and refrence image is the center most images, which is used as the refrence.

```
base_image = 'rio-';
format_image = '.png';
total_images = 12;
```

```
reference = 6;
for image_no = 1:total_images
```

Part # 1, get the stable Harris points

```
    name_image_1 = [base_image num2str(image_no-1) format_image];
```

Invoking the Harris_Laplace function for each image, output array is N X 3 which has spatial cordinates and points with respective scales

```
    myPoints_final_1 = Harris_Laplace_fn(name_image_1,20);
```

Part # 2, for the selected Harris points with respectve scales get associated SIFT descriptors using *Vl_SIFT toolbox*

```
    myI_1 = single(double(imread(name_image_1)));

        Get Descriptors for the Harris corner points

    fc1 = [myPoints_final_1(:,2),myPoints_final_1(:,1),...
                myPoints_final_1(:,3),zeros(size(myPoints_final_1,1),1)];

    [f1{image_no},d1{image_no}] = vl_sift(myI_1,'frames',...
                                                    fc1','orientations');

end
%  %% Part # 3, Finding point correspondence between a pair of images using the SI
for pairs = 1:total_images - 1

    distance = [];
    sift_match_points_f1 = [];
    sift_match_points_f2 = [];
```

d() has all the descriptors as found out in the last step

```
    d_1 = d1{pairs};
    f_1 = f1{pairs};
    d_2 = d1{pairs+1};
    f_2 = f1{pairs+1};
    name_image_1 = [base_image num2str(pairs-1) format_image];
    name_image_2 = [base_image num2str(pairs) format_image];
    myImages_1 = double(imread(name_image_1));
    myImages_2 = double(imread(name_image_2));
```

Compute the distance between the feature descriptors

```
    for p =1:size(d_1,2)
        for q=1:size(d_2,2)
            distance(p,q) = norm(double(d_1(:,p)) - double(d_2(:,q)));
        end
    end
```

Thresholding the distance function

```
 distance_norm = 100*(distance - min(min(distance)))/...
                            (max(max(distance)) - min(min(distance)));
myNorm = distance_norm < 10;
distance = distance.*(double(myNorm));
[row, col] = find(distance);
```

Ratio test of nearest neighbors

```
for check = 1:numel(row)
    row_sort = sort(distance(row(check),:),'descend');
    ratio = row_sort(1)/row_sort(2);
    if ratio > 2
     sift_match_points_f1 = [sift_match_points_f1 ,...
                                                 f_1(1:2,row(check))];
     sift_match_points_f2 = [sift_match_points_f2 ,...
                                                 f_2(1:2,col(check))];

    end
end
```

Visualization of the matched features

```
figure,showMatchedFeatures(myImages_1,myImages_2,...
                 sift_match_points_f1',...
                     sift_match_points_f2','montage');
```

Part # 4, Compute Homography employing the matched points using RANSAC

Store Homography for each pair

```
  H{pairs} = RANSAC(sift_match_points_f2,sift_match_points_f1);

end
```

Part # 5, final stiching using CANVAS approach

Choosing a reference frame, it depends on the available number of frames

```
name_image_reference = [base_image num2str(reference-1) format_image];
name_image_first = [base_image num2str(0) format_image];
name_image_last = [base_image num2str(2*(reference-1)) format_image];
```

Reading the first and last frame to get the dimensions of the canvas, we will use for stitching

```
im2 = double(imread(name_image_reference));
im1 = double(imread(name_image_first));
im3 = double(imread(name_image_last));
```

Define a homographic tranform using maketform structure (inbuilt Matlab) First half and Last half is defined as follows: if you have 5 frames say f1, f2, f3, f4, f5, suppose we chose f3 as our reference, then f1,f2 belongs to first half and f4, f5 to last half

```
H_first_half{1} = H{reference - 1};
H_last_half{1}  = H{reference};
```

For the first half we have to do an inverse of the Homography,because we have defined homographies as , f1 = h1*f2 , f2 = h2*f3, f3 = h3*f4, f4 = h4*f5

```
T_first{1} = maketform('projective',(inv(H_first_half{1}))');
T_last{1} = maketform('projective', H_last_half{1}');
```

Computing the transforms for all the pairs at once

```
for image_index = 2:reference-1
    H_first_half{image_index} = H{reference - image_index}*...
                                H_first_half{image_index-1};
    T_first{image_index}    = maketform('projective',...
                                    (inv(H_first_half{image_index}))');
    H_last_half{image_index}  = H_last_half{image_index-1}*...
                                  H{image_index + reference - 1};
     T_last{image_index}    = maketform('projective',...
                                    H_last_half{image_index}');
end
```

Computing the bounds or size of the Canvas

```
T12 = maketform('projective', (inv(H_first_half{reference - 1}))');
T32 = maketform('projective', H_last_half{reference - 1}');
[im1t,xdataim2t,ydataim2t]=imtransform(im1,T12);
[im3t,xdataim2t_new,ydataim2t_new]=imtransform(im3,T32);

xdataout=[min([1,xdataim2t(1),xdataim2t_new(1)]) max([size(im2,2),...
                                        xdataim2t(2),xdataim2t_new(2)])];
ydataout=[min([1,ydataim2t(1), ydataim2t_new(1)]) max([size(im2,1),...
                                        ydataim2t(2), ydataim2t_new(2)])];
```

Part # 6, Panorama visualization

We tranform all the images with the computed xdata and ydata

Reference frame (transformed with I)

```
im2t_ref = imtransform(im2,maketform('affine',eye(3)),'XData',xdataout,...
                                          'YData',ydataout);
Canvas_previous = zeros(size(im2t_ref));
for k = 1: reference-1
    Panaroma_image_first = double(imread([base_image ...
                                         num2str(k-1) format_image]));
    Panaroma_image_last =  double(imread([base_image ...
                                   num2str(reference+k-1) format_image]));
    imtransformed_first  =   imtransform(Panaroma_image_first,...
                         T_first{reference - k}, 'XData', xdataout,...
                         'YData',ydataout );
     imtransformed_last  =   imtransform(Panaroma_image_last,...
                         T_last{k}, 'XData', xdataout,...
                         'YData',ydataout);
%       [ro, col] = size(imtransformed_first);
```

```
    Canvas = max(imtransformed_first , imtransformed_last);
    Canvas_previous = max(Canvas,Canvas_previous);
end

im_panorama = max(im2t_ref, Canvas_previous);
B = medfilt2(im_panorama);
```

Visualization

```
figure,imshow(uint8(B));
```

Published with MATLAB® R2014a