Extended Homework
Panoramic Images: REPORT
A.A. 2015-2016

Roberto Costa

September 16, 2016

Subject:	Image and Video Analysis
Professor:	P. Zanuttigh
Student number:	1128285

1 Performing image stiching to create a panoramic image

The purpose of the lab is to develop an algorithm to perform the stiching of some images, based on SIFT features extraction, to create a panoramic image.
The first step is to take some images of a landscape, in particular 3 rows of 8 images have been taken: the first row is the bottom part of the landscape and the last row is the upper one.
Once the images are taken (trying to keep the same angular distance between any couple of neighbour images), the images have been projected on a spherical surface with the following code.

```
function [ out ] = sphProj( in )
global glob
%sphProj: Perform spherical projection of color images
%   output is a cell array with the projected images,
%   input is a ImageSet object
    out = cell(in.Count,1);
    imInOriginal = read(in,1);
    defineConstants(in.Count,imInOriginal);
    cameraParams = glob.cameraParams;
    %[imIn,~] = undistortImage(imInOriginal,cameraParams);
    imIn = imInOriginal;
    imOut = zeros(glob.outY,glob.outX,3,'uint8');
    %position = zeros(size(imOut,1),size(imOut,2),2);
    for j=1:size(imOut,1)
        for k=1:size(imOut,2)
            x = ...
                glob.planeDistance*tan(k/glob.rX-glob.alpha/180*pi)+glob.imX/2;
            y = ...
                glob.planeDistance*tan(j/glob.rY-glob.beta/180*pi)+glob.imY/2;
```

```
18              %disp(strcat('x=',num2str(x),' ; y=',num2str(y),...
19              %    ' ; x1=',int2str(k),' ; y1=',int2str(j)));
20              imOut(j,k,:)=imIn(round(y),round(x),:);
21              %position(j,k,1)=2*glob.alpha*k/size(imOut,2)-glob.alpha;    ...
                    % x
22              %vposition(j,k,2)=2*glob.beta*j/size(imOut,1)-glob.beta;      ...
                    % y
23          end
24      end
25      out{1} = imOut;
26      %out{1} = imIn;
27
28      for i=2:in.Count
29          imInOriginal = read(in,i);
30          %imIn = undistortImage(imInOriginal,cameraParams);
31          imIn = imInOriginal;
32          imOut = zeros(glob.outY,glob.outX,3,'uint8');
33          %position = zeros(size(imOut,1),size(imOut,2),2);
34          for j=1:size(imOut,1)
35              for k=1:size(imOut,2)
36                  x = ...
                        glob.planeDistance*tan(k/glob.rX-glob.alpha/180*pi)+glob.imX/2;
37                  y = ...
                        glob.planeDistance*tan(j/glob.rY-glob.beta/180*pi)+glob.imY/2;
38                  %disp(strcat('x=',num2str(x),' ; y=',num2str(y),...
39                  %    ' ; x1=',int2str(k),' ; y1=',int2str(j)));
40                  imOut(j,k,:)=imIn(round(y),round(x),:);
41                  %position(j,k,1)=2*glob.alpha*k/size(imOut,2)-glob.alpha; ...
                          % x
42                  %vposition(j,k,2)=2*glob.beta*j/size(imOut,1)-glob.beta; ...
                            % y
43              end
44          end
45          out{i} = imOut;
46          %out{i} = imIn;
47      end
48
49  end
50
51  function [ out ] = defineConstants( n,im )
52  %defineConstants( n,im ) set global glob vaiable
53  %   input n: # of images, input im: sample image
54  global glob
55  calibration;
56  glob.thrMatch = 0.55;
57  glob.thrAngle = 0.1/(2*pi);
58  glob.thrScale = 1;
59  glob.roundSize.x = 1200;
60  glob.roundSize.y = 1400;
61  glob.roundVerticalSize = 1400;
62  glob.delTr = 10;
63  glob.horizontalMultiplier = 1/10;
64  glob.nIm = n;
65  % sizes of the images
66  glob.imX = size(im,2);
67  glob.imY = size(im,1);
```

```
68
69 glob.focalDistance = 18;
70 glob.CCDSizeX = 22.3;
71 glob.CCDSizeY = 14.9;
72 glob.alpha = atan(glob.CCDSizeX/2/glob.focalDistance)*180/pi;
73 glob.beta = atan(glob.CCDSizeY/2/glob.focalDistance)*180/pi;
74
75 glob.planeDistance = (glob.imX / 2 / tan(glob.alpha/180*pi)+...
76     glob.imY / 2 / tan(glob.beta/180*pi)) /2;
77 glob.rSfera = sqrt(glob.planeDistance^2+glob.imX^2/4+glob.imY^2/4);
78 glob.rX = sqrt(glob.planeDistance^2+glob.imX^2/4);
79 glob.rY = sqrt(glob.planeDistance^2+glob.imY^2/4);
80 glob.outX = floor(glob.rX*2*glob.alpha/180*pi);
81 glob.outY = floor(glob.rY*2*glob.beta/180*pi);
82 glob.minCorrespondance = 4;
83 end
```

It has been possible to correct the fish-eye effect with a matlab tool (Camera calibration), by taking from 10 to 20 photos of a chess table, the following code has been used.

```
1  % Auto-generated by cameraCalibrator app on 12-Sep-2016
2  %--------------------------------------------------------
3
4  global glob
5  % Define images to process
6  imageFileNames = {'D:\unipd\Esami\Image and video ...
       analysis\final_project\images\calibration\IMG_8629.JPG',...
7      'D:\unipd\Esami\Image and video ...
           analysis\final_project\images\calibration\IMG_8630.JPG',...
8      'D:\unipd\Esami\Image and video ...
           analysis\final_project\images\calibration\IMG_8631.JPG',...
9      'D:\unipd\Esami\Image and video ...
           analysis\final_project\images\calibration\IMG_8633.JPG',...
10     'D:\unipd\Esami\Image and video ...
           analysis\final_project\images\calibration\IMG_8634.JPG',...
11     'D:\unipd\Esami\Image and video ...
           analysis\final_project\images\calibration\IMG_8635.JPG',...
12     'D:\unipd\Esami\Image and video ...
           analysis\final_project\images\calibration\IMG_8636.JPG',...
13     'D:\unipd\Esami\Image and video ...
           analysis\final_project\images\calibration\IMG_8638.JPG',...
14     'D:\unipd\Esami\Image and video ...
           analysis\final_project\images\calibration\IMG_8642.JPG',...
15     };
16
17 % Detect checkerboards in images
18 [imagePoints, boardSize, imagesUsed] = ...
       detectCheckerboardPoints(imageFileNames);
19 imageFileNames = imageFileNames(imagesUsed);
20
21 % Generate world coordinates of the corners of the squares
22 squareSize = 32;  % in units of 'mm'
23 worldPoints = generateCheckerboardPoints(boardSize, squareSize);
24
25 % Calibrate the camera
```

```
[cameraParams, imagesUsed, estimationErrors] = ...
    estimateCameraParameters(imagePoints, worldPoints, ...
    'EstimateSkew', false, 'EstimateTangentialDistortion', false, ...
    'NumRadialDistortionCoefficients', 3, 'WorldUnits', 'mm', ...
    'InitialIntrinsicMatrix', [], 'InitialRadialDistortion', []);
glob.cameraParams = cameraParams;
%{
% View reprojection errors
h1=figure; showReprojectionErrors(cameraParams, 'BarGraph');

% Visualize pattern locations
h2=figure; showExtrinsics(cameraParams, 'CameraCentric');

% Display parameter estimation errors
displayErrors(estimationErrors, cameraParams);

% For example, you can use the calibration data to remove effects of lens ...
    distortion.
originalImage = imread(imageFileNames{1});
undistortedImage = undistortImage(originalImage, cameraParams);

% See additional examples of how to use the calibration data.  At the ...
    prompt type:
% showdemo('MeasuringPlanarObjectsExample')
% showdemo('StructureFromMotionExample')
%}
```

Some optimizations have been adopted, like computing the SIFT features not to the whole image, but only to the right part of the image for the left image and to the left part of the image for the right image.
The stitching algorithm takes the first image, in the middle height; then takes the bottom image and the higher images and it joins them to the first, kept as a reference. Matlab references provide VLFeat Library, which can be used to compute the mosaic of two images (http://www.vlfeat.org/applications/sift-mosaic-code.html).
Once all the columns are computed, they can be joint horizontally.
The panoramic image produced follows:

Figure 1: Output image