Image Processing - 67829

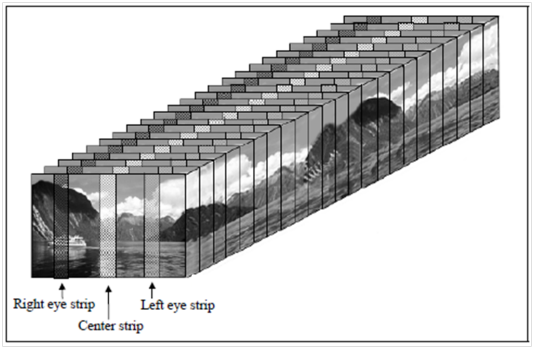
Exercise 4: Stereo Panoramic Video

Due date: 14/1/2015

1 Overview

The purpose of this exercise is to create panoramic images and stereo movies using basic image stitching. The input of your program will be a sequence of images captured by a moving camera, and the output - a mosaic image of the sequence.

Stereo movie is created by producing mosaic images and combining them into a single video. In a regular panorama mosaic each image contributes to the mosaic a strip taken from its center. In stereo mosaicing each mosaic image is produced by taking the strip from the left part of the original image or from the right part of it. As shown in Fig. 1, three panoramas are constructed simultaneously. One panorama is constructed from strips located at the right side of the images, one is constructed from strips located at the left side of the images and a third panorama is constructed from the strips located in the center.



By combining these panoramas into a single movie we create a 3D effect for the viewer. Example can be seen at http://www.vision.huji.ac.il/stereo/.

Given the set of images, your implementation will include the following steps:

- Registration: The motion between each consecutive image pair is computed.
- Stitching: Take a strip from image to create a panorama. This is done several times to create different panoramas from different strip locations.

2 Loading images

You will need to implement a function for loading images from a given directory. The input to the function is the directory of the images and the output is a 4 dimensional vector with all the images in the directory in RGB format. You can assume that the images are already in the required order in the directory and are of the same size. You can also assume that there are only two types of objects in the directory - files that represent images and files that represent the current and upper directory ('.','..'). The function interface should be as follows:

```
function imgs=loadImages(directoryPath)
%
% Read all images from directoryPath
%
% Arguments:
% directoryPath − A string with the directory path
%
% Returns
% imgs − 4 dimensional vector, where imgs(:,:,:,k) is the k−th
% image in RGB format.
%
```

3 Registration

In this section we will describe how to produce a set of transformations $T_{1,2}, \cdots, T_{k,k+1}$, such that each convert the coordinates from next frame to the current one. We assume that there exist only translation between the images. Thus, each transformation is of the form:

$$T_{i,i+1} := \begin{pmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{pmatrix}$$

where dx, dy are the translations between the two images.

In order to do it, for each two consecutive frames you will need to implement a method for registering the most fitted transformations. You can choose to implement one of the following options:

- Registration with Lucas-Kanade. The registration will be done based on Lucas-Kanade with pyramids.
- Registration with descriptors. The registration will be done based on descriptors and RANSAC.

You will find details regarding each of the above method in attached files.

4 Panorama Stitching

This section will guide you into building a set of panoramic images, each corresponds into a different view. By creating a movie of these images we will have the required stereo movie.

Our efforts in the previous section eventually provided us with the set of registered transformations $T_{i,i+1}$, for $i = 1..M-1$, between consecutive frames in a given image sequence of M frames I_i. We would now like to use these transformations to stitch these M frames into one combined panorama frame.

4.1 Transforming to a common coordinate system

The first stage in doing this is to pick a coordinate system in which we would like the panorama to be rendered. We will (somewhat arbitrarily) choose this coordinate system to be the coordinate system of the first frame I_1 in our image sequence. What we mean by this is that the resulting panorama image will be composed of frame I_1 with all the other frames backwarped so that they properly align with it. To achieve this we will need to translate the set of transformations $T_{i,i+1}$ that transform image coordinates in frame I_{i+1} to frame I_i into a different set of transformations $\bar{T}_{1,i}$ that transform image coordinates in frame I_i to this frame I_1.

First we note that given 2 transformation matrices $T_{a,b}$ and $T_{b,c}$ and a point p_a in coordinate system a, we can transform the point to coordinate system c by first operating with $T_{a,b}$ to obtain an intermediate point p_b in system b and then by operating with $T_{b,c}$ to obtain p_c in system c. Each time we operate with a T on a point $p = (x, y)$ we do two things: First we multiply the homogeneous column 3-vector $\tilde{p} = (x, y, 1)^t$ from the left by T and then we renormalize to keep the 3rd element equal to 1. When we operated on p_a first with $T_{a,b}$ and then with $T_{b,c}$ we could have equally well multiplied the homogeneous vector $\tilde{p}_a$ once from the left by the matrix $T_{b,c}T_{a,b}$ and then normalized only once at the very end. We see then that multiplying the transformation matrices $T_{a,b}$ and $T_{b,c}$ has produced a new transformation matrix $T_{a,c} \equiv T_{b,c}T_{a,b}$ that now transforms from coordinate system a directly to c.

We may now use this property to obtain $\bar{T}_{0,m}$ from $T_{i,i+1}$, where $\bar{T}_{0,m}$ transform the m image to the panorama coordinate system. We do this by

- For the first image we set $\bar{T_{0,1}}$ to the 3×3 identity matrix $I =$ `eye(3)`
- For the rest of the images we set $\bar{T}_{0,m} = T_{0,1} * ... * T_{m-1,m}$

This procedure should be implemented in the following function

```
function Tout=imgToPanoramaCoordinates(Tin)
% Tout{k} transforms image i to the coordinates system of the Panorama Image.
%
% Arguments:
% Tin − A set of transformations (cell array) such that T_i transfroms
% image i+1 to image i.
%
% Returns:
% Tout − a set of transformations (cell array) such that T_i transforms
% image i to the panorama corrdinate system which is the the corrdinates
% system of the first image.
```

In this function interface the cell array of `Tin` corresponds in the above discussion to the set of transformations $T_{i,i+1}$ and the returned `Tout` corresponds to the set $\bar{T}_{0,m}$.

4.2 Rendering a single panoramic view

Given the set of T transformations $\bar{T}_{0,i}$ transforming pixel coordinates in frame I_i to the panorama coordinate system, we can proceed to describe the way in which each panorama frame is rendered. Each frame is build from strips taken from all the set of images, where the center of the strip is at the same fixed location over all images. For now, assume that we know the required centers of the strips, their width and the size of the panoramic frame. We now describe how to render each frame based on these parameters and the given transformations.

The first step is to determine the optimal boundaries between each two consecutive strips in the panorama frame. We do this by converting the centers of the two consecutive strips into the panorama coordinates and take the middle point between them as the right and left boundaries of the strips. Consider x_i^{center} as the center in panorma coordinates of the strip taken from image i. The optimal boundaries between strips i and $i+1$ is $(x_i^{center} + x_{i+1}^{center})/2$.

Now that we have all the optimal boundaries we can build the panorama frame by iterating over the images, extract the strip and stitch it. You should use backwarping. This can be done by evaluating for each strip in the i-th image the left upper point and right lower point in image coordinates out of the optimal strip location evaluated before. Then, use meshgrid and interp2 to warp the image slice into the panorama frame.

You should implement the rendering of each panoramic view by the following interface:

```
function [panoramaFrame,frameNotOK]=renderPanoramicFrame(panoSize,imgs,T,imgSliceCenterX,halfSliceWidthX)
%
% The function render a panoramic frame. It does the following:
% 1. Convert centers into panorama coordinates and find optimal width of
% each strip.
% 2. Backwarpping each strip from image to the panorama frames
%
% Arguments:
% panoSize - [yWidth,xWidth] of the panorama frame
% imgs - The set of M images
% T - The set of transformations (cell array) from each image to
% the panorama coordinates
% imgSliceCenterX - A vector of 1xM with the required center of the strip
% in each of the images. This is given in image coordinates.
% sliceWidth - The suggested width of each strip in the image
%
% Returns:
% panoramaFrame- the rendered frame
% frameNotOK - in case of errors in rednering the frame, it is true.
```

4.3 Creating the stereo video

We now describe how to create the video. It is done by using the previous methods to extract transforms, convert them into a common coordinate system, calculate the size of each panoramic frame, render each view and putting them together into a single movie.

At this stage, the only missing step is how to evaluate the size of the panorama frame and the size of each strip. It is calculated based on a parameter passed to the main function - nViews. nViews represents the number of views to extract from each image. Given nViews, we divide the image into equal nViews strips and each one is placed in a different panoramic frame. Based on the number of images and the width of each strip we can decide the size of the panoramic frames. The main function interface should be as follows:

```
function [stereoVid] = createStereoVideo(imgDirectory, nViews)
%
% This function gets an image directory and create a stereo movie with
% nViews. It does the following
%
% 1. Match transform between pairs of images.
```

```
% 2. Convert the transfromations to a common coordinate system.
% 3. Determine the size of each panoramic frame.
% 4. Render each view.
% 5. Create a movie from all the views.
%
% Arguments:
% imgDirectory - A string with the path to the directory of the images
% nView - The number of views to extract from each image
%
% Returns:
% stereoVid - a movie which includes all the panoramic views
%
```

5 Bonus Questions

- Implement rotational+translation panorama: (4 points).
- Blending and improved stitching: change the way the different strips are combined in the `renderPanoramicFrame` function by performing pyramid-blending between adjacent strips (1 point) .

Satisfactorily solving each of these questions will entitle you to extra points in the final grade (so that's a maximum of 5 final grade points in total).

6 Guidelines

- Debugging: Debug every part of your code independently. You can use matlab's estimateGeometricTransform & estimateGeometricTransform to evaluate the performance of your descriptors. You can also take a single strip from the middle in order to test the standard panoramic rendering.
- Your submission file should also be packed in an archive called ex4.zip.