Introduction to Digital Image Processing

Exercise No. 4. - Building Panoramic Images

Due: 2-02-2006

Purpose

The purpose of this exercise is to create panoramic images in both cases of static and dynamic scenes and to use basic stitching and stitching based on min-cut optimization. You will get as an input a sequence of images captured by a moving camera, and implement the following steps:

- Register Images: Compute the motion (pure translation) between each pair of consecutive images, using the pyramid Lucas-Kanade method.

- Warp the color images to leave only the integer horizontal component of the translation.

- Take a strip from each of the warped images to create the mosaic image. You should implement two ways for this task: a basic stitching mechanism or stitching based on min-cut method.

The program you submit should read a set of images and return a mosaic and a matrix with the motion parameters between all the images (will be referred as motion vector).

\[
[mosaic, motions] = ex4(baseImageName, firstFileIndex, lastFileIndex, jump, stitchMethod, overlapStripWidth, displaySeamsOnPanorama, displayStitchProcess)
\]

where:

- **baseImageName** is the pattern of the image name used by sprintf. This pattern should include one integer argument (for example 'im\%02d.jpg')

- **firstFileIndex**, **lastFileIndex** are the indices of the first and last files of images to read. For example, to read a set of images im05.jpg..im50.jpg, 'baseImageName' should be 'im\%02d.jpg' and firstFileIndex, lastFileIndex should be 5,50.

- **jump** is a parameter describing the jump in indexes between 2 consecutive frames that are read. For example: if first and last images are 5 and 10 accordingly and jump is 2 than the indexes of the images that should be read are: 5,7, and 9.

- **stitchMethod** is a string either 'basic' or 'mincut' determining the stitching method to be used.

- **overlapStripWidth** is a parameter of the mincut stitching methods determining the width of the overlap area to be used when looking for the min-cut (should be treated only if the stitch method is 'mincut', see details in the stitching section).

- **displaySeamsOnPanorama** is logical parameter (0 or 1) determining whether to display the seams of the min-cut process over the panoramic image (should be treated only if the stitch method is 'mincut').

- **displayStitchProcess** is logical parameter (0 or 1) determining whether to display the stitching process while it is done (1) or not (0). If the flag is 1 the panorama should be displayed (on the same figure) after every image is added to it.
• **mosaic** - is the resulting panoramic image (with or without the seams on it as determined in the display method)

• **motions** - is a matrix containing the relative motions between each two sequential frames of the sequence as computed in the Lucas-Kanade stage. The size of this matrix is Nx2 where N is the number of images that were read. The first row of this motion vector should be [0 0].

The first stage of the process is aligning the images to each other using Lucas-Kanade Image Alignment with Pyramids. A more detailed description of the entire process as taught in class appears in www.cs.huji.ac.il/~impr/www/Ex4LK.pdf. We will assume for the description of the stitching that the camera is moving generally from left to right. You may use this assumption also in your program. As a convention if the camera is moving from left to right use negative values in your tx components of the motion vector.

1 Stitching Images

You will implement two methods of stitching: basic stitching and stitching based on min-cut.

In both stitching methods, if the function ex4 was called with the parameter `displayStitchProcess` set to 1, you should display a single figure that shows the panorama after each image being stitched to it (use 'figure(figNum)' and 'hold on' in Matlab to avoid blinking figure while displaying the process).

The first stage in mosaicing is motion computation (Translation). Once you did this you can start stitching.

As a pre process to the stitching stage you should implement a limited stabilization of the sequence: warp the input images to cancel the vertical and sub pixel horizontal components of the motion (leave only the integer part of the horizontal translation). This will enable you an easier stitching procedure later.

1.1 stabilization

In this stage you are stabilizing the input sequence to leave only the horizontal integer part of the motion. for example: If the motion between two successive images was tx = -5.3 and ty = 1.3, you should warp the second image towards the first image such that the motion between them after the warping will be tx = -5 and ty =0. Since the motion between two images is non-integer, back warping should be used: for a pixel in the warped image, assign a non-integer location in the second image and use bilinear interpolation (you can use interp2 in your warping function). You may pad the warped images to simplify the warping process but note to perform padding that will not fail for different sequences as the motions between frames changes (choose a same padding to all the images and set it according to the maximum accumulative horizontal offset in the sequence). Note that when you are doing the stabilization you should also keep a motion vector that reflect the residual motion between the images. The main function (ex4) should return the original motion vector and not the motion vector after the update.

1.2 Basic Stitching

To create a mosaic from the images, we collect strips from the images and paste them side by side in the panorama. Since the images are stabilized up to horizontal translation The basic stitching include simple copy-and paste procedure and it is described below.

The strips are collected around the horizontal center of the image, lets call this center c.
We choose the coordinate system of the first image $I_1$ as the coordinate system of the mosaic image. Thus, given the offset (the horizontal translation component) \( \{d_j\}_{j=1}^{n} \) between the images \( \{I_j\}_{j=1}^{n} \) we know the offsets between the mosaic and any of the images, so we know in which offset to put the strips. The question left to solve is what are the boundaries of the strip in the images.

For image $I_j$ we decide on the boundaries of the strip as follows: Let $d_{j-1}$ be the offset between images $I_{j-1}, I_j$ and similarly let $d_j$ be the offset between images $I_j, I_{j+1}$. Then the strip boundaries in image $I_j$ should be $c - \frac{d_{j-1}}{2}, c + \frac{d_j}{2}$. If you get fraction values decide how to deal with it in a consistent way. Then this strip is copied to the mosaic according to the offset between the coordinate system of the mosaic and the coordinate system of image $I_j$. This offset is simply: \( \sum_{k=1}^{j-1} d_k \).

Note that given the motions vector you compute the size of the mosaic before creating it.

### 1.3 Stitching based on min-cut computation

The stitching based on min-cut is aimed to handle cases where there were moving objects in the sequence.

The idea of the min-cut stitching is to stitch each image to the panorama in a place where the seam between the existing pixels and the newly placed pixels will be as little noticeable as possible. This means that the min-cut algorithm will try to compute a seam with much visual smoothness as possible and avoid placing the seam in pixels of moving objects or other changes in the scene.

The min cut algorithm works by reformulating the stitching problem as a problem of finding minimum cost cut on a graph. A graph is defined with two special terminal nodes (the source and the sink), and a minimum cut between those nodes, is computed.

The graph grid is defined over a predefined overlap aligned area (see details below) of the pixels from the new image that you are currently stitching \( (B) \) and pixels that already exist in the panorama \( (A) \). Each pair of pixels (one from A and one from B) in the overlap area define a node in the graph. Edges are defined between each node and it’s 4 neighbors: 2 vertical neighbors and 2 horizontal neighbors (nodes in the boundary of the overlap area will have less neighbors). The cut in the graph will run between nodes, on the defined edges and will set a stitching scheme: all pixels that are in the left side of the cut will be taken from the panoramic image \( (A) \) and all pixels in the right side of the cut will be take from the stitched image \( (B) \).

The weights on the edges are defined as follows: let \( s \) and \( t \) be two adjacent nodes in the overlap region. Also, let \( A(s) \) and \( B(s) \) be the colors of the pixels at the position \( s \) in the panorama and in the new image, respectively. We define the weight \( W \) on the edge between the two adjacent nodes \( s \) and \( t \) to be:

\[
W = \|A(s) - B(s)\|_2 + \|A(t) - B(t)\|_2 \text{ where } \|\cdot\|_2 \text{ is sum of the square differences over all color channels of the pixel.}
\]

In addition to the weights on the edges there are 2 terminal nodes: one node represents the panoramic image \( (A) \) and the other the new image \( (B) \). The edges connecting nodes to the terminal nodes are special edges with infinitely high cost that define constraint arcs: if a node is attached to one of the terminals, it will be connected to it in the min-cut solution as well. In the case of the panoramic image where we assume the camera moves from left to right, the left column of the overlap area should be attached to the terminate node \( A \) (source - panorama) and the right column of the overlap area should be attached to the terminal node \( B \) (sink - new image). See illustration of this scheme in Figure 1.

In order to perform the min-cut optimization we will use a code based on an article: An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Computer Vision, Yuri Boykov and Vladimir Kolmogorov (code is available at http://www.cs.cornell.edu/People/vnk/software.html). This code was wrapped so you can use it directly from Matlab environment (thanks to the help of Michael Mumcuoglu). To com-
pile the code for Matlab or download the compiled code for your platform you should go to www.cs.hjui.ac.il/~impr/www/Ex4MinCutCode and download the relevant files from there.

The first image is added to the panorama entirely, and the current accumulative offset is initialized to be zero. Every time that a new image is added to the panorama do the following steps:

- Check the current accumulative offset between the last image added to the panorama and the new image to be added. If the current accumulative offset is larger than overlapStripWidth/2 add this image, else go to the next image. This stage is for avoiding adding images that are too close to the last added image as it will not produce good results in the min-cut stitching method.

- Define an overlap area between the panorama and the new image to be used by the min-cut algorithm. The overlap area is located around the offset between the mosaic and the new image to be stitched (offset is calculated just like in the basic stitching process) and its width is based on the parameter overlapStripWidth passed to ex4. For example if overlapStripWidth is 50 pixels, in the panorama the overlap area will start 50 pixels before the offset between the mosaic and the new image, and will end 50 pixels after this offset. In the image the overlap will start 50 pixels before the middle of the image and will end 50 pixels after the middle of the image.

- Define horizontalEdges, verticalEdges, sourceNodes, and sinkNodes on the overlap area. The horizontal and vertical weights should be defined using W equation given above. The sourceNodes should be the indexes of the pixels in the first column and the sinkNodes should be the indexes of the pixels in the last column (no need to assign infinity weights on the terminal nodes, it is done by the wrapped code).

- Find the minimal cut (call the function findMinCut with appropriate parameters)

- Build the stitch of the overlap area (call the function Stitch with appropriate parameters) and paste it to the panorama.

- Paste to the panorama the rest of the new image (from the end of the overlap area to the end of the image).

- Set the current accumulative offset to be zero.

You may add padding to the width of the panorama while stitching, but you should cut this padding in the end of the process (vertical padding does not have to be cut, only horizontal padding).

In order to use the graph cut code you should use the following functions:

- \([\text{segmentation}] = \text{findMinCut}(\text{horizontalEdges}, \text{verticalEdges}, \text{sourceNodes}, \text{sinkNodes})\)

where:

- **horizontalEdges** is a matrix of double weights \([\text{size: } M \times (N - 1) \text{ (M-rows, (N-1)-columns)}]\) containing symmetric weight of each node to it's right neighbor. Note that this matrix is missing the rightmost column

- **verticalEdges** is a matrix of double weights \([\text{size: } (M-1) \times N \text{ ((M-1)-rows, N-columns)}]\) containing symmetric weight of each node to it's bottom neighbor. Note that this matrix is missing the lowest row
Figure 1. (Left) A schematic diagram of the overlapping region between the panorama and the new image to be stitched. (Right) Graph formulation of the seam finding problem with the red line showing the minimum cost cut.

- **sourceNodes** is a matrix of integer (x,y) coordinates of nodes [size L x 2] connected to the source terminal (L is the number of nodes connected to the source)

- **sinkNodes** is a matrix of integer (x,y) coordinates of nodes [size K x 2] connected to the sink terminal (K- the number of nodes connected to the sink)

- **segmentation** is the output of this function: a boolean matrix [size: M x N (M-rows, N-columns)] containing the values 0 or 1 representing whether the pixels is connected to the source (0) or the sink (1)

- [res] = stitch(im1, im2, segmentation, displaySeamOnImg)

where:

- **im1** M x N RGB image, the panoramic image in the overlap area

- **im2** M x N RGB image, the new image in the overlap area

- **segmentation** is M x N boolean matrix - the result of the segmentation process of finding the min cut over a specific graph

- **displaySeamOnImg** boolean determine whether to highlight the seam location in the resulting image (determined by the parameter displaySeamsOnPanorama set by the user when calling c4)

2 Bonus

A bonus of 15 points will be given to the students who will have the fastest implementation of the Lucas-Kanade algorithm. The results should still have a sub-pixel accuracy. If you want this issue to be checked in your code Describe in your README file what you did to speed-up the algorithm.

A bonus of 5 points in the final course grade will be given to those who implement Rotation+Translation alignment using Lucas-Kanade with pyramid, as shown in class. In this case, please mention it in your README file with usage instructions.
3 Notes:

- You can find an example of input sequences for testing on www.cs.huji.ac.il/~impr/www/Ex4ExamplesAndTests You are encourage to shoot your own sequences and share them with the other students (send me and email with details and I can put it on the course’s web site). If you shot your own sequences try to shoot pictures from the same position (turn camera, but don’t translate it)

- Check every part of the exercise (motion computation, warping, min-cut) separately on simple images to make sure it works before you move to the next stage.

- Write your code without unnecessary loops, Matlab stile

- If you implemented one of the bonus parts you should write it in your README and submit it as a hard copy of it to the course’s box.