CSC 2530 Assignment: Panoramic Mosaic

Team Member:

- Nigel Morris
- Paul Yang

1. Summary:

This is a report on panoramic mosaic for a CSC2530 assignment. It uses a translation-only image registration technique proposed by Lucas and Kanade [2]. A coarse-to-fine method is used to produce better estimates. Images are put through the following pipeline. They are first warped into cylindrical coordinates, and then, smoothed with a 5 by 5 Gaussian filter with a standard deviation of 1.0. For each pair of neighboring images, they are registered under a coarse-to-fine hierarchy using the Gaussian pyramid [1] to produce better estimate. Once every pair is matched, a panoramic mosaic is generated based on the estimated translation gathered during the image registration process. They are then mapped to a cylinder as texture maps.

2. Image Registration:

Lucas and Kanade's translation-only technique [2] is used for image registration. A coarse-to-fine technique is also used to provide better estimate. The following sections will discuss the translation-only technique, image smoothing, and coarse-to-fine technique.

2.1 Translation-only Technique

The translation-only technique is described up to 4.5 in Lucas and Kanade's paper [2]. The two-dimensional case is used. The 2D version of equation (9) in the paper is

$$h = \left[\sum_x \left(\frac{\partial F}{\partial x} \right)^T \left(\frac{\partial F}{\partial x} \right) \right]^{-1} \left[\sum_x \left(\frac{\partial F}{\partial x} \right)^T \left(G(x) - F(x) \right) \right]$$

where $\frac{\partial F}{\partial x}$ is the gradient of F(x)

Implementation

- F(x) is interpolated using inverse matching. Given a translation, the intensity of the pixel will be interpolated using its four nearest neighbors.
- There matrix $\left[\sum_x \left(\frac{\partial F}{\partial x}\right)^T \left(\frac{\partial F}{\partial x}\right)\right]$ is symmetric.

  Let $\left[\sum_x \left(\frac{\partial F}{\partial x}\right)^T \left(\frac{\partial F}{\partial x}\right)\right] = \begin{bmatrix} a & b \\ b & c \end{bmatrix}$

  It can be shown that its inverse is $\begin{bmatrix} \frac{c}{ac-b^2} & \frac{-b}{ac-b^2} \\ \frac{-b}{ac-b^2} & \frac{a}{ac-b^2} \end{bmatrix}$

Thus, if we sum up all the products of partial derivatives and the product between partial derivatives and difference in pixel intensity, we can solve for the h easily.

Discussion

The technique works well with very simple test data such as matching translated solid rectangles and circles with a solid background. However, it fails when the image becomes complex. The gradient of the images becomes out of control when the data becomes noisy and in each iteration, the estimate just shoots from one direction to another. The next section will discuss smoothing that deals with this problem.

2.2 Image Smoothing

To smooth the images and get a better gradient, a 5 by 5 Gaussian filter with standard deviation of 1.0 [5] is applied to the image.

Implementation

A 1D filter is used to convolve the image along both the rows and columns [5]. The resulting images look blurred. The gradient of the smoothed images looks more under control.

Discussion

The smoothed image does provide a more controlled gradient. However, the program often does not converge within 128 iterations. However, when given a good initial guess, the program does stay around the good estimate. The following section will discuss how to come up with a good initial guess without user input using a coarse-to-fine technique.

2.3 Coarse-to-fine Technique

Implementation

The coarse-to-fine technique uses a hierarchical representation of image called Gaussian Pyramid [1]. It provides a way to down sample the image. Each pixel in the new down-sampled image is a weighted sum of its neighbors under a 5 by 5 mask. The images gathered from the camera is first smoothed with a Gaussian filter and then down-sampled. The height and width of the resulting image is about half of the original image. The resulting image is then down sampled to the next level. In the program, four to six levels are used.

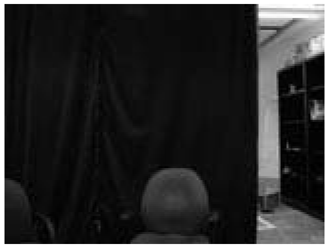
To register a pair of images, the lowest resolution will be used first. The initial guess is 0. The results will be used as initial guess for the higher resolution versions.

Discussion

The technique works for more complex images. No initial guess is necessary. However, the technique requires many iterations to converge. One way to handle the problem is to weight the contribution of each pixel by its gradient. Due to time constraint, this is not implemented.

2.4 Analysis

This section will discuss one particular case where the registration fails to provide a good estimate. Figure 1 demonstrates the problem. One possible explanation is that there are two strong vertical features that are in the images. One is the boundary of the curtain. The other is the edge between the black shelf and the cabinet. The black region is of the curtain has few features. Therefore, the algorithm has to rely on the small region on the right hand side of both images for registration. In this case, the algorithm cannot tell the difference between the two strong features and it results in a mis-registration.



a) The first input

b) the second input

c) resulting image after registration

Figure 1: a) and b) are inputs for image registration.
c) is the resulting image from the inputs. Note that the registration matches the boundary of the curtain of a) to the boundary of the cabinet in b)

3. Stitching Images

The image stitching algorithm takes a list of image filenames as inputs and constructs a mosaic of the input images. The algorithm extracts each image, warps it and passes it into the image registration routine. It then takes the returned displacement and uses feathering to smoothly paste the displaced image onto the mosaic. The completed mosaic can then be viewed using the OpenGL viewer as described below.

4. User Interface

We created a GUI interface to view mosaics. The interface breaks up a given panorama into chunks of size 2^n x 2^n to enable OpenGL to use the chunks as textures. These textures are then each mapped onto camera facing quadrilaterals arranged in a cylindrical formation. The user then rotates the camera around the vertical axis thus panning the mosaic. The interface also provides a button to initiate the mosaic construction and provides browse window to select the input images sequences.

Figure 2: The user interface showing a hand stitched panorama (left).

5. Results & Future Work

Figure 3 shows one resulting panoramic mosaic. It is hand-stitched from two smaller mosaics that were generated automatically. They are obtained from 46 and 8 images respectively. The reason is that the algorithm does not register the three images between the two sequences as discussed in 2.4. Notice that the mosaic creeps upward. This may be due to lens distortion uncorrected by the warping.

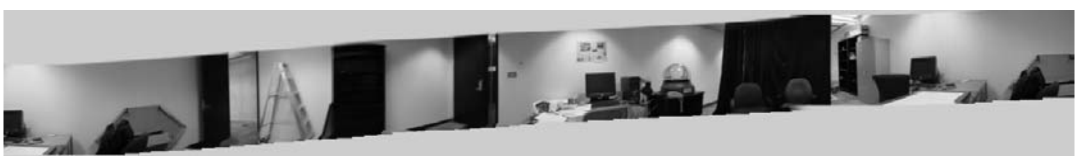


Figure 3: The resulting panoramic mosaic.

The algorithm actually works with fairly sparse camera input and can handle camera shifts of up to half of the image width given sufficient features (Figure 4). In the results shown here the algorithm was not provided with any initial guess as to the translation between images. We found that supplying a guess can enable the algorithm to stitch even more sparse data sets. A guess for each image pair would be needed and this could be implemented in the user interface if more work was done. Also a global alignment routine could be attached to the image stitching program to produce clean 360 degree mosaics.

a) Sparse input from camera

Figure 4: b) The computed mosaic

6. References

[1] J.R. Bergen, P. Anandan, K.J. Hanna, and R. Himgorani. *Hierarchical Model-based Motion Estimation*

[2] B. Lucas, T. Kanade. *AnIterative Image Registration Technique with an Application to Stereo Vision*

[3] R. Szeliski. *Video Mosaics for Virtual Environment*

[4] R. Szeliski, H. Shum. *Creating Full View Panoramic Image Mosaic and Environment Maps*

[5] R. Fisher, S. Perkins, A. Walker, E. Wolfart. *Spatial Filters Gaussian Smoothing*, http://www.dai.ed.ac.uk/HIPR2/gsmooth.htm