# Introduction

Algorithms for aligning images and stitching them into seamless photo-mosaics are among the oldest and most widely used in computer vision. Image stitching algorithms create the high-resolution photo-mosaics used to produce today’s digital maps and satellite photos. They also come bundled with most digital cameras and can be used to create beautiful ultra wide-angle panoramas (see Figure 1).



Figure The view from the top of the Freedom Tower. The camera was in operation for five hours, capturing 567 images, which were then stitched into a massive, 14.5-gigabyte image.

What, then, are the essential problems in image stitching? As with image alignment, we must first determine the appropriate mathematical model relating pixel coordinates in one image to pixel coordinates in another. Next, we must somehow estimate the correct alignments relating various pairs (or collections) of images. Once we have aligned the images, we need algorithms to seamlessly cut and blend overlapping images. These issues will be discussed in the following sections.

# Movement Model

Before we can register and align images, we need to establish the mathematical relationships that map pixel coordinates from one image to another. A variety of such parametric motion models are possible, from simple 2D transforms, to planar perspective models, 3D camera rotations, lens distortions, and mapping to non-planar (e.g., cylindrical) surfaces. (see Figure 2 for illustration of possible movement models). In this project we assume a translational movement model. That is, between two consecutive images each pixel in an image moves *u* pixel in the x-direction and *v* pixels in the y-direction, as illustrated in Figure 3.



# 

Translation

Perspective

3D rotation

Affine

Figure Movement models

***I1***

***I2***

**u**

**v**

Figure

# Computing the translation using Lucas-Kanade

We describe how to estimate the motion between two gray-scale images I1; I2 using a simple motion model of 2D Translation:

For every pixel *(x,y)*, we assume a motion of *(u,v).* That is, the motion is constant for all pixels (Translational model). The equations are derived from the constant brightness constraint:

) =

The Taylor expansion of I2 around *(x,y)* gives:

) =

Combining the two equations yields:

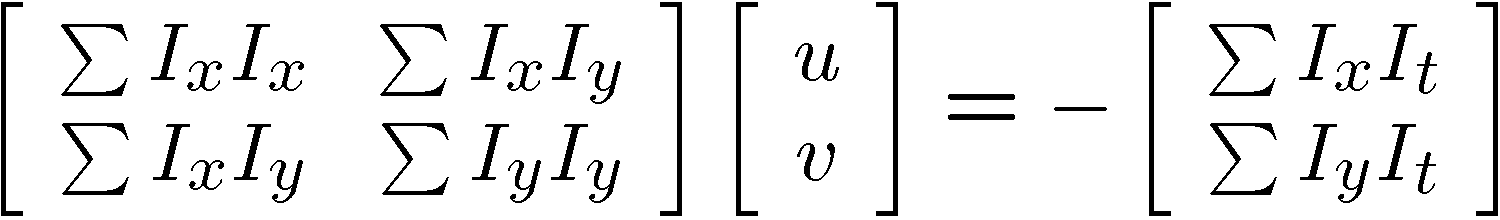
= 0,

where, and .

We wish to minimize the least square error:



Using the normal equations we get the set of equations:



The algorithm outline is as follows, representing the motion *s=(u,v)*

1. Compute a Gaussian pyramid for both images of k levels, *Pyr1* and *Pyr2.*
2. Start with some initial guess (e.g., s = (0,0)).
3. For every level of the pyramids j from k to 0:

* Update *s* according to this level.
* Compute the derivate images *Ix* and *Iy*.
* Compute the structure tensor A.
* Iterate until convergence:
  + Warp I(t+1) according to the current motion estimate to get I(t) using bilinear interpolation.
  + Compute
  + Solve for the motion equations and update

# Image Stabilization

As a preprocess to the stitching stage a limited stabilization of the sequence was implemented: input images were warped to cancel the vertical and sub pixel horizontal components of the motion. This enables an easier stitching procedure later.

In this we leave only the horizontal integer part of the motion. For example: If the motion between two successive images was u = 2.4 and v = 3.7, we warp the second image towards the first image such that the motion between them after the warping will be u = 2 and v = 0.

# Stitching images using graph-cuts.

****

Figure

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 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 defines a node in the graph. Edges are defined between each node and its 4 neighbors: two vertical neighbors and two horizontal neighbors (nodes in the boundary of the overlap area will have fewer 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 taken from the stitched image (B). The weights on the edges are defined as follows: let *u* and *v* be two adjacent nodes in the overlap region. Also, let A(u) and B(u) be the colors of the pixels at the position *u* in the panorama and in the new image, respectively. We define the weight W on the edge between the two adjacent nodes *u* and *v* to be:

In addition to the weights on the edges there are two 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.

# Results

****

# Running the code

1. Make sure you include the directory containing the code and it's subdirectories into Matlab's path
2. Run the mosaic.m file with the parameters as shown in the comments.

# References

B.D. Lucas and T. Kanade “***An Iterative Image Registration Technique with an Application to Stereo Vision***” IJCAI '81 pp. 674-679

S. Baker and I. Matthews“***Lucas-Kanade 20 Years On: A Unifying Framework***” IJCV, Vol. 56, No. 3, March, 2004, pp. 221 - 255.

Kwatra, V., Schödl, A., Essa, I., Turk, G., & Bobick “***Graphcut Textures: Image and Video Synthesis Using Graph Cuts***” In *ACM Transactions on Graphics (ToG)* (Vol. 22, No. 3, pp. 277-286). ACM.

Szeliski, Richard. “***Computer Vision: Algorithms and Applications***”. Springer, 2010.‏