CS512 final project: report

Gady Agam
Illinois Institute of Technology

Abstract

1. Problem statement

Panoramic mosaics, or image stitching, is one of the oldest problem in computer vision. The purpose of it is to merge images or photos which describe the same scenery into a single graph. Nowadays the problem has been solved quite well and widely used on digital cameras and smart phones. For example, Iphone has merged this technique into there camera application to take a panoramic scenery.

This project will base on the paper [1] and [2].

2. Proposed solution

The first step is to extract and match SIFT features between all of the images. It use SIFT features because it's invariant under rotation, scale changes and partially affine change. For each matching point pair (Px, Py) between two images, we accept it only when Py is much better then the second candidate Py' (0.8*distance of (Px,Py') > distance of (Px,Py)).

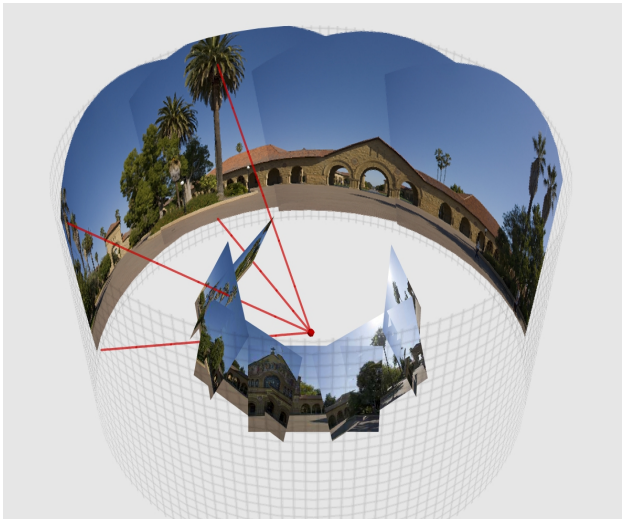
The second step is to matching the images. We assume all images that inputed are all in the same panorama. We don't have to find all matches, so for each image we only consider at most 6 “best image matches” base on the number of feature matches. After determined the matched images, we calculate the Homography matrix between them. To do this, we use RANSAC algorithm. The way we compute the Homography matrix is the same as Assignment 2 (co-planar version). Although two images may have relatively large number of matched feature points between them, it is also possible that most of them are outliers. So after image matching we use the “inliers” and “outliers” from the RANSAC to verify the correctness of image matching. Although the analysis process is kind of complex, the result formula is simple:

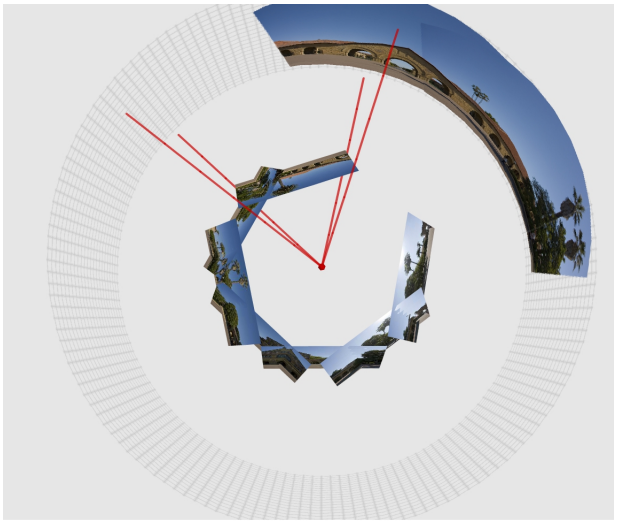
$$n_i > \alpha + \beta n_f$$

Where n_i is the number of inliers and n_j is the total number of feature matches. Here we let alpha=8.0 and beta=0.3.

The third step is Bundle Adjustment. Now we have Homography matrix, but we still need to find matrix K and R of each image (something likes camera calibration). So this step uses Leveberg-Marquardt algorithm to minimize an error function in order to find these R and K matrix. We can use this R and K to warp the original images onto cylindrical or spherical coordinate system. If we don't use these coordinate system, there will be a problem of the result panorama. I will talk about it later. After this step, we can say that the images are “basically in the right positions on the panorama”.

I want to say something more about the cylindrical coordinate. Although my program implement (but doesn't use because of the un-implement of Bundle adjustment) spherical coordinate, I found some image that can explain the cylindrical coordinate nicely:





The red point in the medial is the camera. We assume the scenery is far from the camera so we can pretend that the camera is not moved but only rotated when taking the photos. Because the camera rotated when taking photos, the 3D relationship of the photos is showed around the red point. The photos are tangent to a circle whose radius is the focus length of the camera. Now we need to project the photos onto the outer cylinder to get a panorama. Let u be the angle of horizontal rotation in radian, v be the vertical coordinate (similar as y); (x,y,z) be the 3D coordinate of the point in the photos. Then:

$$u = \alpha \tan^{-1}(x/z); v = \alpha\, y/\sqrt{x^2+z^2}$$

Alpha is the radius of the outer cylinder of above figures. To output the result warped panorama, we only need to let (u,v) be the (x,y) coordinate (u needed to multiplied by 180/PI).

The warp of spherical coordinate is similar, but both u and v are angles in radian:

$$u = \alpha \tan^{-1}(x/z); v = \alpha \tan^{-1}(y/\sqrt{x^2+z^2})$$

The fourth step is Gain Compensation. Since the images may under different exposure conditions, the brightness of them may be different. This step uses the same algorithm as last step to minimize some error to find the right gains of each images.

The last step is Multi-Band Blending. After all the steps above, the result panorama may still have many problems like: 1) Parallax. Because of distortion or mis-registration, the images may not be totally coincide. The small difference in the overlap area will cause image blur or ghosting. 2) Vignetting. The intensity may decreases towards the edge of the image, making the original edges of the images obvious in the result panorama. Except the vignetting, the edge may still be obvious because of other factors. So we need a way to blend the image. The paper use Multi-Band Blending. What it doing is blend low frequencies of the image over a large spatial range, and high frequencies over a short range. Typically we can use Laplacian pyramid to separate the frequencies, but the paper use Difference Of Gaussian instead.

I plan to do the step 1,2,5, and do step 3 if possible. But because of the complexity of step 3, I haven't finished the debugging yet. It seems like that I may misunderstand some details. So now, the rest of this report will base on step 1,2,5 and gives some comparisons of the situation that if step 3 is implemented. These steps will produce a panorama but with several problems:

- The algorithm can do well without step 4 if all images as the same or similar intensities. The step 5 can blending the edge so that it can deal with small difference of intensities. But if the difference is big, it may be obviously. Actually in experiment, if we blend the lowest frequencies in a very large spatial range, the multi-band blending can deal with large

intensity difference in a satisfying level.

- Without step 3, we can't get the rotation(R) and intrinsic(K) matrix, means that we can't warp the image onto cylindrical or spherical coordinate. If we only use the Homography matrix to transform the image and combine them, we do can get the panorama. But the further the pixel from the panorama "center"(actually it's the first image that all other images stitching onto), the more the pixel stretches. So if we are stitching a long panorama, it will shaped like "bow tie". Some example will be given later.

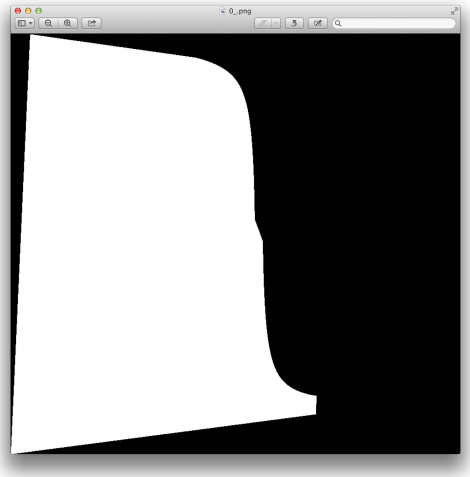
Actually my program do can project the images onto spherical coordinate, but without parameters R and K, these codes are commented.

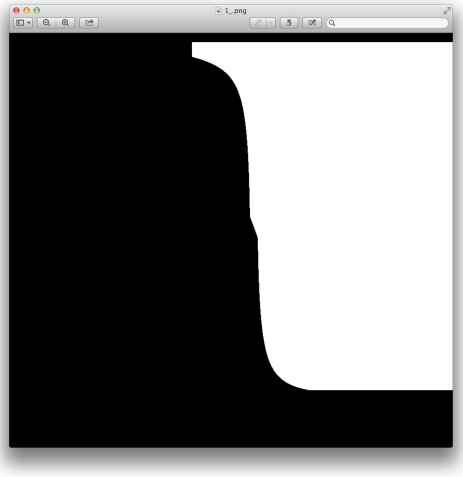
3. Implementation details

Since step 1 and 2 are similar as some steps in previous Assignments, I will bypass it.

For step 5, at first we need to decide which image that each pixel in panorama belongs to. For each image, we give weight to each pixel. The centre pixel has weight 1, and the edge pixels has weight 0. Weight function: W(x,y)=w(x)w(y), where x and y are the coordinate of the pixel, and w(x) varies linearly from 1 at the centre to 0 at the edge.

Then we project the images onto the coordinate of the panorama. The weight parameters are also project onto the panorama coordinate. So now each pixel Pi in the panorama may corresponding to several pixels in different image with different weight. We let Pi belongs to the image that has the max weighted pixel. If Pi belongs to Pa in image A, then update the weight of Pa to 1, otherwise the weight =0. Now the weight parameters became the "mask" of the image. Here's an example of the "weight images" of two matched image:





Now we begin to decompose the frequencies of the images. For image i:

$$I^i_{(k+1)\sigma}(x,y) = I^i_{k\sigma}(x,y) * G_{\sqrt{((2k+1)\sigma)}};\ where\ I^i_0(x,y) = I^i(x,y)$$

$$B^i_{(k+1)\sigma}(x,y) = I^i_{k\sigma}(x,y) - I^i_{(k+1)\sigma}(x,y)$$

$$W^i_{(k+1)\sigma}(x,y) = W^i_{k\sigma}(x,y) * G_{\sqrt{((2k+1)\sigma)}}$$

Let me explain. (*) denotes convolution. I is the image, $B_{(k+1)sigma}$ is the image that represents spatial frequencies of I in the range of wavelengths [k*sigma,(k+1)*sigma]. We convolute the I

to a gaussian kernel with standard deviation equals sqrt((2k+1)sigma) and update I (we will call this sqrt(...) “ascending sigma”), result to a stack of Is with different level of gaussian blur. We then calculate the difference between two neighbors in the stack, and represent them as Bs. We do the same thing to the weight image.

Now each image has been decomposed into a stack of Bs. We need to blend the Bs on the same level in the difference stacks. For level k, we blend the Bs in to image P:

$$P_{k\sigma}(x,y)=\frac{\sum_{i=1}^{n} B_{k\sigma}^{i}(x,y)W_{k\sigma}^{i}(x,y)}{\sum_{i=1}^{n} W_{k\sigma}^{i}(x,y)}$$

Notice that the larger k is, the lower frequencies B represents, and more blurry W becomes. So we blur high frequencies in a short range and low frequencies in a long range. Finally, sum up all image Ps and we can get the result panorama.

In practice I set the sigma to 2 pixel. So that the first B will contains some detail informations like edges and color changes in small area.

In practice separating each image into 3 frequency ranges (3 B images in the stack) is enough to get a satisfying result.

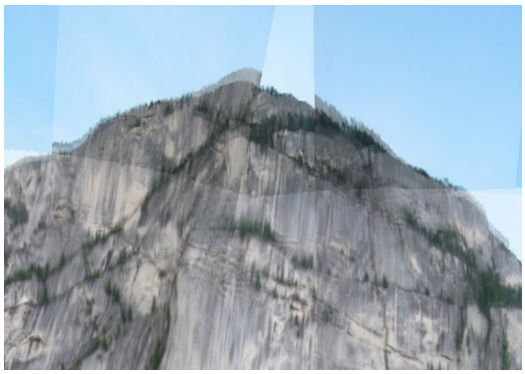
4. Results and discussion

Here is a panorama that without blending:

In this panorama, if two images have overlap area, we just calculate the average color between them. Because of this, we can see that the edges of original images are obvious and the pedestrians on the left side are ghosting.

Here is a picture with Multi-band blending:

We can see that the edges are blurred but the details are still remained. The ghosting phenomenon is gone. Some pedestrians are remained but some are not. Actually, if there're some parallax in the image, multi-band blending can also deal with them. For example:



Without blending

With blending

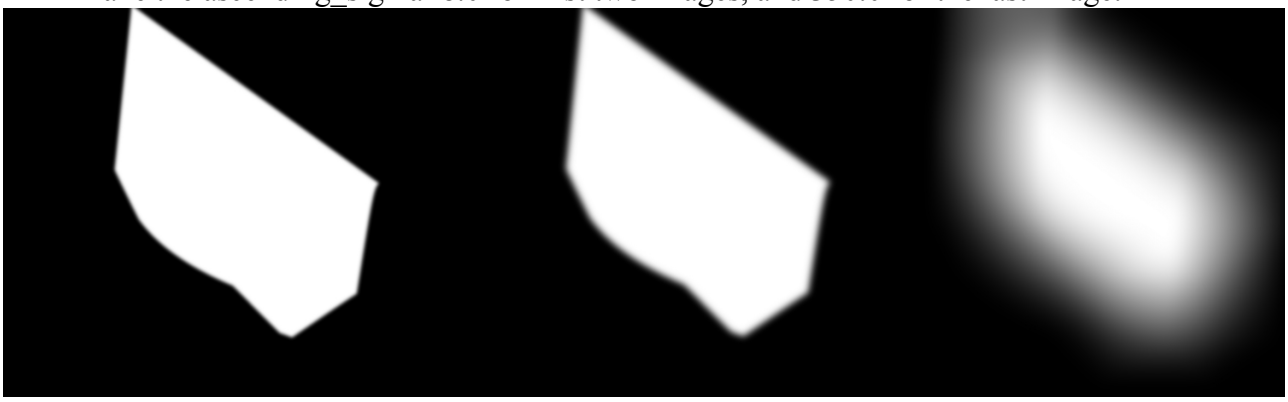
Remember I declared that multi-band blending can deal with large intensity(overall gain) difference in experiment. Here is a panorama that ascending_sigma=20.0, kernel size=ascending_sigma*5+1 for the lowest frequency range:

Here we can see some big intensity differences in the image. If we set the

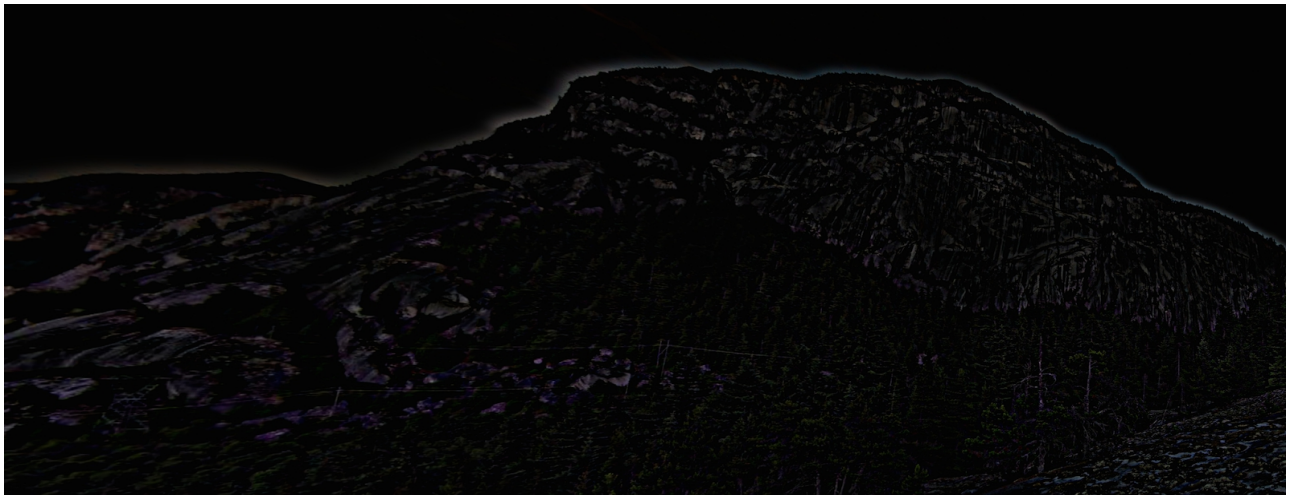
ascending_sigma=350.0, kernel size=ascending_sigma*1.5+1, the blending will becomes:

The multi-band blend does not “correct” the intensity difference but only “blur” it. So if the result panorama is large but the overlap areas are not dense, we can still feel that some part of the image is darker, but the transition is smooth. On the other hand, “Gain Compensation” directly change the overall gains of all images, making all images under similar exposure conditions. Maybe you notices that part of the left forest of above panorama is blurred. This is not caused by multi-band blending but because of the stretching of original image and the difference of resolution in original images.

Below is the three levels of the decomposed weight image. To make the difference obvious, I make the ascending_sigma=8.0 for first two images, and 350.0 for the last image:



Keeping the ascending_sigma unchanged, we give the Ps (panoramas of each level of B):



Level 1 of P

Level 2 of P

Last level of P

Remember that we said if we don't warp the images onto cylindrical or spherical coordinate, the result image will be like a “bow tie” if it's long. Here's an example:

It's not hard to understand. If the view angle is large, like 120 degree, then directly project the image in 3D cylinder/sphere onto a 3D planar will cause stretching when away from the center.

Here is a same panorama that using the sphere coordinate. The camera parameters are set to some reasonable values:

It's much nicer.

Summary: Panoramic mosaics is an old problem in computer vision, and it's nicely solved. The paper I chose gives complete processes that can achieve this purpose. Many algorithms are developed in this area, and the process are not the same. For example, multi-band blending can be replaced by an algorithm called “Poisson image editing”, which can do even better then multi-band blending in some aspect. It is pity that I haven't finish debug of the Bundle Adjustment, but if we only deal with small number of images or the images are not token based on rotation of camera, the program can give nice panorama result.

5. References

[1] M. Brown and D.G. Lowe, “Recognising panorama” *Proc. ICCV* 2003.
[2] M. Brown and D.G. Lowe, “Automatic Panoramic Image Stitching using Invariant Features”, *International Journal of Computer Vision, 2007*
[3] R. Szeliskim, “Computer Vision: Algorithms and Applications”, *2011*
[4] http://sourceforge.net/adobe/adobedatasets/panorama/home/Home/