Flattening The Pot

Amin Azimov and Kamal Aghayev

08/01/2019

1 Introduction

This PDF includes both manual for users and working process.

Last year we have developed a program to cut a strip of size n from a BMP image. This project is a logical continuation of the previous one as it enhances the functionality of the last year's project. This time we present to you the program that is able to:

- calculate Pixel Scaling Factor(PSF) of images in a given directory;
- unwrap images from a given directory using PSF values from a given file;
- glue images from a given directory.

2 Manual for Users

The project consists of *src* directory and *Makefile* that is used to generate output files (*psf*, *resize*, *glue*). src itself consists of *bmp.[c—h]pp* - source and header files to manipulate BMP images, *tensor.[c—h]pp* - source and header files for Tensor class, *marker.[c—h]pp* - source and header files for Marker class, *utils.[c—h]pp* - source and header files for different casual functions, *psf.cpp* - main source file for *psf* program, *resize.cpp* - main source file for *resize* program, *glue.cpp* - main source file for *glue* program, *resize_all.sh* - bash script for resizing all the images from a given directory.

2.1 Makefile

Makefile has only 2 rules:

- **out/*** used to create *out/* directory with executable files;
- **rm** used to delete *out/* directory with all files in it.

2.2 PSF

This program calculates PSF values of the object on the images and saves the result into file. An output file is a simple plain text file consisting of PSF values each on a new line and the maximum shift value at the end. An executable file uses options to control different variables. To see the examples check "Examples" subsection. Following options can be used while executing a file:

- **-D dirname**: To calculate PSF values using images from *dirname* directory (imgs/ by default);
- **-o filename**: To save PSF values into *filename* (psf by default);

- **-s n**: Number of PSF values to be find (10 by default);
- **-d n**: Downscaling factor. Images will be reduced by n to calculate PSF values (4 by default);
- **-S n**: Size of a marker to be used to find PSF values (5 by default);
- **-i n**: Number of images to be analyzed to calculate PSF values (10 by default);
- **-v**: To stop showing progress.

2.3 Resize

This program resizes an image using PSF values. It uses options to control different variables. To see the examples check "Examples" subsection. Following options can be used while executing a file:

- **-p filename**: To use PSF values from *filename* (psf by default);
- **-i filename**: To unwrap *filename* image;
- **-o filename**: To save resized image into *filename* (inputFileResized.bmp by default);
- **-v**: To stop showing progress.

2.4 Glue

This program glues given images. It uses options to control different variables. To see the examples check "Examples" subsection. Following options can be used while executing a file:

- **-s n**: Size of a strip to be croped from images before glueing;
- **-p filename**: To use PSF values from *filename* (psf by default);
- **-i dirname**: To glue images from *dirname* (imgs/ by default);
- **-o filename**: To save glued image to *filename* (glued.bmp by default);
- **-v**: To stop showing progress.

2.5 Resize All

This script uses *resize* program to resize all the images from a given directory. It requires 3 arguments:

1. Directory containing the images to be resized;
2. Directory to store resized images;
3. Name of file containing PSF values to be used.

2.6 Examples

Note that if you don't use any required option, its default value is used.
To create executable files you need to pass to the directory containing *src/* directory and *Makefile*(later called project directory). Then in terminal type **make** command. This should create *out/* directory with programs and script. It's required that you run *resize_all.sh* script from a project directory.

2.6.1 PSF

To find PSF values of images from *imgs* directory and save the values into *psf* file use the following command:

```
./out/psf
```

To find PSF values of images from *pot-360/* directory and save them into *psf* file using *20* steps, marker size of *10*, *50* iterations downscaling images *6* times use the following command:

```
./out/psf -D pot-360/ -o psf -s 20 -S 10 -i 50 -d 6
```

2.6.2 Resize

To resize *img.bmp* image use the following command:

```
./out/resize -i img.bmp
```

To resize *img.bmp* image using PSF values from *somepsffile* file and to save it into *output.bmp* file use the following command:

```
./out/resize -i img.bmp -o output.bmp -p somepsffile
```

2.6.3 Resize All

To resize images from *pot-360/* directory using *psf* file and to save them into *output/* directory use the following command:

```
./out/resize_all.sh pot-360/ output/ psf
```

2.6.4 Glue

To glue images from *in/* directory use following the command:

```
./out/glue -i in/
```

To glue images from *in/* directory using strip size of *100* and save glued image into *output.bmp* file use the following command:

```
./out/glue -i in/ -o output.bmp -s 100
```

To glue images from *in/* directory using PSF values from *somepsffile* file and save glued image into *output.bmp* file use the following command:

```
./out/glue -i in/ -o output.bmp -p somepsffile
```

Note that you cannot use strip size together with some PSF file, you need to choose one of two possibilities.

3 Working Process

As recommended we did not start the project from the beginning but used the previous project and upgraded it. The biggest change is that we passed to C++ to be able to use classes and STL containers. Since we developed since the last year several functions were rewritten and fixed but the basic concepts and ideas stayed the same.

3.1 Pixel Scanling Factor

We decided to do more than we were required and write an algorithm that will calculate PSF values instead of us.

3.1.1 Observations

We tried to understand how human being solves the same problem and came to the solution that men try to find not the point but the area where the needed point is located. We as men try to remember the area where the needed point is located and find the same area on the next image. Moreover, since we have pot photos taken after 1 °rotation the lighting of the same spot can be a bit different which can totally change the values of a certain pixel. The rotation on the images are not very precise, so in order to find PSF values of a pot ,we need several comparisons.

3.1.2 Algorithm

The idea is to use markers in order to find the same spot in two different photos. First, we try to find the most distinguishable area on a first photo. Here can be used several approaches, but we found our first approach good enough to be used in the final program. In order to find such an area, we get the grayscale value of the spot. Then we find the standard deviation of the values of this spot using the following formula:

$$\sigma = \frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n-1}, \tag{1}$$

where n is the result of a product of width and height of the spot.
If the values in the spot do not differ at all σ will be equal to 0 and more the values differ bigger σ will be. After we find the most distinguishable spot on the first image we need to find the same spot on the next image. Again, there are more than one solutions for this problem and we used several of them and will present you only two.

1. Given values of a spot on the previous image and on the current image to find their difference. If two spots are the same, then all the values will be close to each other and to 0. Then we find a standard deviation of the difference. If the spots were the same standard deviation will be close to 0 and more the spots differ bigger the standard deviation will be. The goal of this solution is to find the spot with the minimal standard deviation in difference with the original spot. The con of this algorithm is that if there is a similar spot with different lighting on the image it can be considered as the result since standard deviation shows the deviation of the values.

2. Consider a spot as an n-dimensional vector. Given values of a spot on the previous image and on the current image to find the distance between two spots using the following formula:

$$\sqrt{\sum_{i=1}^{n}(x_{0_i} - x_{1_i})^2}, \tag{2}$$

   where x_0 and x_1 are spots and x_{0_i} and x_{1_i} are i-th pixels of the spots.
   If two spots do not differ at all the result of the calculation will be 0 and more two spots differ bigger the result will be. This solution was used in the final version of a program.

Both of the aforementioned solutions will not give a 100% accuracy due to two points.

1. The values of the original spot may have a small standard deviation which will lead to the situation when a lot of spots are close enough to the original.

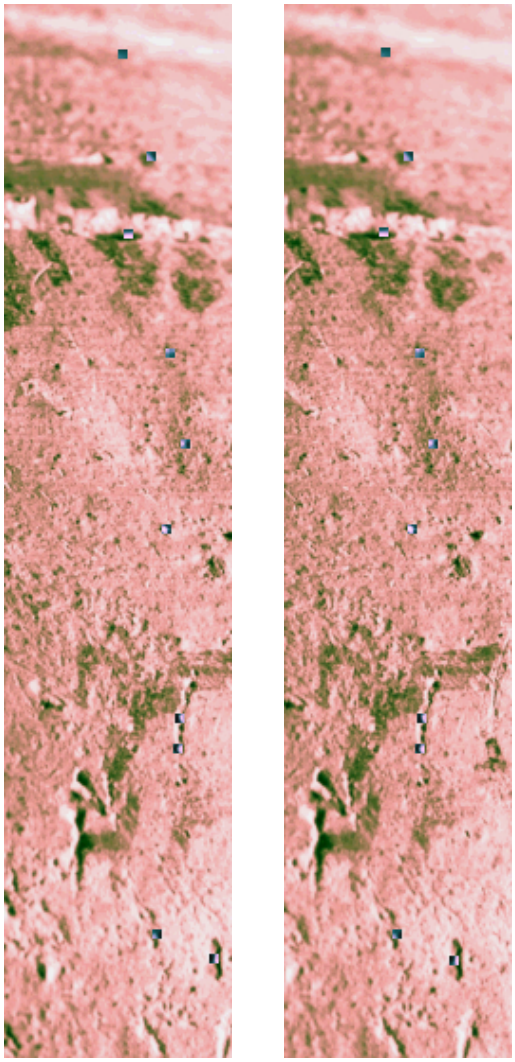
2. There can be two similar spots on the image that can be confusing for the algorithm.

In order to avoid mistakes as mentioned before we calculate PSF values not on only 1 pair of photos but on more than 1 pair of photos and then find the "average" of the calculations. This step seemed to be very simple, however, we came up with two solutions for this step.

1. After finding n PSF values, find the mean of the values and use it as the final result. This solution is good not considering the big mistakes that algorithm may make. If one of the PSF values differs from the whole population, the final result may differ from the real value.
2. After finding n PSF values, find the median of the values and use it to find and delete "fake" values in order to get only "real" values. Then, find the mean value of the remaining data and consider it as the final result. This solution shows itself a lot better in the case of the presence of "fake" values.

The result of the aforementioned algorithm is good but slow. In order to make it faster, we decided to downscale the input images.
As shown in the practice, to get a better result of calculating the PSF values better is to choose bigger marker size, greater the number of iterations and less the downscaling factor, but it can be time-consuming.

3.1.3 Results



(a) Image with markers on the areas with maximum STD

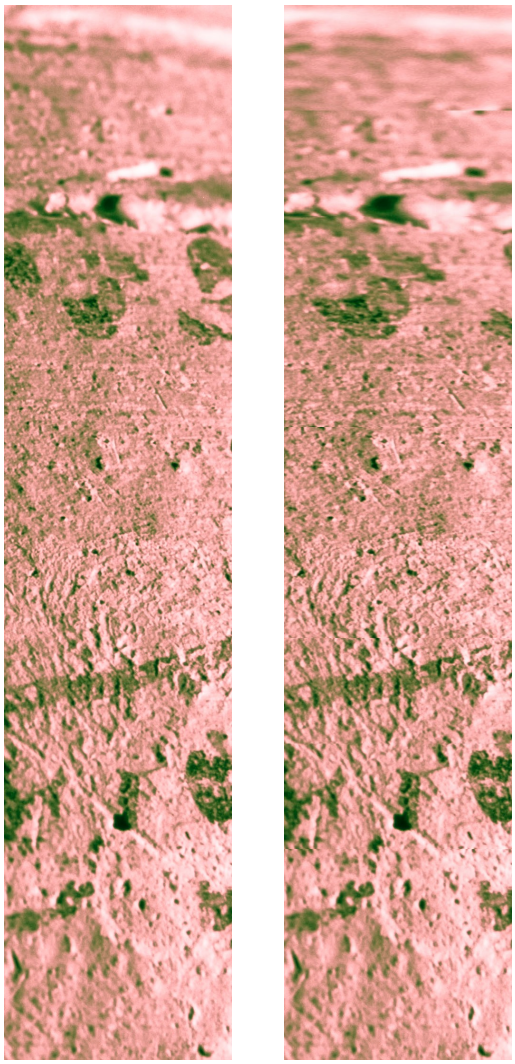
(b) Image with markers in the same areas as on the first image

Figure 1: Result of the algorithm calculating PSF

As can be seen on the **Figure 1**, results of the algorithm are pretty good. Due to the fact of excluding the "fake" data points, PSF values can be considered to be correct.

3.2 Resizing

The idea of the algorithm is that given an image resize it using PSF values for the different parts of an image. We need to define how can we write data on the floating point number of pixels on the image. To do this we write the data on the whole part of the floating point number pixels. After we remain with the floating number that is less than 0. To write a data on the number of pixels less than 0 we need to write this amount of the data on a pixel then choose another data that needed to be written and write the new data on the remaining part of the pixel. This operation is repeated until all the data is written. At the end we need to cut the strip from the middle of an image to get the final result. The result of resizing algorithm can be seen on the **Figure 2**.



(a) Original image (b) Resized image

Figure 2: Result of the resizing algorithm

3.3 Gluing

The idea of the algorithm is to glue all unwrapped images. To do so we need to find the part of the images that is unique. Since we resized all the part of an image to have the same radius along whole pot we can consider the pot being cylindrical which can be easily glued by placing all the images one after another. As can easily be proved all the images differ from each other by the amount of maximum PSF value determined earlier. In order to glue all the unwrapped images, we need to cut a strip of a size of the maximum PSF value from all the images and then place them one after another.

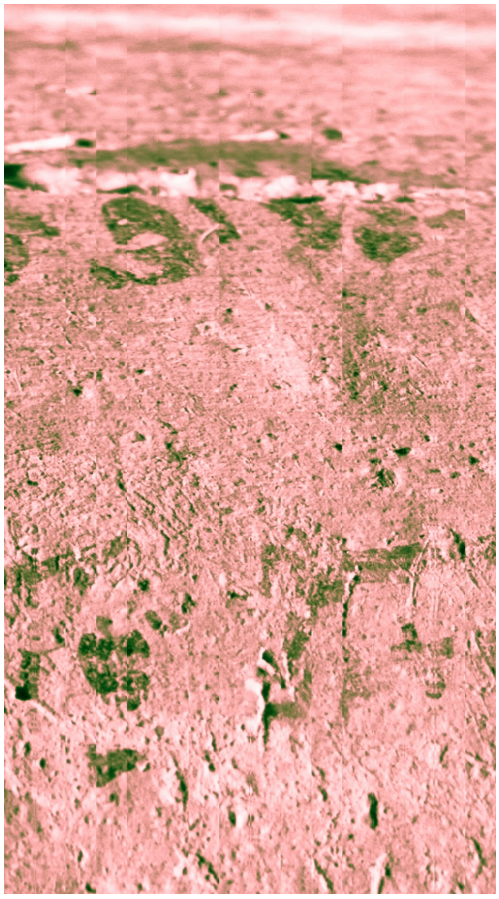


Figure 3: A result of the gluing of 16 images of the pot

The result of gluing of 16 images of the pot can be seen on the **Figure 3** as well as the result of gluing of 360 images of the pot can be seen passing by the following link: https://bit.ly/2M1qrmv (250 MB).