Data Structure and Algorithms
Flattening Pot

Rahim Sharifov, Sara Shamilova

January 2019

1 How to launch

Program to resize given Pot strip AzCroppedPotXXX.bmp. PotXXX.txt is its corresponding txt file that contains shifting values of this image:

gcc resize.c
./a.out AzCroppedPotXXX.bmp PotXXX.bmp .

result: **resizedAzCroppedPotXXX.bmp**

Shell script to resize given 16 image , extract 480 pixel , and stitch together:

bash script.sh

result: **FlattenPOT.bmp (16 resized strip,16 extracted strip**

Finding shifting between two consecutive image:

gcc shifting.c
./a.out AzCroppedPotXX.bmp AzCroppedPot(XX+1).bmp

result: **Pot(XX+1).txt**

In ZIP file that we submit we added 16 txt files with shifting values that we calculated using program beforehand. Because findind shifting is time-consuming process(at least 20 minutes for 16 images). In shell script there is commented section that finds shifting before then resize strips (But there need to be AzCroppedPot24.bmp to find shifting of AzCroppedPot25.bmp).

2 Objective

Objective of thiys final project is to resize given strip,that is taken from Pot, according to its "shifting" values which varies depending on y-coordinate. If we take 2 horizontally same but vertically different points from AzCropped-Pot25.bmp and switch to AzCroppedPot26.bmp, if look up these 2 points, we will see that they are not horizontally same anymore,because of shifting.This comes with a problem if we want to stitch these pictures,pixels will be double as was shown Figure 3.

3 Our approach

Image needs to be resized so that its shifting at top and bottom becomes equal. To do this we divided image vertically into 22 parts (22 steps). Then we find shifting values of these parts. By dividing max shifting to the rest we find psf values. With these psf values we enlarge each part of image with its corresponding psf value.

3.1 Finding psf

In order to be precise we preferred to find shifting values with program rather than by hand.Program shifting.c takes two inputs which are two consecutive strips of Pot. Compares all 22 parts of one image with corresponding parts of other image. It reads rectangle of pixels with width of 50 pixels and length of 350 pixels from first part of Image and searchs this rectangle in first part of another image by comparing rectangles pixel by pixel. Comparing pixels is challeging because it is a real image so when you rotate the pot because view angle changes also affect of lights rgb values of the same pixel obviously changes. So checking equality won't work. So we subtract rgb values to see how close they are, if difference is lower than 15, pixels close enough to be the same.With x-coordinate of best matching rectangle we find shifting value of this part with some calculation and go on with second part of image and so on. So it finds all shifting values and puts them into txt file.

3.2 Resizing

As we have program to find shifting values of image. We use it to find shifting values of each image,so each image has its own shifting values. So, program resize.c takes 2 inputs which are pot strip and corresponding .txt file with shifting values. Program opens this txt file, finds maximum shifting value. By dividing all shifting values to the maximum shifting optain psf values. With these values and scaling algorithm it resizes each part of image with corresponding psf value. Program opens a target file resizedAzCroppedPot.bmp with

$$newsize = actualsize * maxPsf$$

With variables CoeffBuffer, Coeff , formula for nth pixel, Tn means nth Target Pixel Sn means nth Source Pixel.

$$T_n = coeff * S_n + (1 - coeff) * S_{n+1})$$

Program finds nth target pixel by this formula.

Example: psf=1.75
We need enlarge every pixel by 1.75.At the beginning coeffBuffer is equal to psf. If coeffBuffer greater that 1 coeff become 1 that means we first need to take pixel itself. So coeff=1 and coeffBuffer become 0.75.So eqation becomes:

$$T_1 = 1 * S_1$$

Program checks now coeffBuffer is less than 1 that means we dont need whole pixel we just need part of it , and some part of another so coeff equal to coeff-Bufer that is 0.75 and equation becomes:

$$T_2 = 0.75 * S_1 + 0.25 * S_2$$

Because we used 0.25 portion of second pixel this is why

$$CoeffBuffer = psf - (1 - coeff) = 1.5$$

Now program check if coeff is less than 1 which is true this means we are done with first pixel of fSource , so it reads new pixel. So we are at 3rd pixel of ftarget. CoeffBuffer=1.5 is greater than 1, so coeff=1 and coeffBuffer=0.5. formula becomes

$$T_3 = 1 * S_3$$

. coeffBuffer less than 1 coeff=0.5 coefBuffer=psf-(1-0.5)=1.25

$$T_4 = 0.5 * S_2 + 0.5 * S_3$$

And so on, this way program resizes the picture shown in the Figure 2.

3.3 Stitching

After resizing, extracting 480 pixels from resized image, we now stitch these extracted strips. Because we had shifting values of all strips. Biggest shifting value is the quantity that represent how many pixel we need to stitch from that image. So program stitch.c takes all resized and extracted strips as an argument. From names of arguments it takes their numbers one by one and opens corresponding txt file and finds biggest shifting value and stitch that amount of pixel.

4 Conclusion

As a result optained stitched image is better than before (Figure 3). As we resized every image doubled pixels are minimized.

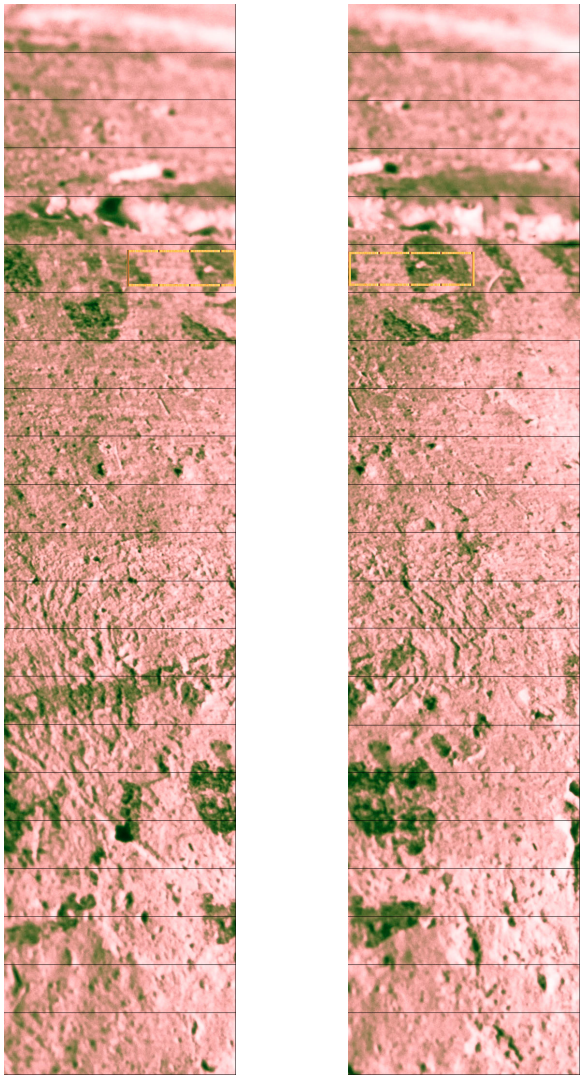

Figure 1: Comparing Rectangles

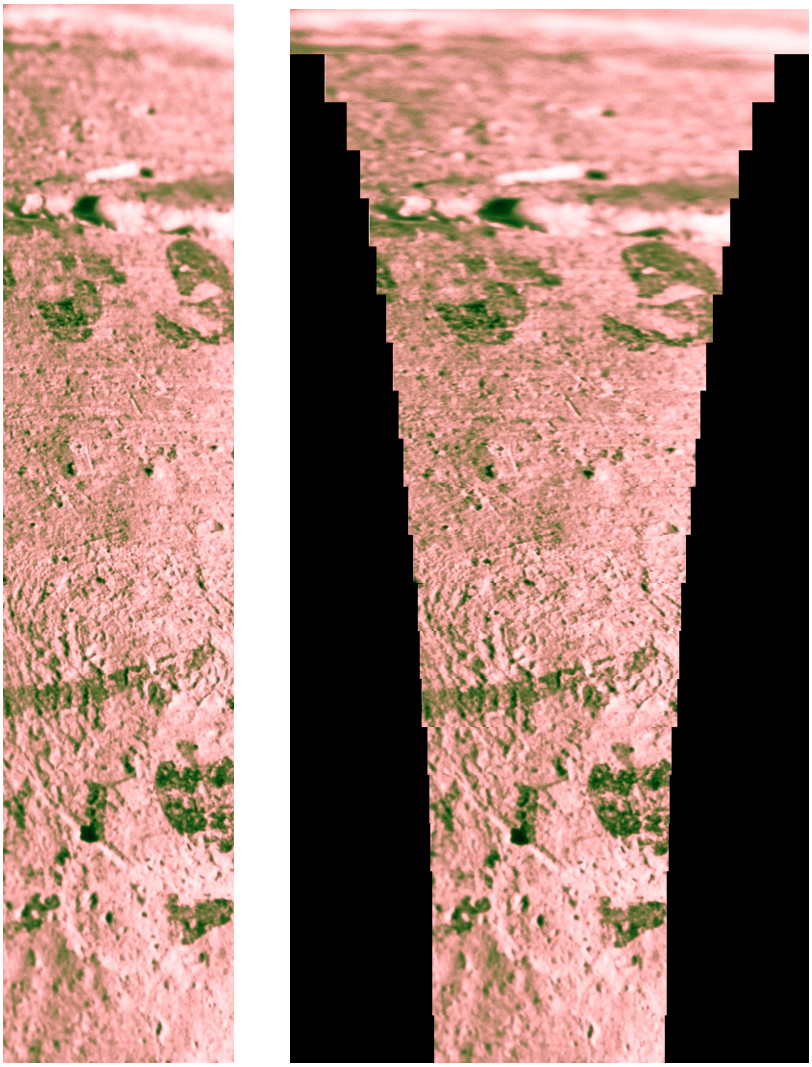

Figure 2: resizing

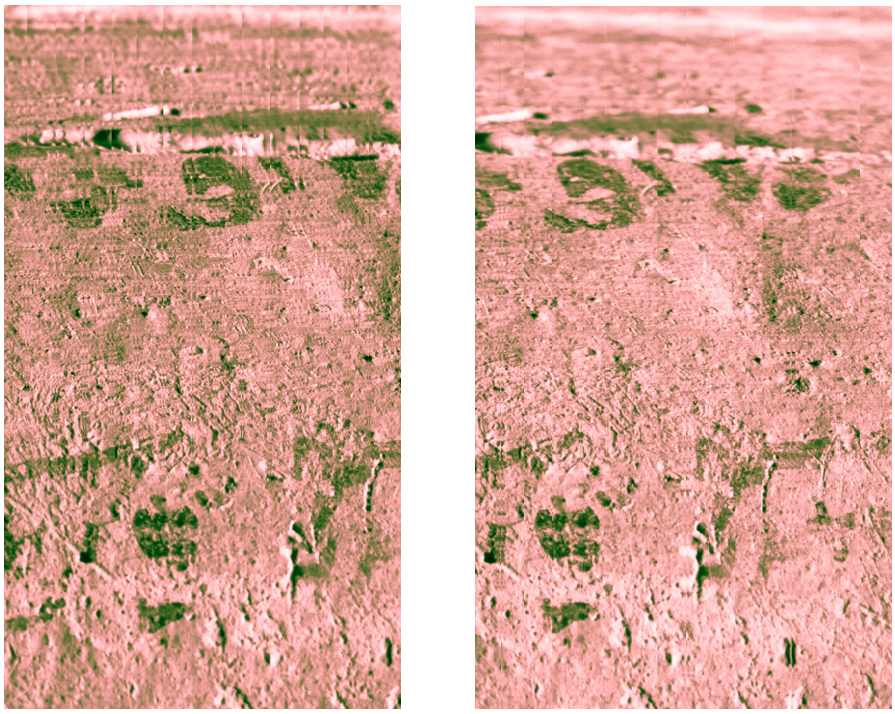

Figure 3: Stitching before resizing and after resizing