COMPUTATIONAL PHOTOGRAPHY

Project 2 Report

MAY 26, 2020

MATTESINI RICCARDO - BUNCHALIT EUA-ARPORN
2019280795 - 2019280794

Contents

Selecting topic

The first choice to be made is how to represent the final image. We are going to stitch together only three images, and so it makes sense to use a flat compositing surface (a rectangle with size proportional to the images size).
Once the source pixels have been mapped onto the final composite surface, we have still to decide how to blend them in order to create an attractive-looking panorama. If all of the images are in perfect registration and identically exposed, this is a trivial problem. However, visible seams (due to exposure differences), blurring (due to mis-registration), or ghosting (due to moving objects) can occur.

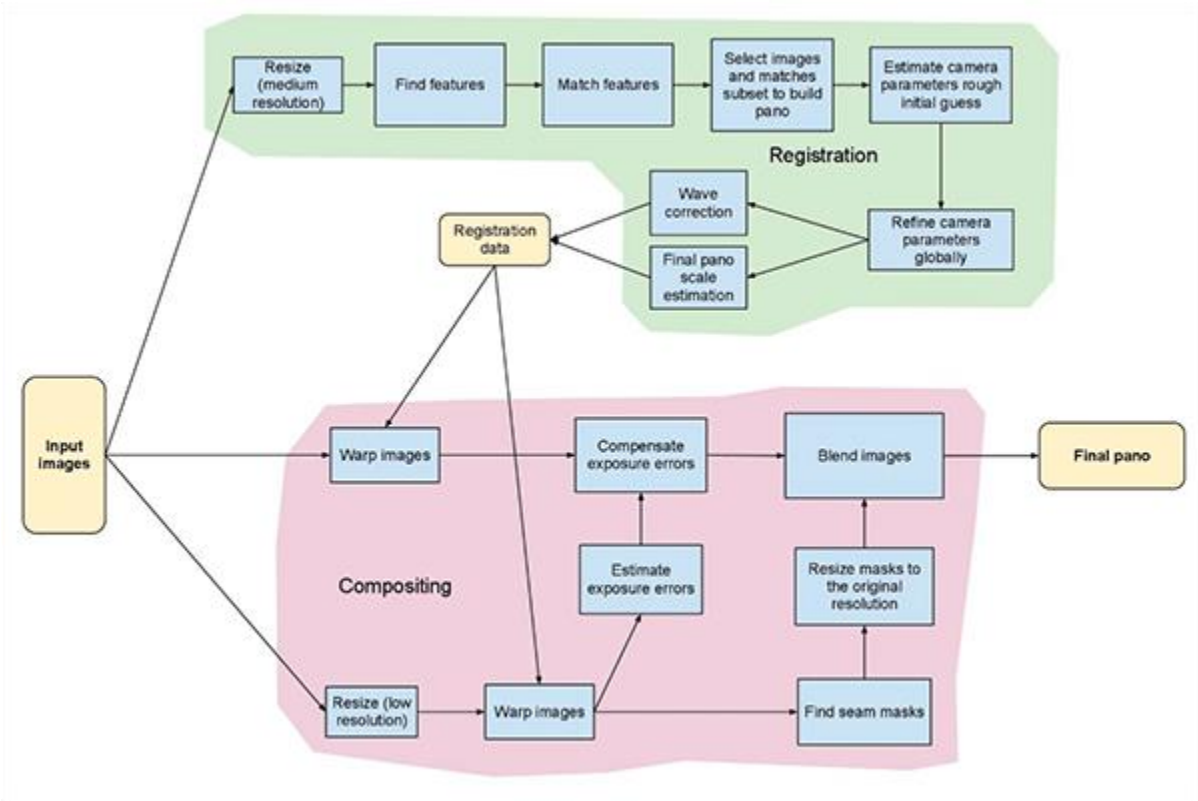

Figure 1 Image stitching pipeline

Creating clean, pleasing-looking panoramas involves both deciding which pixels to use and how to weight or blend them.
For the project, we decided to investigate every aspect of the image stitching pipeline, in order to have a good overview of the whole process and understand possible outcome of our experiments in a more complete way, i.e. not just check if the image is good-looking or no and "randomly" change parameters till it looks great.

However, we will concentrate in the “blend images” process, testing different approaches and trying to define pros and cons of each of them.
Finally, some possible improvements are proposed, based on test that we did and can be used both live and as post-processing techniques.

Survey

Warp images

In our implementation and experiments, image warping was widely used and more specifically, perspective projections were used. This is a natural approach in the case of few images, where we select one of the images as thereferenceand tothen warp all of the other images into its reference coordinate system. The result is a flat panorama.
However, if we were to consider wider fields of view, this kind of representation would stretch the pixels near the border excessively. In this case, a cylindrical projection would be more appropriate, with, for example, a final cut of the black borders to make it “flat”. Follows a brief description of its working mechanism.
Image warping works by applying a function to the images’ domain to obtain rotations, translations or general warps.

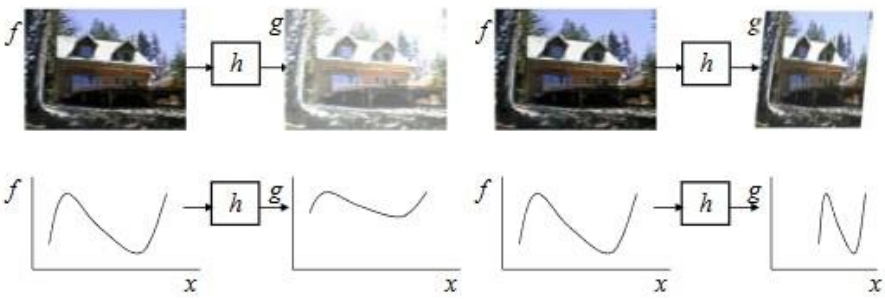

Figure 2 Left: range modifications for point processes, Right: domain modifications for image warp

The function takes the form

$$g(x) = f\big(h(x)\big)$$

The main difference with point processes is that the latter transforms the range of the image, not its domain, and takes the form

$$g(x) = h\big(f(x)\big)$$

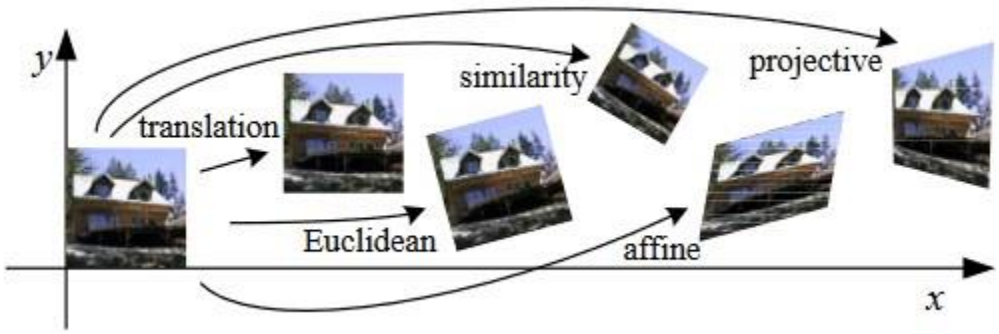

Figure 3 Basic set of geometric image transformations

This brief introduction of the warp image technique will continue in the experiment section where we describe the reasons for adopting the perspective transformations over the others.

Feature detection

Feature detection is a method for computing abstractions of image information and making local decisions at every image point whether there is an image feature of a given type at that point or not. The resulting features will be subsets of the image domain, in the form of isolated points, continuous curves or connected regions.
We made use of three different feature detection algorithms: SIFT, SURF and ORB. We address in the experiments section the properties, up- and down-sides of these three methods.

Blending

Image blending is aimed to remove artifacts which are produced when stitching different images together (due to various reasons such as exposure difference, small motion of the objects, etc.). After the input image have been mapped onto the final composite surface, the next step is to apply image blending to create a final aesthetic panorama.
We focused on two approaches for image blending which are Alpha blending and Laplacian pyramid blending.

Alpha blending

Alpha blending is the process of combining a translucent foreground color with a background color, thereby producing a new color blended between the two. In addition to the red, green, and blue components of each color, there is an additional optional fourth component, referred to as the color's "alpha." Alpha means transparency and is particularly useful when you want to draw elements that appear partially see-through on top of one another. The alpha values for an image are sometimes referred to collectively as the "alpha channel" of an image. Alpha Blending is based on Compositing Digital Images by Thomas Porter and Tom Duff. It is a convex combination of two colors allowing for transparency effects in computer graphics. Note that Alpha = 1 for opaque and 0 for transparent. The equations for blending "Image a" over "Image b" are

$$I_{blen} = \alpha_a I_a + (1 - \alpha_a)\alpha_b I_b$$

$$\alpha_{blen} = \alpha_a + (1 - \alpha_a)\alpha_b$$

Laplacian pyramid blending

Pyramid is when we work with images of the different resolution of the same image. For example, while searching for something in an image. To solve this, we will create a set of images with different resolution and search for an object in all the images. These set of images with different resolution are called Image Pyramids. In order to create image with higher resolution, we up sample it by using $g(i,j) = \sum_{k,l} f(k,l)h(x - rk, y - rl)$ and create image with lower resolution by down-sampling $g(i,j) = \sum_{k,l} f(k,l)h(ri - r, rj - r)$. When coming to Image blending, Pyramids gives seamless blending to the images.

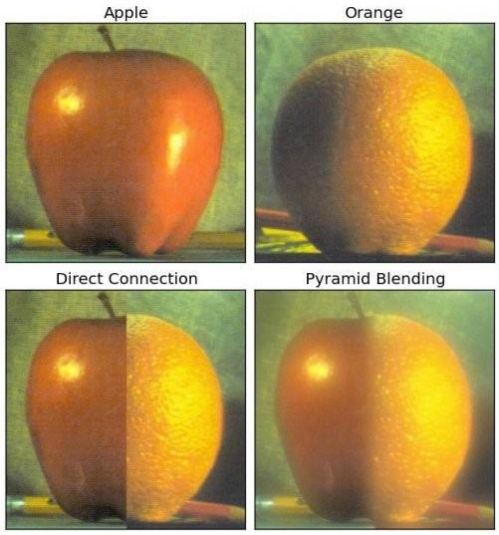

Figure 4 Typical example of Laplacian pyramid blending

Experiment and Comparison

In this section we summarize what was done, including trials and unsuccessful strategies adopted. Experiments to validate the reason for the choices of feature detection algorithms as well as blending algorithms and others are also presented here.

Warp images

Continuing from the previous section, we are going to describe the parametric transformations, including the perspective transformation that was used in the other experiments as well.

Parametric

The first kind of transformations are the parametric transformations, which apply a global deformation to the image and are controlled by a small number of parameters.

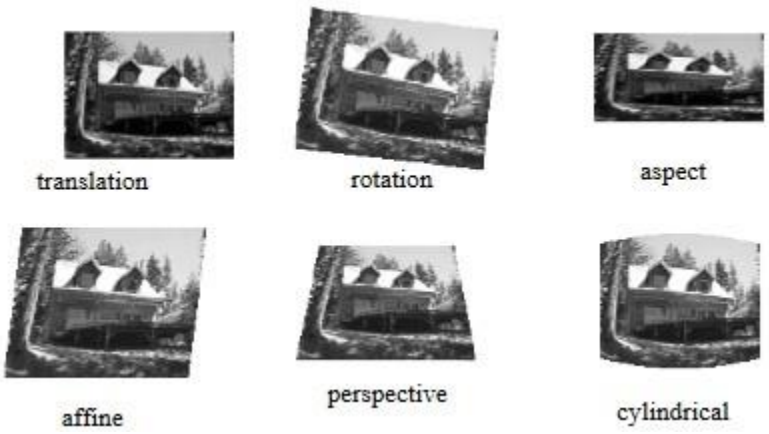

Figure 5 Parametric transformations examples

Forward warping is the process of computing the values of the pixels in the new image $g(x)$ given the source image $f(x)$ and the transformation $x' = h(x)$. There are some drawbacks such as undefined behavior when x' is not an integer or when the new position is in "between" two or more pixels. In this case, we can use the so-called splatting technique, which distributes the value of the pixel among the nearest neighbors in a weighted fashion, however, suffering from aliasing and loss of high-resolution details.

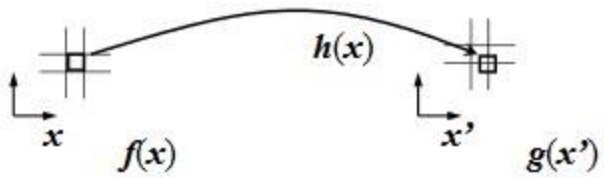

Figure 6 Forward warping detail of source and destination locations

Another solution is to use the inverse warping process. In this second approach, we formulate the problem as that of resampling, from destination pixels x' to source pixels x, the source image $f(x)$ given the mapping $x' = \hat{h}(x)$, where $\hat{h}(x)$ is the inverse of $h(x)$. By means of this second process, we can make sure that the new image hasn't any empty pixels, because we fill them one by one, which is another major issue in the forward warping process.

To compute the value of $f(x)$ at a non-integer location x, a FIR resampling filter is applied

$$g(x,y) = \sum_{k,l} f(k,l)h(x-k, y-l)$$

where (x, y) are the sub-pixel coordinates and $h(x, y)$ is some interpolating and smoothing kernel.

Figure 7 Left: Perspective warp, Right: Cylindrical warp

The get simpler computations, a two-dimensional to one-dimensional domain scaling is applied

$$g(\boldsymbol{A}x) \Leftrightarrow |\boldsymbol{A}|^{-1} G(\boldsymbol{A}^{-T}\boldsymbol{f})$$

for which, in the case of perspective transformations, the matrix $\boldsymbol{A}$ is the linearized derivative of the perspective transformation, which is the local affine approximation to the stretching induced by the projection. The transform $\boldsymbol{A}^{-T}$, whose frequency response is the projection of the final desired spectrum, is used as a pre-filter to the image $f(x)$

Feature detection

As said before, three detection algorithms were considered: SIFT, SURF and ORB.

SIFT

Scale-Invariant Feature Transform is a feature detection algorithm in computer vision to detect and describe local features in images. It was published by David Lowe in 1999.
In general, SIFT algorithm can be decomposed into four steps:

1. Feature point (also called key points) detection
2. Feature point localization
3. Orientation assignment
4. Feature descriptor generation

First step is to estimate a scale space extremum using the Difference of Gaussian (DoG). Secondly, a key point localization where the key point candidates are localized and refined by eliminating the low contrast points. Thirdly, a key point orientation assignment based on local image gradient and lastly a descriptor generator to compute the local image descriptor for each key point based on image gradient magnitude and orientation.

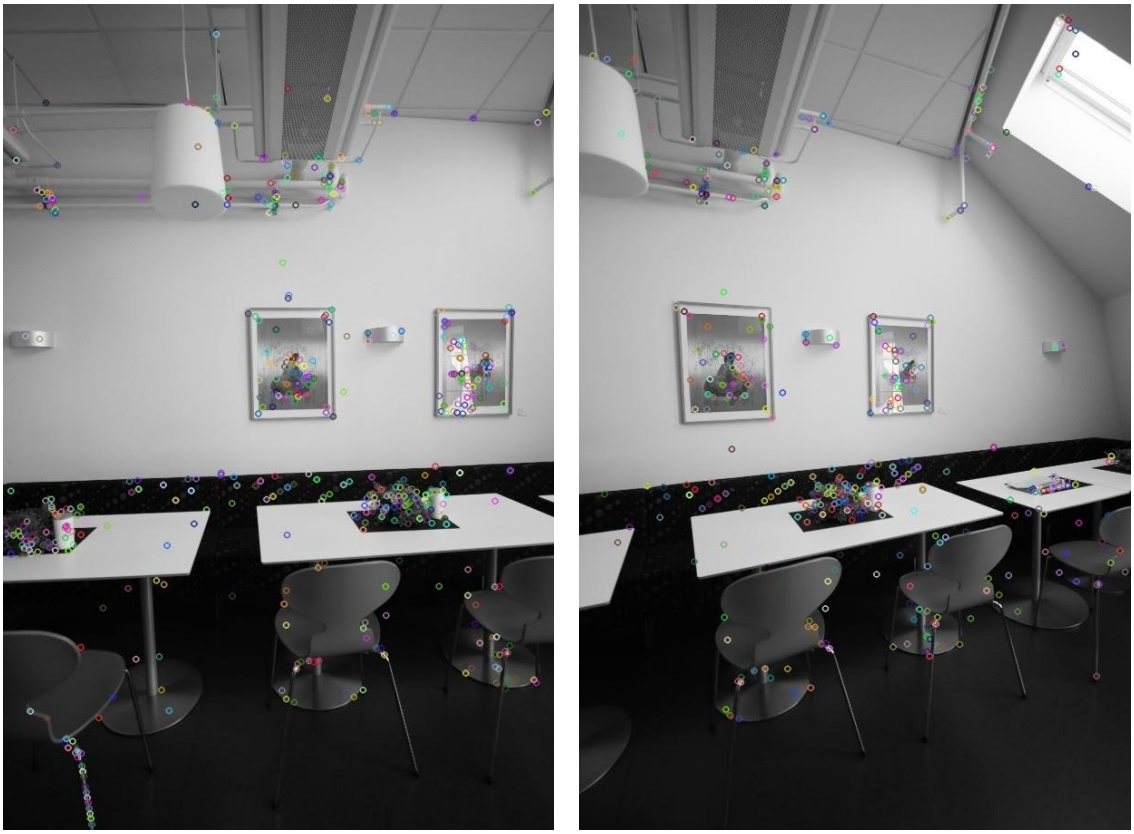

Figure 8 Result from SIFT, descriptor 1 - 612 points, descriptor 2 - 470 points, matches - 256 points

SURF

Speeded Up Robust Features is a local feature detector and descriptor. It was partially inspired by the scale-invariant feature transform (SIFT) descriptor. The standard version of SURF is several times faster than SIFT and more robust against different image transformations than SIFT. SURF approximates the DoG with box filters. Instead of Gaussian averaging the image, squares are used for approximation since the convolution with square is much faster if the integral image is used. Also, this can be done in parallel for different scales. The SURF uses a BLOB detector which is based on the Hessian matrix to find the points of interest. For orientation assignment, it uses wavelet responses
in both horizontal and vertical directions by applying adequate Gaussian weights. For feature description also SURF uses the wavelet responses. A neighborhood around the key point is selected and divided into subregions and then for each subregion the wavelet responses are taken and represented to get SURF feature descriptor.
The sign of Laplacian which is already computed in the detection is used for underlying interest points. The sign of the Laplacian distinguishes bright blobs on dark backgrounds from the reverse case. In case of matching the features are compared only if they have same type of contrast (based on sign) which allows faster matching.

Figure 9 Result from SURF, descriptor 1 - 1666 points, descriptor 2 - 1955 points, matches - 540 points

ORB

ORB is a fusion of the FAST key point detector and BRIEF descriptor with some modifications. Initially to determine the key points, it uses FAST. Then a Harris corner measure is applied to find top N points. FAST does not compute the orientation and is rotation variant. It computes the intensity weighted centroid of the patch with located corner at center. The direction of the vector from this corner point to centroid gives the orientation. Moments are computed to improve the rotation invariance.
The descriptor BRIEF poorly performs if there is an in-plane rotation. In ORB, a rotation matrix is computed using the orientation of patch and then the BRIEF descriptors are steered according to the orientation.

Figure 10 Result from ORB, descriptor 1 - 500 points, descriptor 2 - 500 points, matches - 100 points

We decided to choose only one of the three implementations in order to better compare the results of the following experiments. We decided to use the SURF implementation, not only because, on the paper, it is faster but also because, on average, we got better (at least aesthetically) results with it.
As was mentioned before, we preferred to use Speeded Up Robust Features (SURF) as our feature detector and descriptor. The main problem was deciding between SIFT and SURF,

since they are the most used and more reference/documentation can be found about them compared to ORB or other algorithms.
The SURF implementation is faster, even if in our case the difference is so small that we wouldn't even notice. However, at first sight, as appears from the pictures, SURF comes into help when we are dealing with the seams of stitching images. While in the case of just two pictures, there is not much difference as it is quite a simple stitching process, when we deal with 3 or more pictures the difference starts to appear clearer. In fact, as we are using a quite simple and not refined or optimized algorithm, the presence of seams when distortion happens, is pretty evident.

Figure 11 Left: SIFT, Right: SURF

Figure 12 Left: SIFT, Right: SURF

Moreover, SURF is considered generally better when dealing with rotation invariant, blur and warp transform than SIFT, which is better in different scale images (Darshana & Banerjee, 2017). Since we are given the image scale as fixed and equal among the images, our choice seemed reasonable.

Pixel selection and weighting

Creating clean, pleasing-looking panoramas involves both deciding which pixels to useand how to weight or blend them. We first discuss spatially varying weighting, pixel selection and then more sophisticated blending.
The simplest way to create a final composite is to simply take an average value at each pixel. Simple averaging usually does not work very well, since exposure differences, misregistrations, and scene movement are all very visible.
One way is to use a mask, as for alpha blending, to sooth the two images where they overlap.

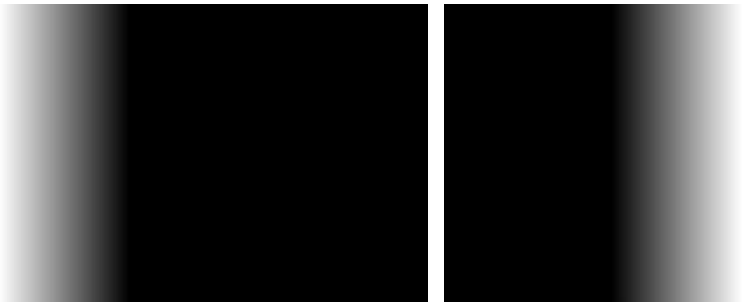

Figure 13 Example of left and right mask

Figure 14 Example of input images with left and right masks applied

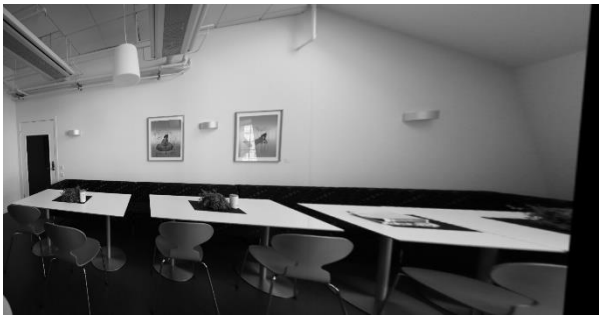

Figure 15 Output image

When considering this approach, we need to choose a proper smoothing window size, in order to get a good-looking output. https://angrytools.com/gradient/image/ was used to create the masks.

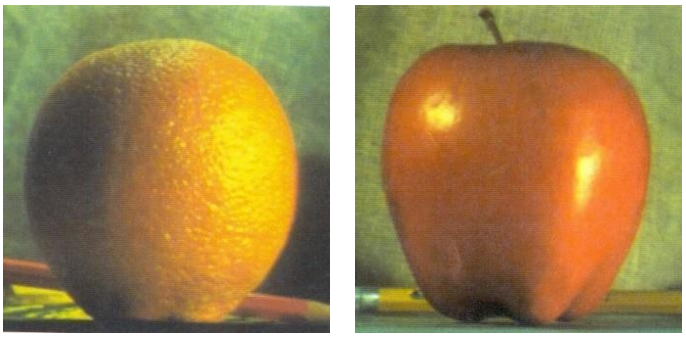

Figure 16 Input images

Figure 17 Left: window size of 10%, Right: window size of 50%

While it may seem from this example that the bigger window size the batter result we obtain, it is mainly due to the fact that the input images share the same background and only few details are different. In fact, in the next example, we see how having a too big smoothing window leads to ghost effects or obvious exposure issues.

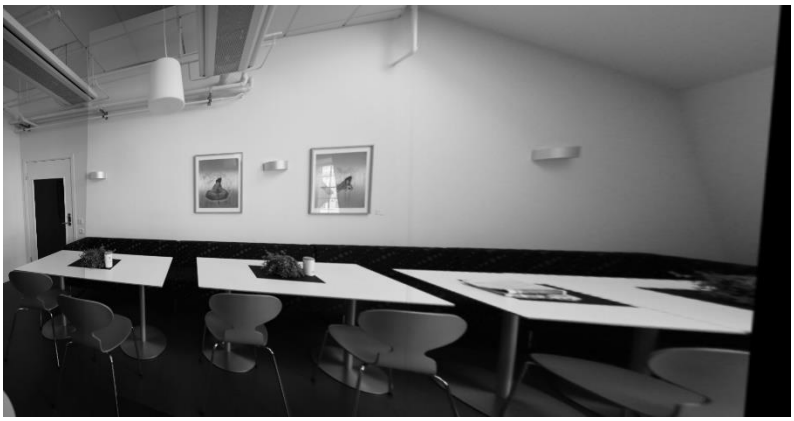

Figure 18 Example using an 80% window size, is clear the overlap issue on the left side

The optimal window size should respect the following two criteria (Efros, blending.pdf, 2010)

1. Window = size of the largest prominent feature
2. Window <= 2 * size of the smallest prominent feature

A better approach would be to place the seams in regions where the images agree, so that transitions from one source to another are not visible. In this way, the algorithm avoids "cutting through" moving objects where a seam would look unnatural.

Image blending

Image blending is the process of creating a smoother transition, than just pasted collage, from one image to another. Our implementation idea mostly refer to (Efros, Pyramids.pdf, 2005), (Efros , i0zbj8uvcao7lp, 2015) and (Pulli, lecture-5-stitching-blending.pdf, 2015). Before going into each blending technique, we tried a simple blending technique by averaging each pixel in an overlap part.

Figure 19 Example of blending by averaging each pixel in an overlap region

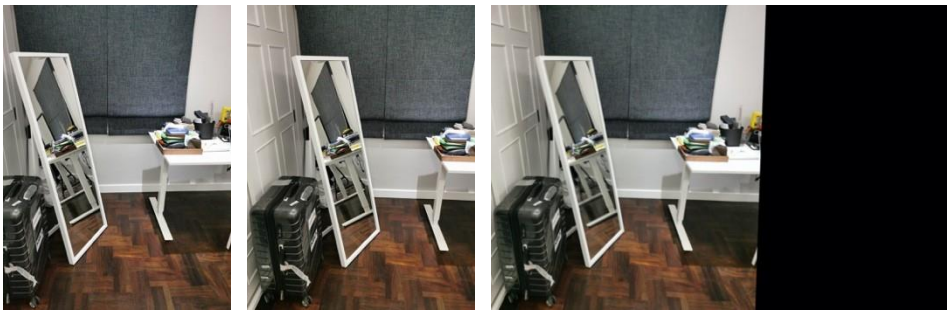

Figure 20 Example of blending by averaging each pixel in an overlap region (2)

However, this result has an obvious artifact along the stitching edge on Figure0-7. This effect is likely to be caused by a different exposure in the input images. On the other hand, in Figure0-8 which input images have same exposure level, this artifact does not appear. This algorithm performs well.

Then we tested three blending algorithms, alpha blending, alpha blending with feathering and Laplacian pyramid blending. We then applied two algorithms which are alpha blending with feathering and Laplacian pyramid blending to panoramic images.

Alpha Blending

Alpha blending is the process of combining a translucent foreground colour with a background colour, thereby producing a new colour blended between the two. We first implement the alpha blending with a mask as shown in (Efros , i0zbj8uvcao7lp, 2015).

This is the result of our implementation

Figure 21 Example of photomontage style implementation

Here, we successfully blend the doll part (foreground) to the new background.
Then, we also implement the alpha blending with feathering according to (Efros , i0zbj8uvcao7lp, 2015), using the following formula to calculate the output

$$I_{blend} = \alpha I_{left} + (1 - \alpha) I_{right}$$

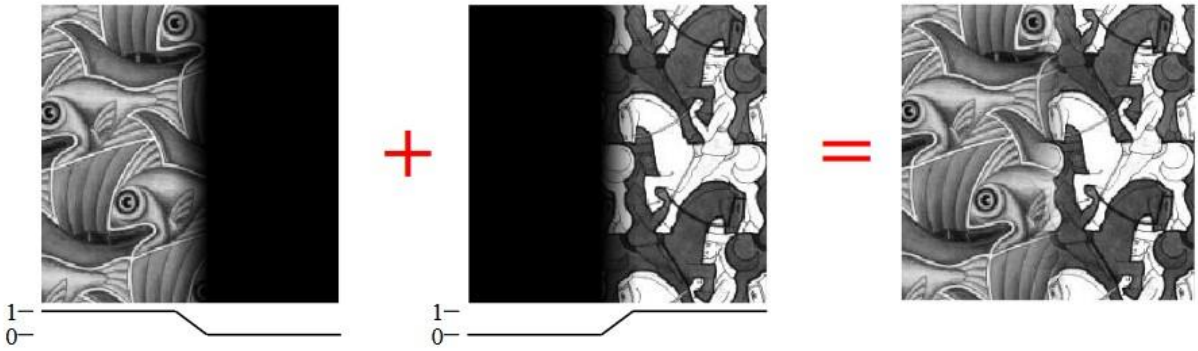

Figure 22 Feathering / linear blending example

Finally, we apply this to create a panoramic image (using SURF as described in the feature detection section). The following is our original implementation

Figure 23 Original set of images (1)

Figure 24 Linear blending left + right

Figure 25 Linear blending output (1)

Figure 26 Original set of images (2)

Figure 27 Linear blending output (2)

From the Original image (1) set we can see that the two input pictures are slightly different, more specifically, the position of the face is different, once looking on the right side and once looking straight to the camera. Therefore, this results in a ghosting effect in the Panorama image (1).

Figure 28 Particular from output image (1)

We archive a better result in Panorama image(2) as there weren't significant moving object in the scene. However, we can still see some blurriness in the area of three's leaves. Also, our algorithm only adjusts the ratio of the right picture in the stitching process, so we can see more distortion on the right side of the picture (as also mentioned before).

Laplacian pyramid blending

This is another blending method, which consists in splitting images into different channels, then separately blend each channel. We follow (Rubshtein, 2013) for the basic implementation.
Below is our original implementation

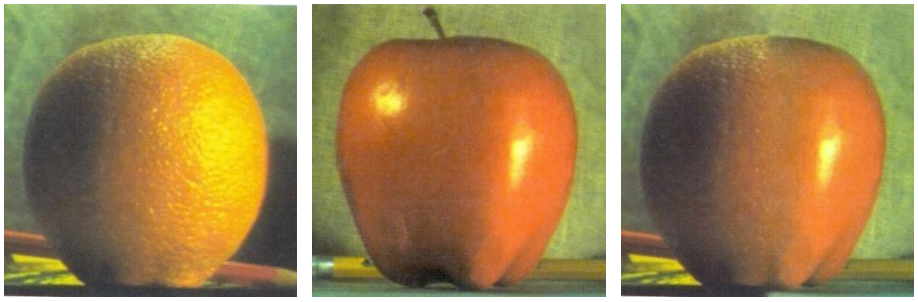

Figure 29 Laplacian pyramid blending example with an orange and an apple

Then we implement this technique on the Original image (1) set instead of Alpha blending technique. This new pyramid blending is also aiming to solve the ghosting effect we found in Panorama image (1).

Figure 30 Laplacian pyramid blending on Original image (1) set

However, even we can fix the ghosting effect, the edge of the original image is obvious. We later discover that this is a result of the masking problem. For the mask, we use the exact same size of the input image. Due to the limited time, we not yet find a sophisticate solution to this. However, our solution to this is to move the position of the mask to the right and down by a small distance to create an overlap between the mask and input image.

Figure 31 Laplacian pyramid blending on Original image (1) set after the mask trick is applied

With this solution, we get a better blending vertical and top edge. Leaving only the bottom edge to be obvious.

Conclusion

In conclusion, we decide to use SURF method as it gives us the best result aesthetically and faster than SIFT and ORB. Then we applied blending techniques to the picture. We found that the alpha blending gives us the most satisfying result under the condition that there are no moving objects in the scene. Unfortunately, we don't have a good tripod to experiment with, but we think that if all input images are shot with a quality tripod, the result of Alpha blending will likely to be better than Pyramid blending. However, when there is some motion in the scene the Laplacian pyramid blending can get rid of the ghosting effect. Note again that there still some flaws in our pyramid blending implementation which need to be fixed in order to receive a better result. Below we propose some potential future improvement that could enhance the quality of the panorama picture.

Improvements

In this section, we enclose the possible improvements to our algorithm.
In general, many approaches to image stitching exist, for example Poisson blending would certainly give great results in terms of good-looking outputs. We don't study this approach here, as we are not just looking for a perfect output but mainly looking to understand the mechanism of image stitching. However a reference can be found at (Pulli, /stitching-and-blending.pdf, 2014), (Abdullah & Agha) and (Anat & Assaf, 2004). We briefly introduce this approach, which works by copying the gradient field from the input image and reconstructing the final image by solving a Poisson equation.

Figure 32 Poisson blending example

The following are ideas that we found interesting but for various reasons (time, knowledge, complexity, etc.) we couldn't implement in this particular project, but were tested on their own to better understand the potential they have.

Some of them were already tested by others in image stitching algorithms but they require a different approach to the one we already have taken.

Seam carving

Seams for image editing are widely used. Many researches from (Agarwala, et al., 2004) with the interactive digital photomontage which uses seams to find the perfect boundary for multiple image composition, going through (Jia, Sun, Tang, & Shum, 2006) with drag-and-drop pasting, which extends the Poisson editing technique computing an optimal seam between source and target images, and then to (Wang & Cohen, 2007) who proposed a simultaneous matte and compose solution. This solution allows to scale the size of the foreground object and past it to the background for seamless stitching of images. In this area, seam carving is an important and often used algorithm for development. It is a content-aware image resizing algorithm and functions by establishing a number of *seams* (paths of least importance) in an image and automatically removes seams to reduce image size or inserts seams to extend it.

Figure 33 Expansion using 30 seams, Left: original image (500x250px), Right: Output expanded image (530x250px)

Figure 34 30 seams found for the expansion of the image

Seam carving also allows manually defining areas in which pixels may not be modified and features the ability to remove whole objects from photographs.

Figure 35 Left: original image (500x250px), Right: Output image (500x250px), one chair disappeared

As it is a simple implementation, it is prone to errors, especially when the object that we want to remove lies on a complex background or when the background is too simple (as a plain color) as the algorithm is trying to transform and warp the edges to cover the space left by the vanishing object and it ends up either messing the whole picture or leaving it untouched.

Figure 36 Left: original image (256x256px), Right: Output image (256x256px), a distorted part of the stone appears

This implementation can be exported, as it was tried by (Jia, Sun, Tang, & Shum, 2006) or (Wang & Cohen, 2007) into the image stitching algorithm in order to better capture the seaming points from which the overall blending quality would be favored by.

Parallax removal

When the images composing the panorama seems not perfectly aligned, i.e. blurry or ghosted in some places, it can be caused by many factors such as small motions (trees in the Original images (2) set), large-scale scene motions (face in the Original images (1) set) or parallax (failure to rotate the camera around its optical center).
The latter can be handled by replacing the projection equations for global alignment with equations used to model camera translations. The 3D positions of the matched feature points and cameras can then be simultaneously recovered, although this can be significantly more expensive than parallax-free image registration. Once the 3D structure has been recovered, the scene could (in theory) be projected to a single (central) viewpoint that contains no parallax.
When there is a large-scale scene motion a solution is to simply select pixels from only one image at a time as the source for the final composite.

However, when the motion is reasonably small, general 2D motion estimation can be used to perform an appropriate correction before blending using a process called local alignment. An alternative approach to motion-based de-ghosting is to estimate dense optical flow between each input image and a central reference image. (Equinor, 2018).

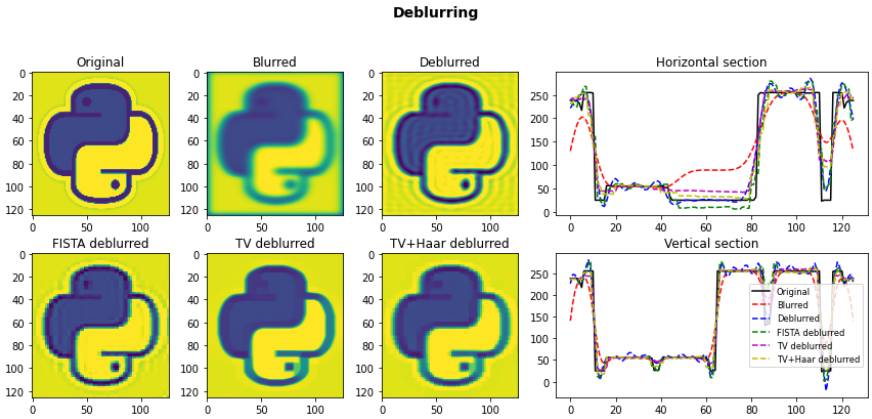

Figure 37 Deblurring example

References

Abdullah, A., & Agha, Z. (n.d.). *Efficient Poisson Blending for Seamless Image Stitching.* Retrieved from /pdfs.semanticscholar.org: https://pdfs.semanticscholar.org/9366/68c90bc5439fe42cfeb00000d790c984a41e.pdf

Agarwala, A., Dontcheva, M., Agrawala, M., Drucker, S. M., Colburn, A., Curless, B., . . . Cohen, M. F. (2004). Interactive digital photomontage. *SIGGRAPH.*

Anat, L., & Assaf, Z. (2004). *eccv04-blending.pdf.* Retrieved from webee.technion.ac.il: https://webee.technion.ac.il/people/anat.levin/papers/eccv04-blending.pdf

Darshana, M., & Banerjee, A. (2017). Comparison of Feature Detection and Matching Approaches: SIFT and SURF. *Global Research and Development Journal for Engineering*.

Efros , A. (2015). *i0zbj8uvcao7lp*. Retrieved from piazza.com: https://piazza.com/class_profile/get_resource/hz5ykuetdmr53k/i0zbj8uvcao7lp

Efros, A. (2005). *Pyramids.pdf*. Retrieved from graphics.cs.cmu.edu: http://graphics.cs.cmu.edu/courses/15-463/2005_fall/www/Lectures/Pyramids.pdf

Efros, A. (2010). *blending.pdf*. Retrieved from graphics.cs.cmu.edu: http://graphics.cs.cmu.edu/courses/15-463/2010_spring/Lectures/blending.pdf

Equinor. (2018). *pylops*. Retrieved from github.com: https://github.com/equinor/pylops

Jia, J., Sun, J., Tang, C.-K., & Shum, H.-y. (2006). Drag-and-drop pasting. *SIGGRAPH.*

Pulli, K. (2014). */stitching-and-blending.pdf*. Retrieved from web.stanford.edu: http://web.stanford.edu/class/cs231m/spring-2014/docs/stitching-and-blending.pdf

Pulli, K. (2015). *lecture-5-stitching-blending.pdf.* Retrieved from web.stanford.edu: https://web.stanford.edu/class/cs231m/lectures/lecture-5-stitching-blending.pdf

Rubshtein, A. (2013). *image-blending-using-pyramid*. Retrieved from compvisionlab.wordpress.com: https://compvisionlab.wordpress.com/2013/05/13/image-blending-using-pyramid/

Wang, J., & Cohen, M. (2007). Simultaneous Matting and Compositing. *CVPR.*