Universiteit Leiden
The Netherlands

Opleiding Informatica

A study of different approaches for improving the stitching of spherical panoramas

Ruben van der Waal

Supervisors:
Dr. Michael S. Lew & Dr. Erwin M. Bakker

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

15/08/2017

Abstract

We introduce and compare different strategies to stitching spherical panoramas. Three different techniques are explored, namely blurring, blending and dynamic stitching. These techniques perform differently in our tests and their advantages and disadvantages are discussed. The source images are taken with the Samsung Gear 360, an all-in-one spherical panoramic camera. The software included with the device performs poorly in some cases. Problems exist with alignment, ghosting and content cropped in the result. The proposed techniques are directly compared with output of Samsung's software. Preliminary results show promising improvement especially on more complex, text-containing input.

Acknowledgements

In the first place I would like to express my appreciation to my supervisor Dr. Michael S. Lew for his continued feedback, time and for providing the device for me during the entirety of the project.

I would also like to thank my parents for their support and encouragement throughout the project.

Contents

Chapter 1

Introduction

The process of stitching two images together is complicated with many options along the way. Since many differences can exist making the transition harder to hide when photos are taken from different positions, at different times or with different exposures. The process of stitching spherical panoramas introduces some unique problems as well, such as the fish-eye input and having to stitch an image to the same one twice on both ends.

This bachelor thesis is made under supervision of Dr. Michael S. Lew and Erwin M. Bakker of the Leiden Institute of Advanced Computer Science (LIACS).

For this paper the object of interest is the Samsung Gear 360. A 360 degree high-resolution panorama camera marketed for the general public. The device consists of two 180+ degree camera's. The camera's are aimed away from each other to create a full 360 degree panorama. At 4 megapixel each the resulting image is around 8 megapixel in size. See Figure 1.1

Figure 1.1: The Samsung Gear 360

However a lot of critique arose from costumers when they saw the stitching results from the included software. The transition between the two images is sometimes very visible, the alignment is not done well and worst of all content is lost on the transition. While all these problems are avoidable since both cameras create sufficient overlap for a correct stitch.

As approach for the project a new implementation from the ground up seemed to be the best course of action. The option of correcting the results from Samsung was there, but one of the problems we wanted to improve upon was fixing the missing content at the borders. This is not possible with just the output from Samsung.

Figure 1.2: Example output from Samsung

Note how the writing on the monitor in Figure 1.2 is supposed to spell out “Sample Writing”, yet some letters are missing. This scene is the main example in the paper because it contains various different transition types. There is the wooden desk area with a very simple texture, various straight lines to check alignment and the more complex writing on the screen.

This thesis will start off with a survey of the current state of the technology on this subject and will also further identify the problems of the process. The rest of the thesis consists of the different parts of the stitching process with for each stage a dedicated chapter. Results of each stage are given at the end of the chapters and afterwards compared against Samsung’s results in a separate chapter.

Chapter 2

Survey

The goal of this survey is to review the state-of-the-art stitching techniques and the different steps leading up to this process, categorize them on the different stages of the stitching process and identify problems.
Many difficulties are connected to the stitching process. Issues can arise due to, but are are not limited to, lens faults, differences or changes in illumination, parallax, complex scenery near borders, constraints on processing time and noise in the image. Spherical panoramas add more difficulty in terms of correcting the fish-eye distortion, have less lean way in terms of positional error and have a more severe vignetting effect.

Image stitching finds many uses in today's technology:

- Efficient panorama creation for mobile devices.
- Various street view services use many spherical panoramas to map out cities.
- Insertion of objects into images, for example sky replacement in scenic images.
- Texture synthesis for various uses.

In the work of Chu et al [LCC+15] six major stages are defined in the stitching process. See Figure 2.1.

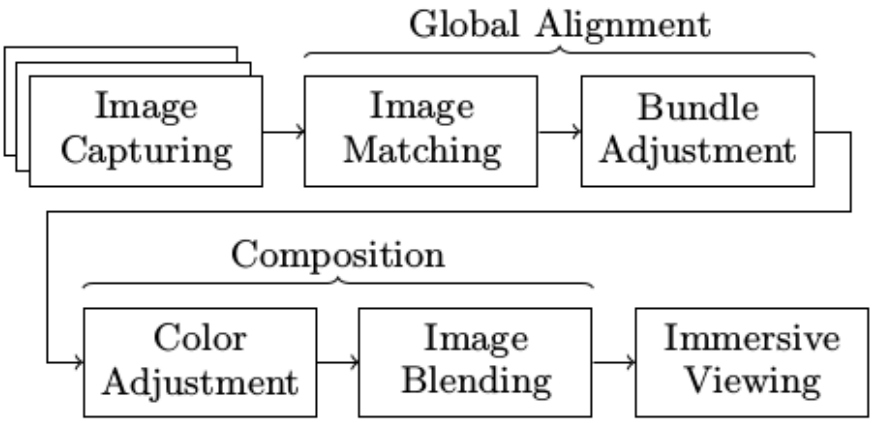


Figure 2.1: Pipeline of panorama stitching

In the context of this project the stages refer to the following:

- **Image Capturing:** Image Capturing is executed with the Samsung Gear 360. The raw data is obtained from the memory card in JPG-format. Images are roughly 8000 by 4000 pixels in dimension.The image content is in fish-eye format.
- **Image Matching:** The process of positioning multiple image in the correct arrangement. A trivial step in the case of this project since there are only two images and it is known that they need to be positioned side by side. Stitching one side before the other is also trivial.
- **Bundle Adjustment:** Camera angles on objects in the picture can differ due to the photos not being captured from the exact same point. This is called parallax. Bundle adjustment is applying a transformation on the image to correct this.
- **Color Adjustment:** Among color adjustment belongs equalizing the contrast between both images. Lighting can be different for both cameras creating a contrast between the images. Secondly fish-eye lenses are notorious for creating a so-called vignetting effect in their pictures. The vignetting effect is an effect where the outer parts of the image appear to be darker than the inside. This is especially noticeable when viewing panoramas as a whole as opposed to zoomed in.
- **Image Blending:** Now that the images are correctly arranged, aligned and corrected they still need to be joined together in a way the transition between the two images is maximally invisible.
- **Immersive Viewing:** For spherical panoramas some tools exist to view the image as if projected on the inside of a globe. Images can also be viewed in virtual reality for an enhanced and more immersive viewing experience.

The papers in the survey will be categorized by these different stages, with emphasis on the stages focused on in this project. Since most papers do not specialize in either image matching or bundle adjustment these stages are grouped in this survey as Global Alignment.

2.1 Image Capturing

Chu et al [LCC+15] proposes an interactive method to capture panoramas on a mobile device. The user is guided through creating his panorama systematically using the device's gyroscope. The program lets the user line up the photo by alignment with white rectangles on the screen. These rectangles are stationary in the scene and turn orange upon successful capture of the image. This method ensures that all data is captured to create a spherical panorama. See Figure 2.2.

(a) User lines camera up with the white rectangle

(b) Image is captured

Figure 2.2: Example of panorama capturing process in Chu et al [LCC+15].

2.2 Global Alignment

In the work of Petkova and Nuri [PN11] a new method of feature point detection is proposed. The method divides images into a height map and detects planes. From these planes the contours are used as feature points for matching. See Figure 2.3. This method is faster than edge or corner detection but results in many more feature points, which slows down the matching process. To combat this a feature points are reduced, combined and grouped. Groups are then matched with their closest match in the corresponding image.

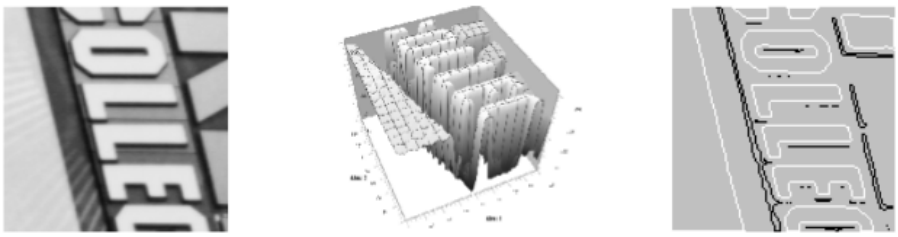


Figure 2.3: Left is the input image, Middle the 3D presentation, Right extracted Feature points that outline objects. [PN11]

In the work of Bian et al [BXXL12] an impressive algorithm is shown that propagates deformation near the stitching edge gradually throughout the image. To do this first feature descriptors are extracted using SURF. These points are then matched and used to calculate a deformation vector. For the matching points an eight dimensional vector is calculated. This deformation now needs to be smoothly spread across the image. Every deformation vector uses an edge in the image as its guide line. After applying all the deformations the image

can be reconstructed with use of the Poisson equation. Some results can be seen in Figure 2.4. Noticeable is how much care is put in the design of the algorithm to preserve edges.

(a) Image is matched wrong intentionally.

(b) Image stitching result using Deformation Propagation [BXXL12].

Figure 2.4: Deformation propagation example

Bundle adjustment is generally done only through matched feature points, but in the specific case that Pesti et al. [PEH+08] covers this is not enough to adjust the images. The algorithm deals with orthographic imagery which are images taken from above the surface of the earth straight down to make maps. These are used in popular applications such as Google Maps, MapQuest, Yahoo! Maps and Microsoft Virtual Earth. These maps need to be geographically correct. Popular applications use highly advanced planes and cameras to take the photos, minimizing the difficulty in the adjustment and stitching process. Pesti et al proposes to use user annotation to set reference points in the images and use these as anchor points.

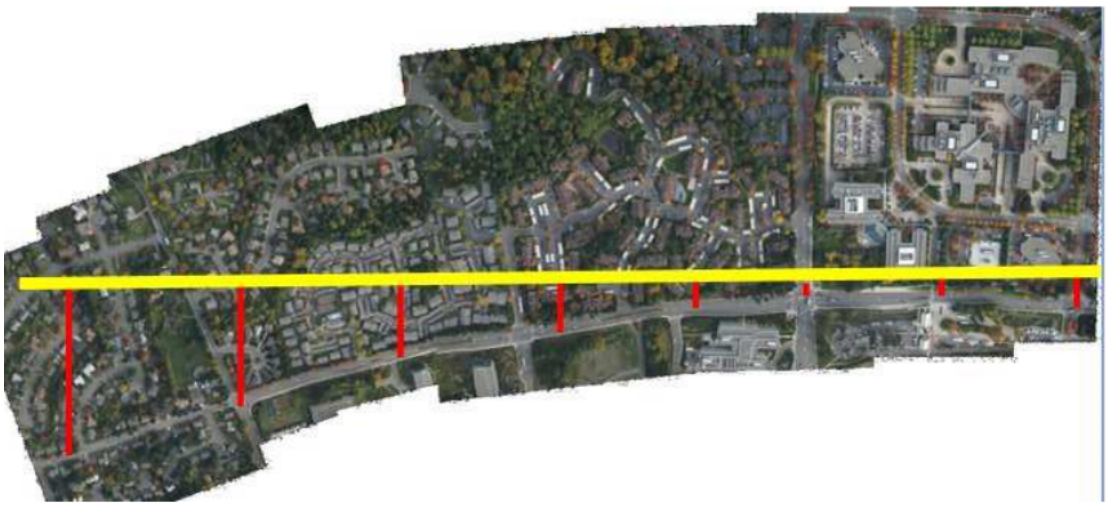


Figure 2.5: A straight road appears to be curving without geographical constraints.

In Figure 2.5 an example of the accumulating error can be seen. The images curve further and further away from their corresponding geographical position, even though the stitching is seamless. The proposed algorithm follows these steps:

- Initial estimates for camera model parameters are made for each camera in a "not estimated" state, that has sufficient ground reference pairs.
- Nonlinear optimization (bundle adjustment) is used to globally optimize the parameters of all cameras with estimates. Both the user-supplied ground reference pairs and constraints introduced by feature match pairs are used in this global optimization step.
- Synthetic ground reference pairs are temporarily created where two images overlap and at least one has a camera with a known model. These are used to initialize camera parameter estimates in future iteration steps.

This process is repeated until all images are corrected. The main contributions this research gives is that due to the robustness of the algorithm much less expensive hardware can be used. Therefore greatly increasing the amount of regions that can be mapped.

2.3 Color Adjustment

Multiple solutions to equalize color and luminance in panoramas exist. In the work of Pham and Pringle [PP95] color correction is performed using polynomial mapping functions. The overlapping area is used to establish a mapping function that maps corresponding pixels to each other. This achieves excellent results, however the prerequisite is that the alignment of the images is perfect. Otherwise the mapping is incorrect as well as the resulting function. In our case due to a relatively small overlap and high levels of parallax in situations where

objects are near the device such an approach is difficult.

A second option is to construct a mapping function from the color histograms in the overlap, as proposed by Zhang et al. [MJYD01] The color correction used in the source image is then also used in the adjacent image. This method does not rely on exact alignment of the images and one-to-one pixel correspondence. Therefore this method is a lot more robust, however the results are not as good as with the polynomial mapping.

Tian et al. [TGTC02] introduces another method that calculates the averages of each channel in the overlapping area and equalizes these to the corresponding channels of the adjacent image. This calculated is done in sRGB space. sRGB space is gamma-corrected RGB space. sRGB uses a non-linear value system to fit a higher range of colors. Therefore averaging out a channel in sRGB space does not correspond to the actual average as perceived by humans.

Various similar approaches have been proposed by scientists using different color spaces, examples are the work of Ha et al. [HKL+07] and Brown and Lowe [BL07]. Another method is to only consider certain channels, as in the work of Meunier and Borgmann [MB00].

The work of Xiong and Pulli [XP09] performs the operations in linearized RGB color space. A global adjustments is performed to minimize correction across images. A problem when correcting the color in images is that when images are corrected sequentially an error can accumulate. Doing the global adjustment helps smooth out the color across images. The algorithm is designed for mobile devices and is highly efficient. Results of the color correction can be seen in Figure 2.6

(a) original

(b) color corrected using Xoing and Pulli. [XP09]

Figure 2.6: Color correction

A second problem related to color adjustment is the vignetting effect, see Figure 2.7. This is generally caused by physical limitations of the lens. Especially fish-eye lenses are notorious for this.

Figure 2.7: Example of vignetting effect

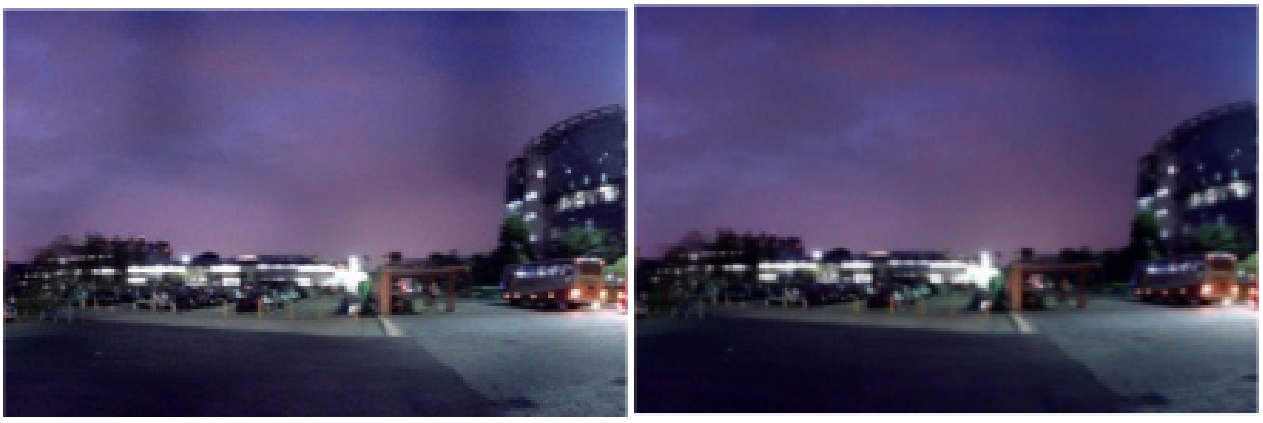
Chu et al [LCC+15] corrects the vignetting effect before applying global corrections. Luminance y in scene point X and irradiance E_X:

$$y = f(kE_X)$$

where k is the exposure value, is a composite of gain value, exposed time and aperture value. Function f is the camera response function. We now rewrite the irradiance E as the product of scene radiance L and the vignette correction function M:

$$y = f(kM(\theta)L_X)$$

Where $\theta = atan(x_p/f)$, x_p is the distance of the pixels to the center of the image. f is the focal length (in pixel unit). M is the vignette model: $M(\theta) = 1 + v(cos^4(\theta) - 1)$, where v is the vignette parameter to be estimated.



(a) No Vignette Correction

(b) With Vignette Correction

Figure 2.8: Result of Vignette Correction [LCC+15]

In Figure 2.8 can be seen that the vignette effect has been strongly reduced.

2.4 Image Blending

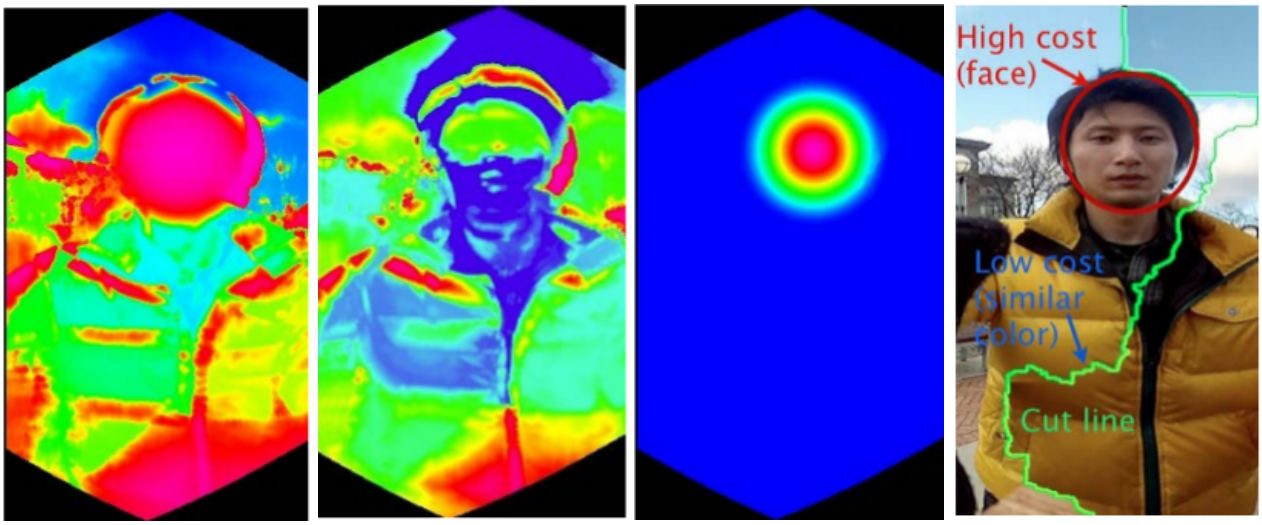
In the work of Ozawa et al [OKK12] the issue that when faces are near edges of the panorama source photos are distorted is addressed. The transition can cut through the faces. Especially when time parallax is present in the image this causes extreme distortion in faces. Time parallax is caused by the source photos of the panorama not being taken at the exact same time. Movement happens between the exposures and difference in the scene are introduced. A solution is proposed where facial recognition is incorporated in the stitching process. So besides color difference between corresponding pixels faces are also taken into account. The following formula is used to calculated the seam between the images:

$$C_t(i,j) = \alpha_c C_c(i,j) + \alpha_f(j)$$

Where i is the current pixel and j is a neighbouring pixel. $C_c(i,j)$ is the cost function for color continuity, $C_f(j)$ is the cost function for if a face is detected at the pixel j. α_c and α_f are weights.

$$C_c(i,j) = 1/N_c\{||A(i) - B(i)||_{L2} + ||A(j) - B(j)||_{L2}\}$$

Here A and B are two matched color vectors in the images, $A = [R_A, G_A, B_A]^T$ and $B = [R_B, G_B, B_B]^T$. The normalization constant N_c normalizes the value to lay between 0 and 1. The total set of edge transitions is considered and a shortest path is found using Dijkstra's algorithm.



(a) Total cost (b) Color cost (c) Face detected cost (d) Result

Figure 2.9: Different steps involving the algorithm of Ozawa et al [OKK12].

As can be observed in Figure 2.9 the added difficulty calculated by the face cost guides the stitching line away from the face so that it does not cut across it.

In the work of Greg Ward [Waro6] a technique is introduced that works well even when images are not perfectly aligned or with identical exposure and works in HDR (High Dynamic Range). The technique is based on two observations:

- Low frequencies may be blended without introducing visible artifacts.
- High frequency splices are interpreted as edges, but can be hidden by edges in the input.

The input is divided in low frequency bands and a single high frequency band. The low frequencies are blended together using a sinusoidal blend function. The high frequency band is spliced rather than blended. An edge is detected in the input and this is where the transition occurs.

(a) A simple splice (b) Photoshop^{tm} (c) Burt-Adelson method [BA83] (d) Greg Ward [Waro6].

Figure 2.10: Two HDR images stitched with different techniques.

In Figure 2.10d the result of a stitch performed with the proposed technique can be seen. A good result but upon closer inspection some errors in the rolling mist can be seen, namely creation of edges where they should not be as defined.

The work of Yan et al [YHLX13] details a method to create infinite panoramas of stereo images. Stereo imagery is a technique to encode 3D into a 2D image. Stereo images contain disparity, keeping this consistent between images and avoiding artifacts on seams is an extra challenge when stitching stereo images. The method contains three steps. First graph cuts are used to find an optimal seam to stitch a pair of stereo images. Second the disparity range of one stereo image is adjusted to maintain consistency. Lastly the two images are blended together using gradient domain optimization [PGB03]. Key improvements in this research are:

- The disparity-aware energy function used with the graph cut algorithm.
- A revised warping-based disparity scaling technique. This maintains consistency across images in disparity.

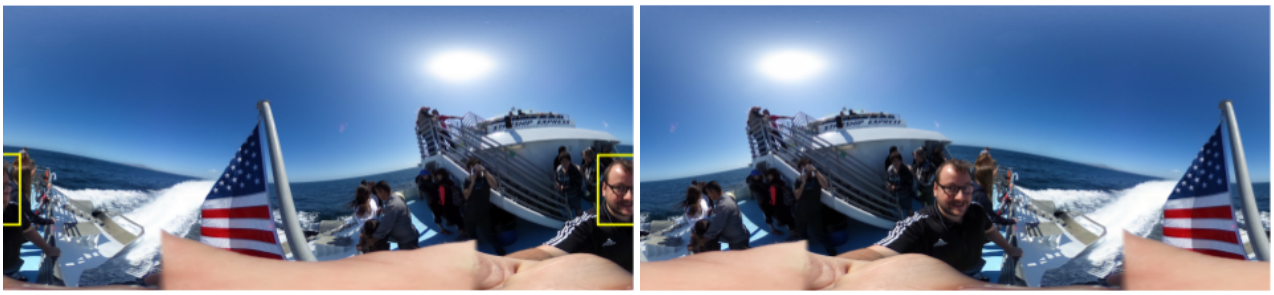
Figure 2.11: A panorama produced of five separate stereo images using Yan et al [YHLX13].

Also images that do not necessarily belong to the same scene can be stitched together creating infinitely wide

stereo panoramas.

2.5 Immersive Viewing

An interesting proposal was done by Pohl and Grau [PG16] to improve the viewing experience when viewing spherical panoramas in 2D. When viewing the image in 2D it is in the equirectangular format. The equirectangular format has the property it can be shifted around and no complex recalculations need to be done. Therefore Pohl and Grau propose to have the face of the photographer in the center, see Figure 2.12. The paper describes the face of the photographer as area of interest. Others area of interest could also be defined such as groups of faces or another object of importance. Allegedly the photographer is not always the center piece of the photo.



(a) Photographer's face detected in 2D spherical panorama (b) Panorama centered on photographer

Figure 2.12: Automatic shifting for spherical panoramas [PG16]

The work of Chu et al [LCC+15] also proposes a method of interactively viewing the panoramas. Pan360 projects the image onto a sphere and the user can navigate the viewpoint by motion sensor, touch gesture or default motion. The downward direction of the sphere is aligned with gravity and zooming is always possible with the pinch gesture. See Figure 2.13.

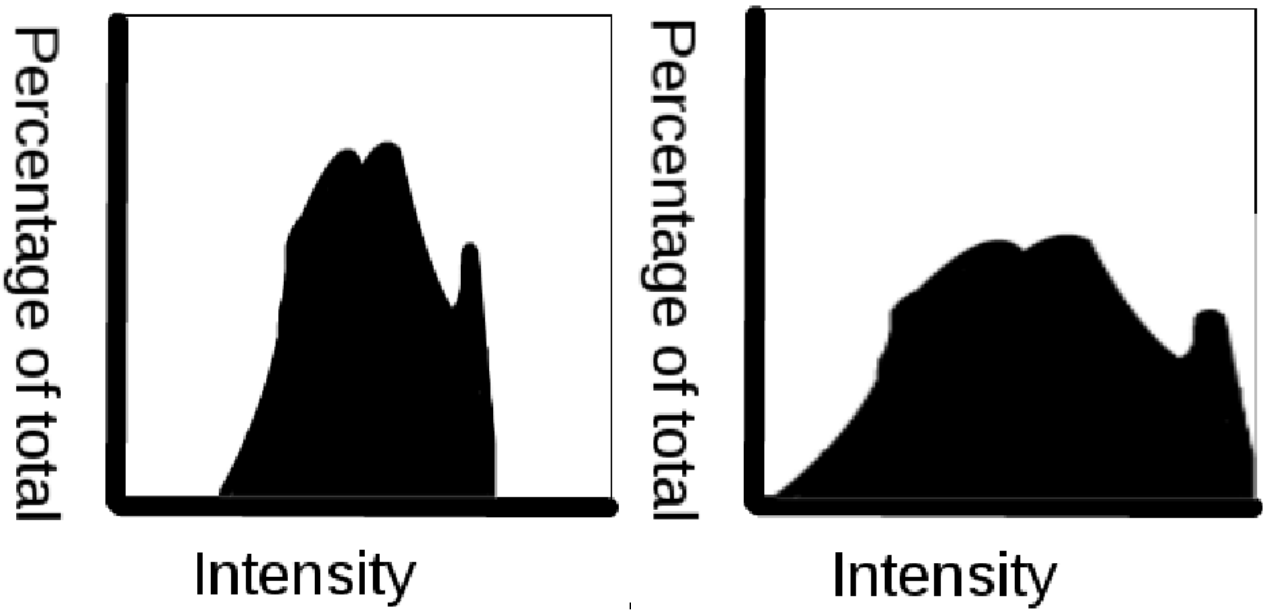
Figure 2.13: Illustration of the view direction controlled by INS [LCC+15].

Chapter 3

Color Contrast

One of first problems that can be observed in the raw photo data is that the contrast and brightness in the two images is not equal. Especially if one camera is aimed towards a light source and the other is not a big difference can be seen.

A technique that can be applied to combat this is called ”histogram equalization”, a standard feature of the OpenCV library. A map is created of how many pixel have which intensity. The resulting values are then smoothed out. A visualization of this process can be seen in Figure 3.1. Note how this is an approximation.



(a) Before equalization.

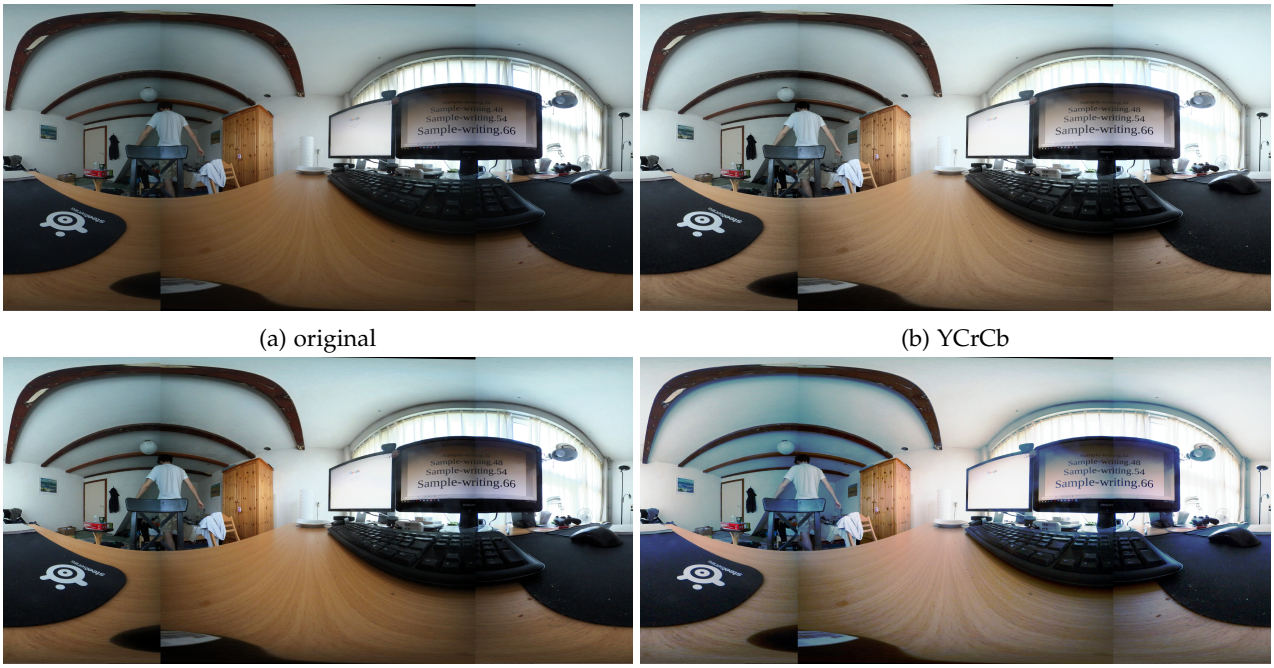
(b) After equalization.

Figure 3.1: Histogram equalization

However histogram equalization only works on a single channel. A solution could be to split the RGB channels and equalize each one and then merge them together again. This seems tempting at first but is based on a wrong interpretation of how the equalizing process works. Equalization should be performed on the intensity

values of the image, not the color components themselves. In other words we do not want to disturb the color balance in the image.

To still apply histogram equalization to a RGB image we need to first convert to a different color space. A color space that has separate channels for intensity values and color. Candidates are: YCbCr, HSV, HLS, YUV.



(a) original (b) YCrCb

(c) HSV (d) RGB

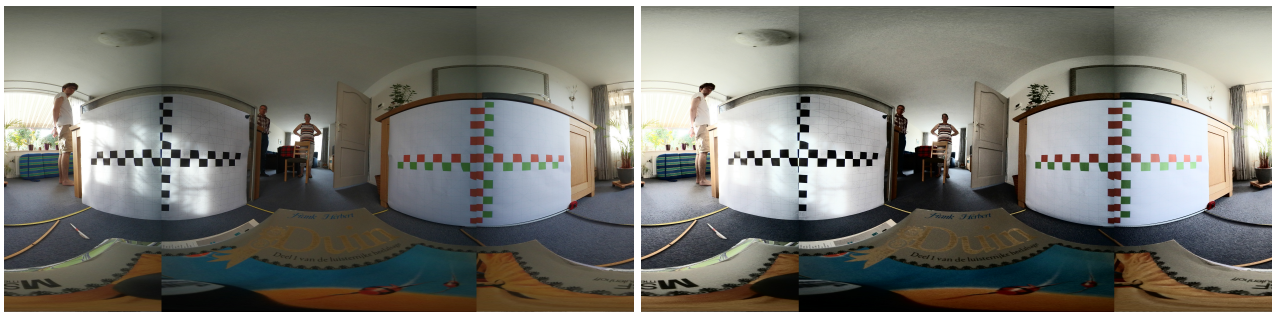
Figure 3.2: Color histogram equalization in different color spaces.

Figure 3.2b is the RGB image converted to YCrCb with the Y-channel equalized. Figure 3.2c is converted to HSV with the V-channel equalized. Figure 3.2d is the RGB image with all channels equalized.

Both the corrected YCrCb and HSV images show definite improvement in overall contrast and a bit more similarity between the different parts.

Interesting is that there is still some difference to be observed between the HSV and YCrCb histogram equalization. YCrCb has a more white-washed overall appearance and HSV retains richer colors. For comparison the RGB is included, clearly some of the colors in the image have been shifted with tones of blue and purple near the borders.

One of the issues with the contrast not being addressed by this operation is the so-called vignetting effect. In this case due to physical limitations with fish-eye lenses in general the edges of the image are darker than the middle. Histogram equalization does not effect this problem.



(a) HSV (b) YCrCb

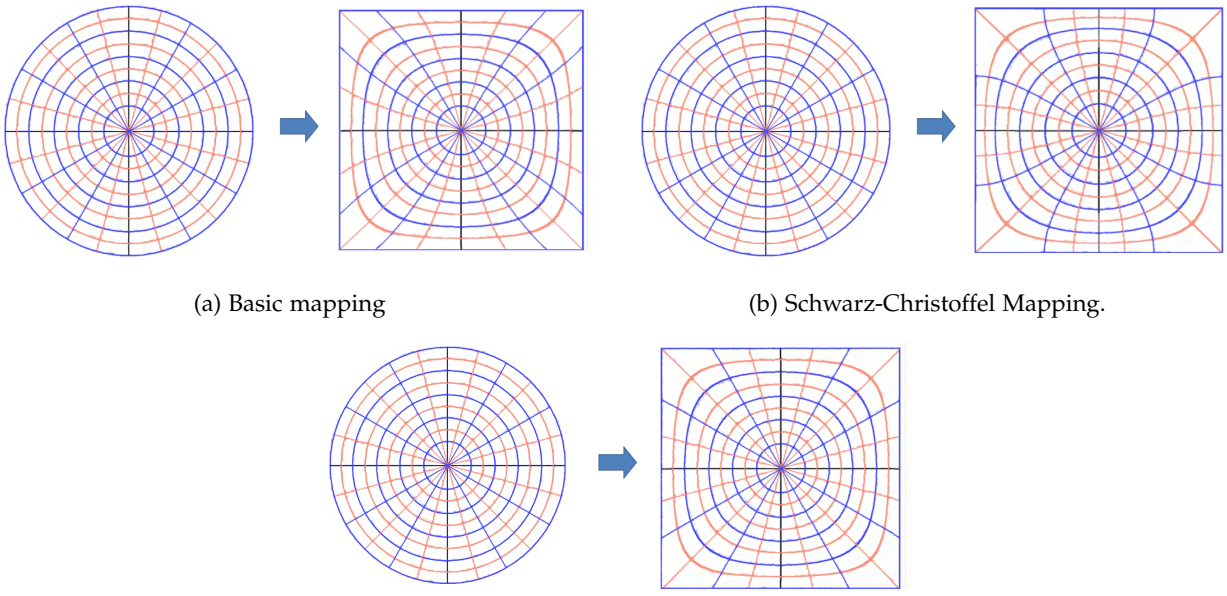
Figure 3.3: Different color spaces comparing noise amplification.

Another problem is that histogram equalization can amplify noise present in images, as seen in Figure 3.3b. With testing this seemed to be more of an issue with the YCrCb color space and histogram equalization. This deemed to be another reason to choose the HSV color space.

Chapter 4

Projections

The initial problem at hand is the input data itself. The device outputs two high resolution fish-eye images. However for convenience the stitching process requires a rectangular image. It is possible to perform calculations on the raw fish-eye image as seen in the work of Kazhdan and Hoppe [KH10].
There are many ways to project a circle to a square image. As can be seen in Figure 4.1 and the work of Fong [Fon15].



(a) Basic mapping

(b) Schwarz-Christoffel Mapping.

(c) Fernandez-Gausti Mapping

Figure 4.1: Display of different circle to square transformations. [Fon15]

While these transformations produce images that could in theory be stitched together, they can not easily be projected on a globe.
The natural pick for square to globe projections is the same one used for world maps, the equirectangular projection.

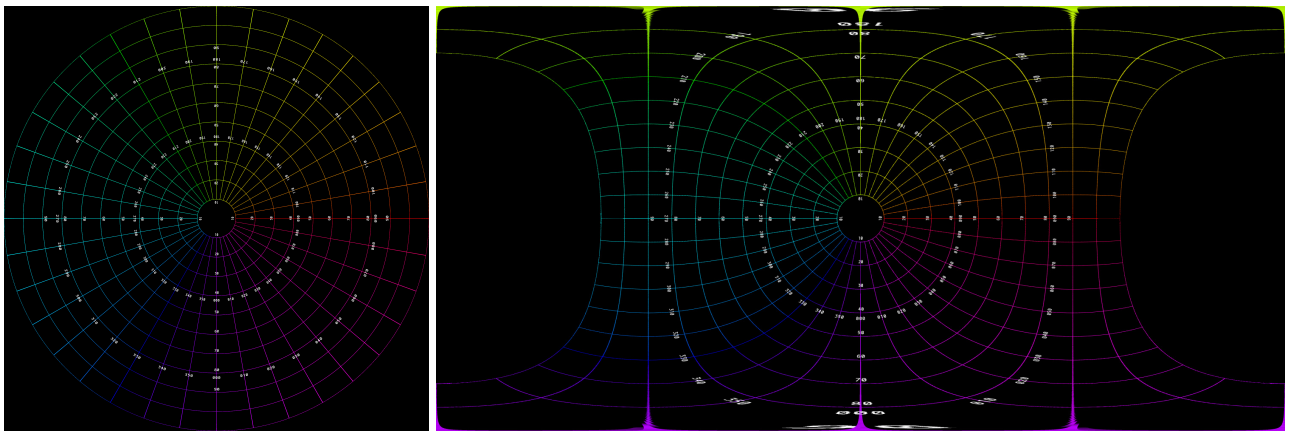


Figure 4.2: Equirectangular projection

To achieve this transformation the following conversions are done:

- 2D equirectangular to longitude/latitude

  $longitude = \pi * i$
  $latitude = \pi * j$

  Where i denotes the equirectangular x coordinate and j the y coordinate.

- longitude/latitude to 3D vector

  $P_x = cos(latitude) * cos(longitude)$
  $P_y = cos(latitude) * cos(longitude)$
  $P_z = sin(latitude)$

  Where P denotes the positional vector in 3D space.

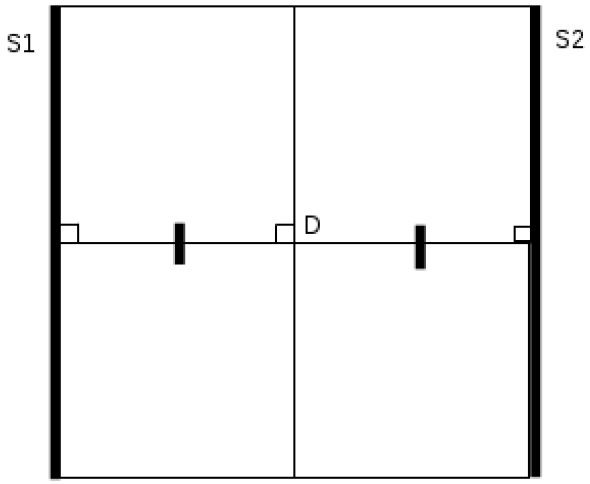
- 3D vector to 2D fish-eye

  $\theta = arctan(P_z, P_x)$
  $\phi = arctan(\sqrt{P_x{}^2 + P_z{}^2}, P_y)$
  $r = \phi / FOV$
  $x = r * cos(\theta)$
  $y = r * sin(\theta)$

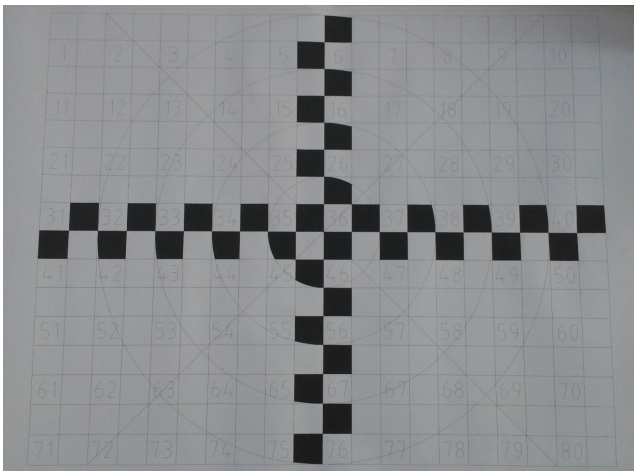
  Where the FOV (field of view) is the maximum angle the camera lens can observe in radials, r the distance from the center of the circle to the fish-eye coordinate and x and y are the fish-eye coordinates.

This conversion is between the unit circle and a 1:1 rectangle. In other words the coordinates are normalized. Therefore translation and scaling needs to be done before and after conversion.

We now still need to find the FOV of the cameras. The following test setup is used to acquire this. See Figure 4.3.



(a) Test setup layout. D is the position of the device, S1 is screen 1 and S2 is screen 2.



(b) Screen layout.

Figure 4.3: Test setup overview.

Every square in 4.3b is exactly 5cm by 5cm. The device is positioned perpendicular to both screens and in the middle, meaning the overlaps between the device's view are positioned aimed at the middle of both screens. The distance of the device to either screen is 50cm. See Figure 4.3a. The distance from the camera to both screens is equal. Also both screens are placed parallel to each other.

Figure 4.4: Test photo, one camera only

Due to the device being fairly compact it is difficult to aim the camera facing exactly perpendicular to the screen. Therefore we take the average of both overlaps to calculate the true FOV.
On the side of the black and white screen the overlap is roughly 2cm. On the side of the green and red screen the overlap is 6cm. With basic trigonometry we can now calculate the angle of the overlap.

$$tan(\alpha) = opposite\ edge / adjacent\ edge$$

$$\alpha_{rg} = atan(6/50)$$

$$\alpha_{rg} \approx 0.12\pi$$

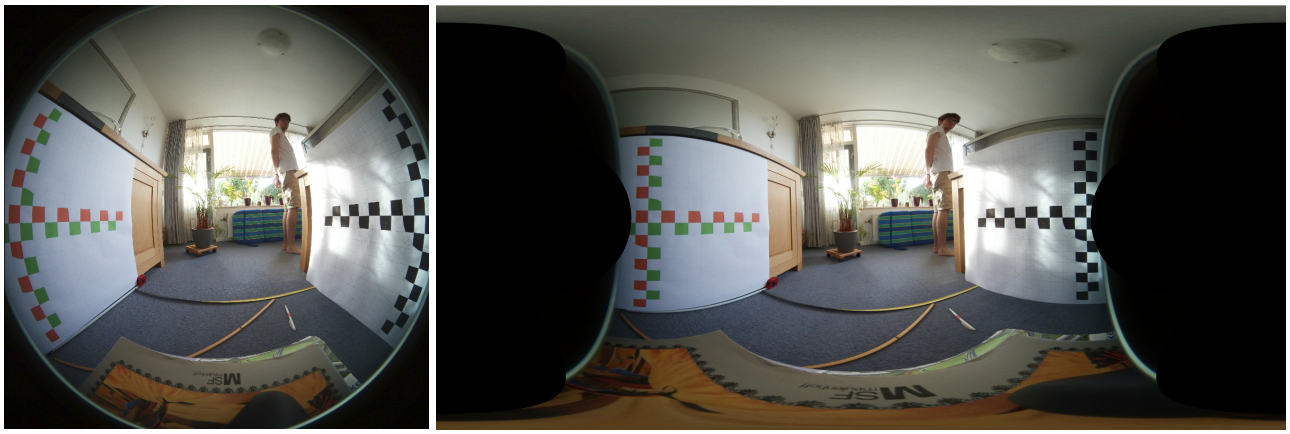
$$\alpha_{bw} = atan(2/50)$$

$$\alpha_{bw} \approx 0.04\pi$$

$$\alpha = (\alpha_{rg} + \alpha_{bw})/2$$

$$\alpha = (0.12\pi + 0.4\pi)/2 = 0.08\pi$$

Here α_{rg} is the angle by which the camera overlaps on the red-green screen and α_{bw} is the angle by which the camera overlaps on the black-white screen. We can conclude from this that the total FOV of one camera is 1.08π radians or 194°. Implementing this gives the following results. See Figure 4.5.



(a) Raw output of the device.

(b) Equirectangular transformation, FOV = 1.08π = 194°.

Figure 4.5: Equirectangular transformation.

As can be seen in Figure 4.5 lines that previously appeared to curve are now straightened out. The image has the correct proportions.

Chapter 5

Manual Positioning

The next step in the process is positioning the image so that every corresponding point aligns optimally. At first an attempt was made to implement automatic positioning, this was discontinued due to a lack of time and the focus in the project shifted to the stitching process.

Figure 5.1: A possible rotation

In Figure 5.1 an example orientation is shown, the orientation is obviously not correct. At this stage of the process the user is asked to give their input for the optimal positioning. Using the keyboard keys both images can be manipulated independently. The degrees of freedom for each image half are:

- Translation over the y-axis.
- Cropping the image on the right side.

- Rotating the image.
- Applying a trapezoid transformation see Figure 5.2.

All transformations are done in real time. In order to achieve this the the original image is down-scaled to one fourth of it's original size and the transformations are applied on this copy. Upon finalization the orientation is modified to fit the full size image.

(a) original (b) Trapezoid transformed

Figure 5.2: Trapezoid transformation

The transformation in Figure 5.2 is sometimes necessary because the cameras in the device seem to not be aimed exactly at the same angle away from each other. This transformation can correct that to a degree.

Chapter 6

Blurring

Blurring or also known as smoothing is the process of averaging out pixels to their neighbours in various ways and reducing the overall sharpness of the image. This means loss of texture, detail and edges.
As simple as this operation sounds there are still some options for how the blurring can be performed. There are many types of blurring. Box blur, Gaussian blur, median blur, bilateral blur, to name a few possible candidates.
The most standard of these is the box-blur, this simply takes for every pixel a k by k pixels square around the pixel and calculates the mean. This mean becomes the new value of the pixel. For every blurring method holds that the larger the value for k the stronger the blurring effect becomes. See Figure 6.1.

(a) source image (b) box-blur, k = 7 (c) box-blur, k = 15 (d) box-blur, k = 21

Figure 6.1: Different levels of box-blur applied to an image.

Gaussian blur is similar except that the distances from the pixels to the target pixel are also taken into the equation. So the new pixel value is the mean of the sum of the surrounding pixels multiplied by the distance from the target pixel. See Figure 6.2.

(a) source image (b) Gaussian blur, k = 7 (c) Gaussian blur, k = 15 (d) Gaussian blur, k = 21

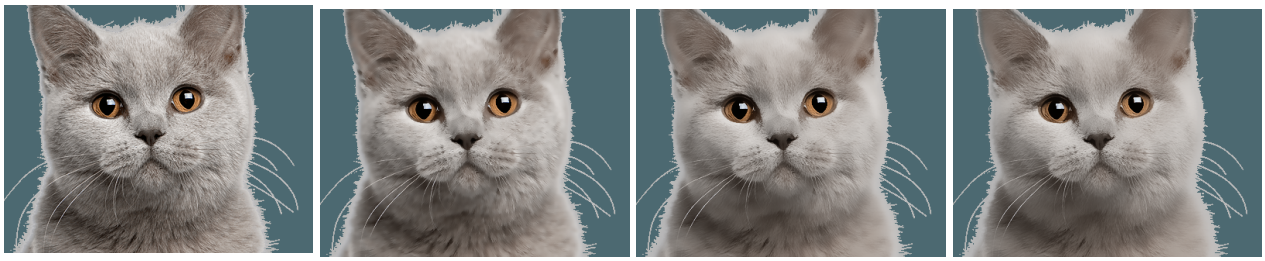
Figure 6.2: Different levels of Gaussian blur applied to an image.

Median blur is fairly straightforward, it takes the median of the pixels neighbours as target value. This smoothing creates large portions of bland color. Which is not fit to hiding transitions. See Figure 6.3.

(a) source image (b) median blur, k = 7 (c) median blur, k = 15 (d) median blur, k = 21

Figure 6.3: Different levels of median blur applied to an image.

Bilateral filtering [PKT+09] is the most complex techniques of these. It is similar to Gaussian blurring in that it also assigns weights to every pixel in range. These weights have two components to them, the first is the same as for Gaussian blur, the distance of the pixel to the evaluated pixel. Referred to as σ_{space}. The second is the difference in intensity between the evaluated pixel and the neighbouring pixel. Referred to as σ_{color}. It is possible to control how much the two components to the weights effect the evaluated pixel via the σ values. In our example both σ_{space} and σ_{color} are set to 75 for every image. Higher settings strengthens the effect. While altering the balance between the different σ's changes the balance of the effects. See Figure 6.4. Notice how hard edges and features of the cat are preserved while the fur texture is smoothed out.



(a) source image (b) bilateral blur, k = 7, $\sigma = 75$ (c) bilateral blur, k = 15, $\sigma = 75$ (d) bilateral blur, k = 21, $\sigma = 75$

Figure 6.4: Different levels of bilateral filtering applied to an image.

For the blurring process the Gaussian blur came out as the best candidate. Whilst being slightly more time intensive than box-blur the result is smoother. Median blur tends to produce plain areas, which will only make the transition more obvious. Bilateral filtering gives interesting effects but the feature of preserving edges in the content is not beneficial to hiding the transition in the image.

(a) Guassian blur 400 pixels wide, k = 55

(b) Gaussian blur 400 pixels wide, k = 155

Figure 6.5: Different levels of Gaussian blur to smooth transition.

In Figure 6.5 it is evident that some level of blurring is beneficial to smoothing out irregularities in the transition. The exact amount will differ between images and also on how well the positioning is done. A problem that is not addressed by this method is the difference in contrast that can exist between the two images. Another problem that is introduced by this method is the sudden transition between the blurred area and the normal parts of the image. Essentially this creates two new edges per transition. In order to reduce this effect the blurring can be done gradually.

(a) Guassian blur 400 pixels wide,
5 sections, k = 15, 45, 75, 45, 15

(b) Gaussian blur 400 pixels wide,
7 sections, k = 15, 35, 55, 75, 55, 35, 15

Figure 6.6: A gradual Gaussian smoothing using multiple sections.

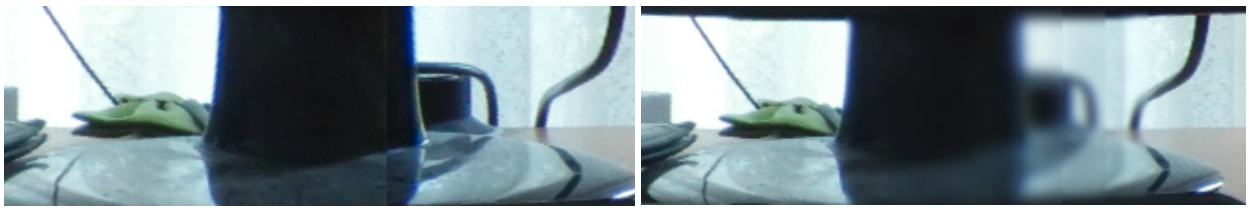
In Figure 6.6 the blurring is done gradually by dividing the blurred area into sections and increasingly and then decreasingly blurring the sections. Since Gaussian blurring is a linear effect, this can be observed from the testing, every section has a linearly increased or decreased k-size.
Lastly the width of the blurring is important, the seem needs to be wide enough to ensure some obvious discontinuities in the image are masked. But when the seam becomes to wide the blurring becomes more obvious in the image reducing the quality of the result again. See Figure 6.7.

(a) Gaussian blur 400 pixels wide,
7 sections, k = 15, 35, 55, 75, 55, 35, 15

(b) Gaussian blur 600 pixels wide,
7 sections, k = 15, 35, 55, 75, 55, 35, 15

Figure 6.7: A gradual Gaussian smoothing using multiple sections, with different widths.

From these tests the best result is figure 6.7a. The total time to apply the blur to the image is roughly 0.5 seconds. This is the fastest solution to stitch the images out of the three proposed ones.



(a) No blurring

(b) Gaussian blur 600 pixels wide,
7 sections, k = 15, 35, 55, 75, 55, 35, 15

Figure 6.8: A highlight comparison between a blurred transition and no blurring.

In Figure 6.8 a highlight is shown of how blurring can help smooth out transitions. In the figure a close up of the monitor stand can be seen. In the before shot there is a slight misalignment and a contrast difference. The blurring removes the sharpness of the transition and the contrast difference is not immediately noticeable in the result. However the sharpness of the image near the transition is permanently lost.

Chapter 7

Blending

Blending is a good method to hide transitions when the content in the image is very repetitive. When image content gets more complex it becomes easier to see semantic inconsistencies like ghosting.
The type of blending used in this implementation is Poisson blending. Generally Poisson blending is used to blend a source image into a destination image. As opposed to blending the edges of two images together, which we want to accomplish. Some creativity is needed there and will be discussed later.

7.1 Poisson blending

Poisson blending [PGB03] requires a source image and a target image. The source image has a specified area to transfer from and the target image an area to transfer to. The area around the target location is known and we want to have a smooth transition from the target image to the source image pasted content. This leaves us with an interpolation problem. It is enough to solve the interpolation for each color component independently and then merge the channels.

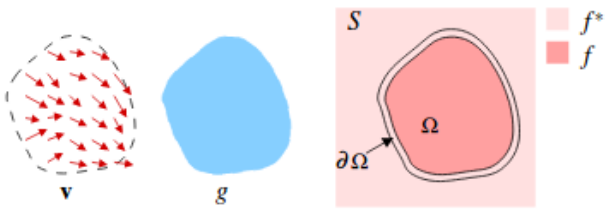


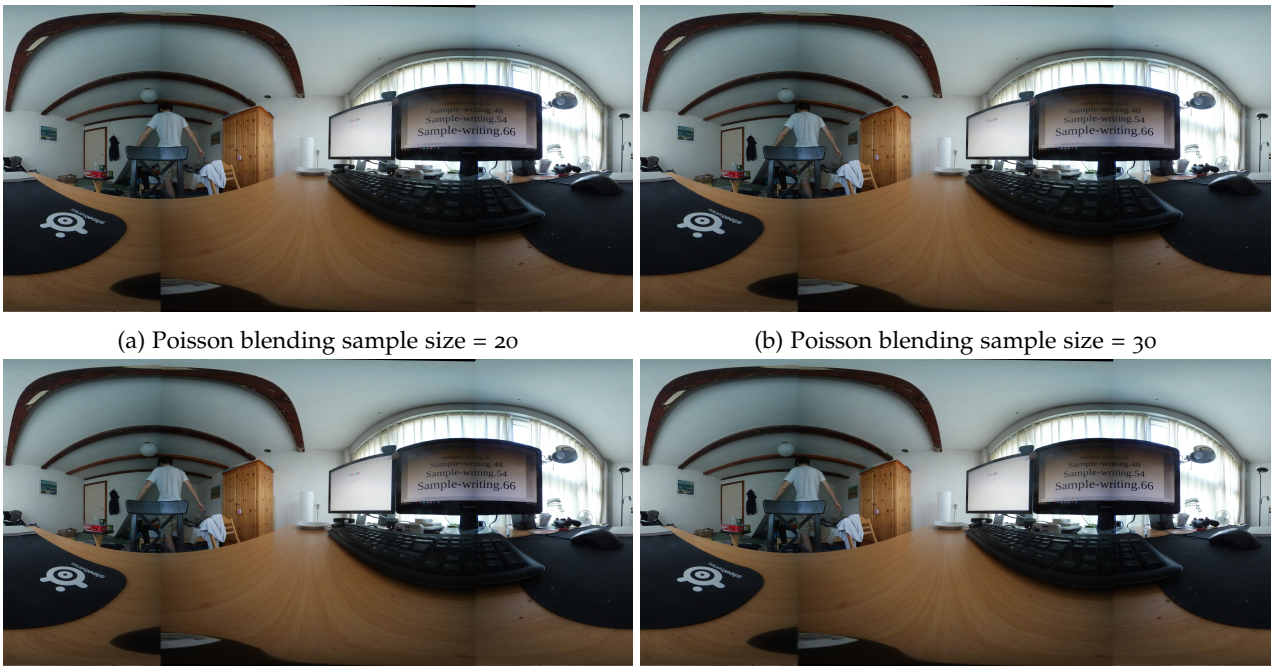
Figure 7.1: Guided interpolation notations. Unknown function f interpolates in domain Ω the destination function f^*, under guidance of vector field v, which might be or not be the gradient field of a source function g [PGB03].

In Figure 7.1 the notation is explained. So Ω is our target area which we need to interpolate with an unknown function f. We have the known function f^* outside Ω. g denotes the source content we need to transform and v the content of the target image at the specified place. We can use the guidance field v with the minimization

problem derived from the interpolation problem to create a new minimization problem which we can solve using the Poisson equation [PGB03].

7.2 Tests

To apply this in the image stitching process an overlap area is specified. This is an area from the top of the transition to the bottom and is centered around the transition. It is certain amount of pixels wide, referred to as the overlap. This overlapping area is taken from either of the images and determines how wide the blended area will be.
This area is used as the source image and square regions are repeatedly taken from the source image and blended with the corresponding area in the target image. In essence they are patched over the seam. Since the source image on one side is identical to the target image the blending there is seamless. Problems as seen with blurring such as where the blurring introduces new transitions are avoided.



(a) Poisson blending sample size = 20

(b) Poisson blending sample size = 30

(c) Poisson blending sample size = 40

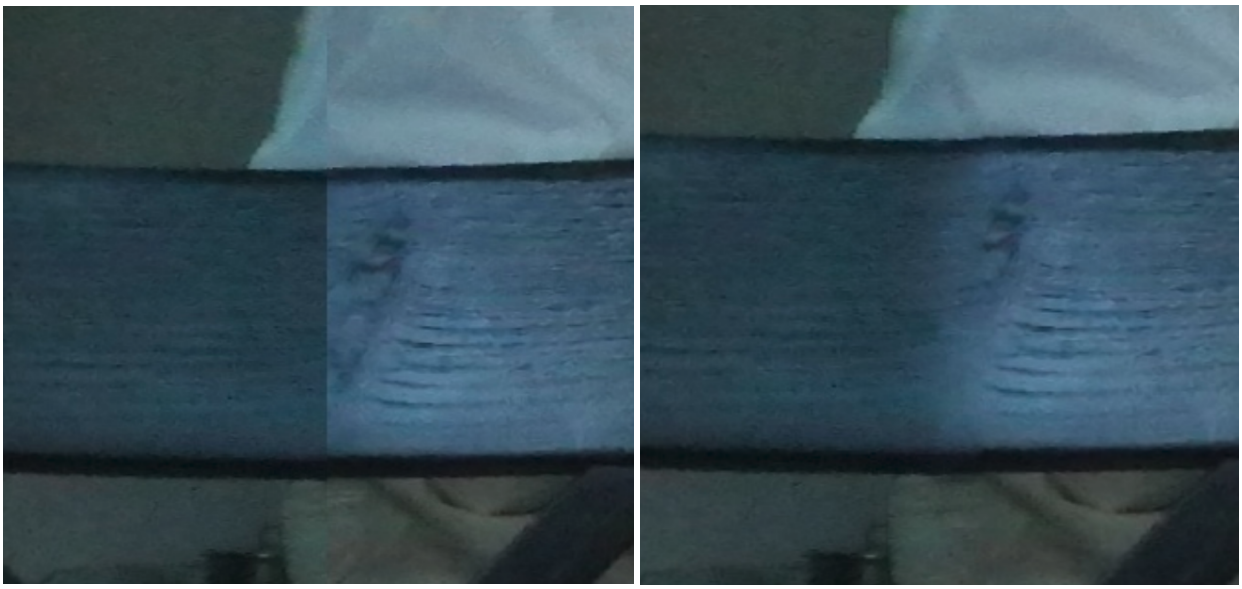
(d) Poisson blending sample size = 50

Figure 7.2: Poisson blending with different sample sizes.

What can be observed from Figure 7.2 is that a small amount of blending can already mask the sharp line of the transition. In general bigger errors in the transition need more blending, so an optimal amount of blending depends on the quality of the positioning. In this image most inconsistencies are resolved at a sample size of 50 pixels. Some close-ups of this will be discussed later.

sample size (pixel)	time(s)
20	45
30	154
40	691
50	2087

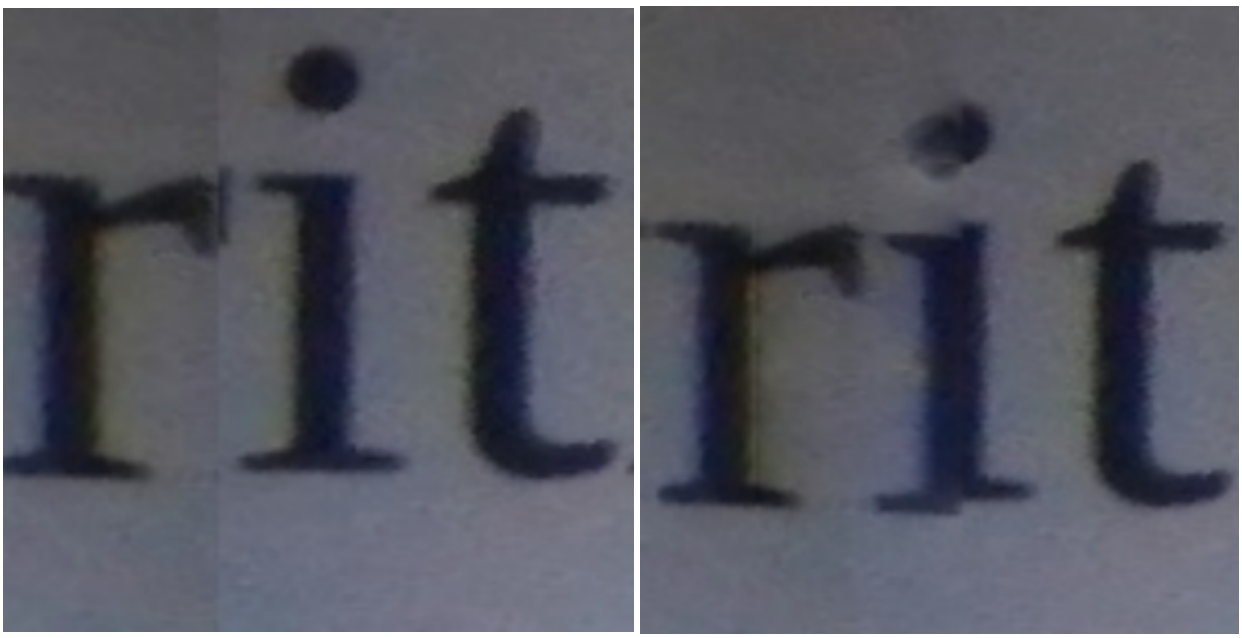
A 30 by 30 pixel area is sampled from a 30 by 4000 overlap area. 40 by 40 pixels is gives a 40 by 4000 overlap area, meaning an increase with a factor of 1.33. The increase in time however is from 154 seconds to 691 seconds, a 4.5 times increase. Therefore the processing time grows exponentially for larger overlap sizes. See above table.



(a) no blending (b) Poisson blending, sample size = 50

Figure 7.3: A highlight side by side comparison of a simple situation

In Figure 7.3 a side by side comparison can be seen of the initial situation and after processing with Poisson blending very little of the transition can be noticed on the wood texture. Also the cloth areas in the image have a smooth transition. The image transitions seamlessly into the blended area as well. Also the difference in contrast is taken into account with this method. There is a gradual transition in contrast between the two images. It is evident from this example that simple textures are blended very well with Poisson blending.



(a) no blending (b) Poisson blending, sample size = 50

Figure 7.4: A highlight side by side comparison of complex situation.

In Figure 7.4 a more complex scene can be seen. What makes this scene more complex is that sleight semantic inconsistencies in writing are easily noticed. Inconsistencies in wood textures are much harder to spot. In this blending example some problems with the implementation can be seen. Since the blending is done with the content of the left image in this case a badly overlapping part of the image is taken and blended into the right part. This causes the letter “i” to be deformed. Some improvement can still be seen on the letter ”r”, the misalignment errors are masked in this case. Poisson blending clearly does not perform as well on semantically strict content.

Chapter 8

Dynamic stitching

The dynamic stitching method is a bit different from earlier discussed techniques since it does not try to hide a seem but rather try to make the seem follow a path where it is hard to see the actual transition. In order to do this some steps need to be taken.
First off the images need to be positioned in a way where the overlapping parts are exactly positioned over top each other with the symmetry axis in the center of the image. See figure 8.1.

(a) Left side, transition 1 (b) Right side, transition 1 (c) Left side, transition 2 (d) Right side, transition 2

Figure 8.1: Both sides of both transition positioned exactly over each other.

8.1 Color distance

We then calculate the Euclidean distance between the pixels corresponding pixels to obtain a map of weights for the transition. We can do this in different color spaces. In this test we show RGB and HSV color space. The Euclidean distance D is calculated as follows:

$$D = (left.r - right.r)^2 + (left.g - right.g)^2 + (left.b - right.b)^2$$

For the RGB color space.

$$D = (left.h - right.h)^2 + (left.s - right.s)^2 + (left.v - right.v)^2$$

For the HSV color space.
To be precise this is the squared Euclidean distance, but since we are comparing distance with each other inside the same image it saves computing power to leave the value squared. A fundamental problem with this approach is that both RGB and HSV Euclidean distances do not accurately represent how humans perceive color similarity. HSV does come closer in terms of correctness, but is still not close to perfect.

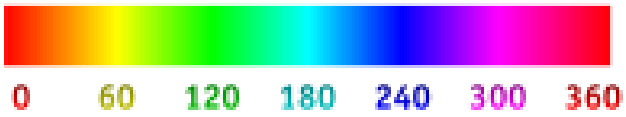
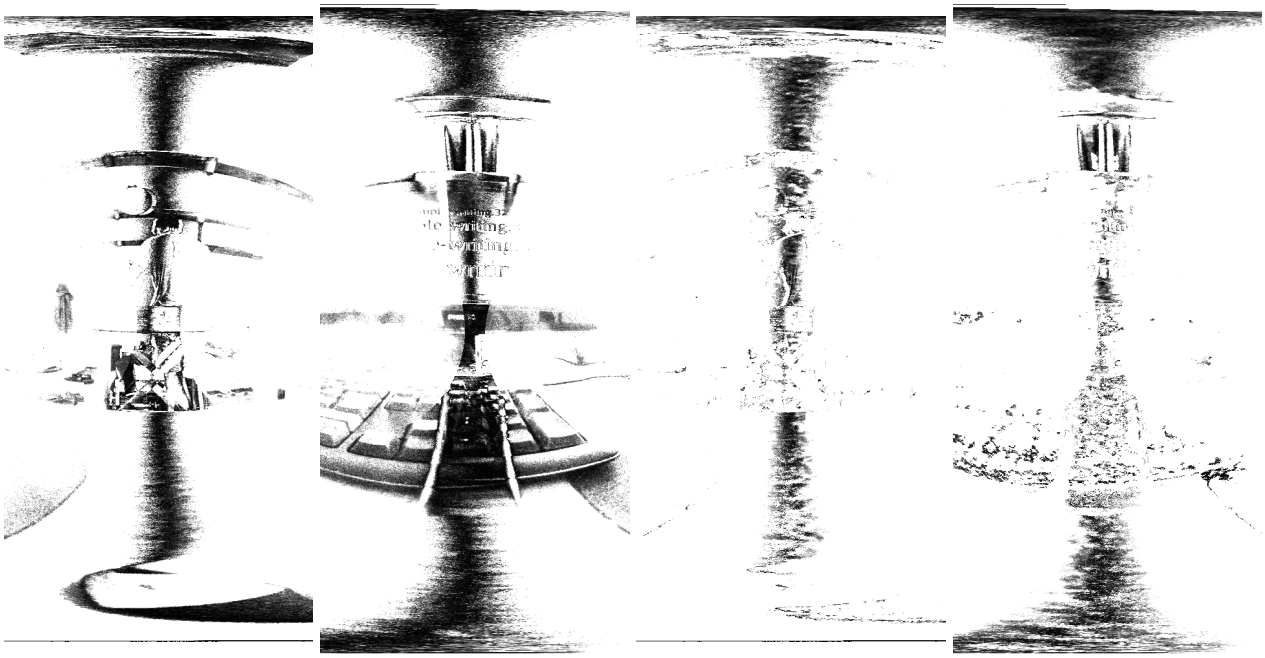


Figure 8.2: Gradient of hues.

In figure 8.2 a gradient of all possible hues can be seen. From this we can observe some of the problems with using the hue as basis of similarity. For one both sides of the spectrum end with red which in our calculation would be maximally different. Also not all colors are perceived by humans to be similar in the same quantities as the Euclidean distance makes them out to be.



(a) Weights based on Euclidean RGB distance, transition 1

(b) Weights based on Euclidean RGB distance, transition 1

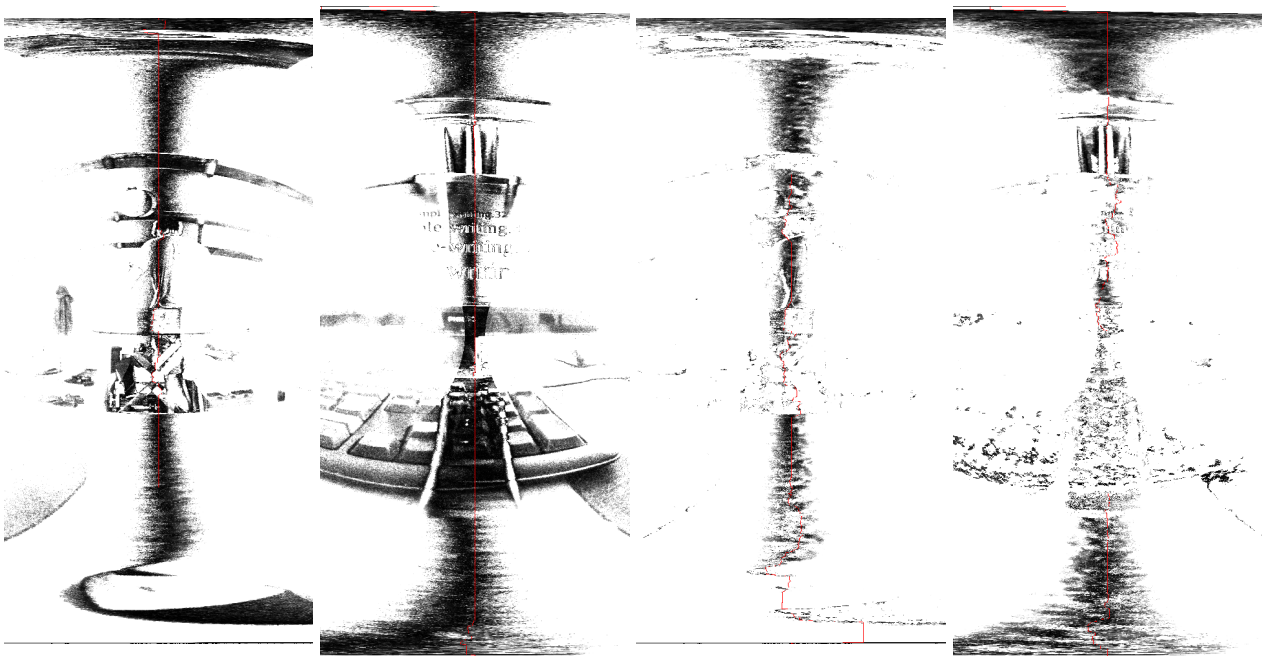
(c) Weights based on Euclidean HSV distance, transition 1

(d) Weights based on Euclidean HSV distance, transition 1

Figure 8.3: Weights for Euclidean distance, normalized to [0 and 255]
Darker pixels is lower weights.

8.2 Shortest path

Now we need to calculate the shortest path from the top of the image to the bottom. In this case we use the A-star algorithm. However given the complexity and complexity of the problem finding the shortest path from top to bottom is too big a problem. The search space being roughly 2000 by 4000 weights. Therefore we divide the problem into smaller sections. Four points evenly along the center of the image are set and the shortest path between each pair is calculated and combined as the full solution. For practical use if a pair takes too long to finish it is timed out and a straight line is used instead.



(a) Shortest path based on RGB weights, transition 1

(b) Shortest path based on RGB weights, transition 2

(c) Shortest path based on HSV weights, transition 1

(d) Shortest path based on HSV weights, transition 2

Figure 8.4: Weights for Euclidean distance, normalized to [0 and 255]
Darker pixels is lower weights, shortest path in red

The results can be seen in figure 8.4. For RGB transition 1 the lowest quarter of the map was not completed. For HSV transition 1 the lower middle part and for transition 2 the top quarter. Paths that timed out do not show up in this image. Time out occurs after 5 minutes.

Every point in the path is iterated and a mask is created from this, this mask is then applied to the images for the result. See Figure 8.5

(a) Dynamic stitch based on RGB

(b) Dynamic stitch based on HSV

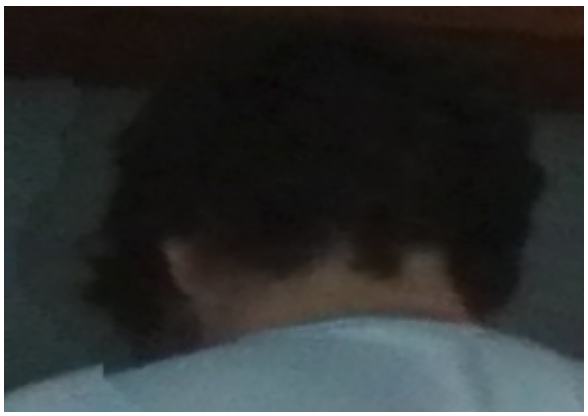
Figure 8.5: Complete result dynamic stitching

Whilst the differences between RGB and HSV based dynamic stitching are hard to predict from the weight maps. The differences on the results are significant. See figure 8.5. We will display some highlights with key differences in performance. Areas that timed out will not be taken in consideration.

(a) RGB highlight, chair

(b) HSV highlight, chair

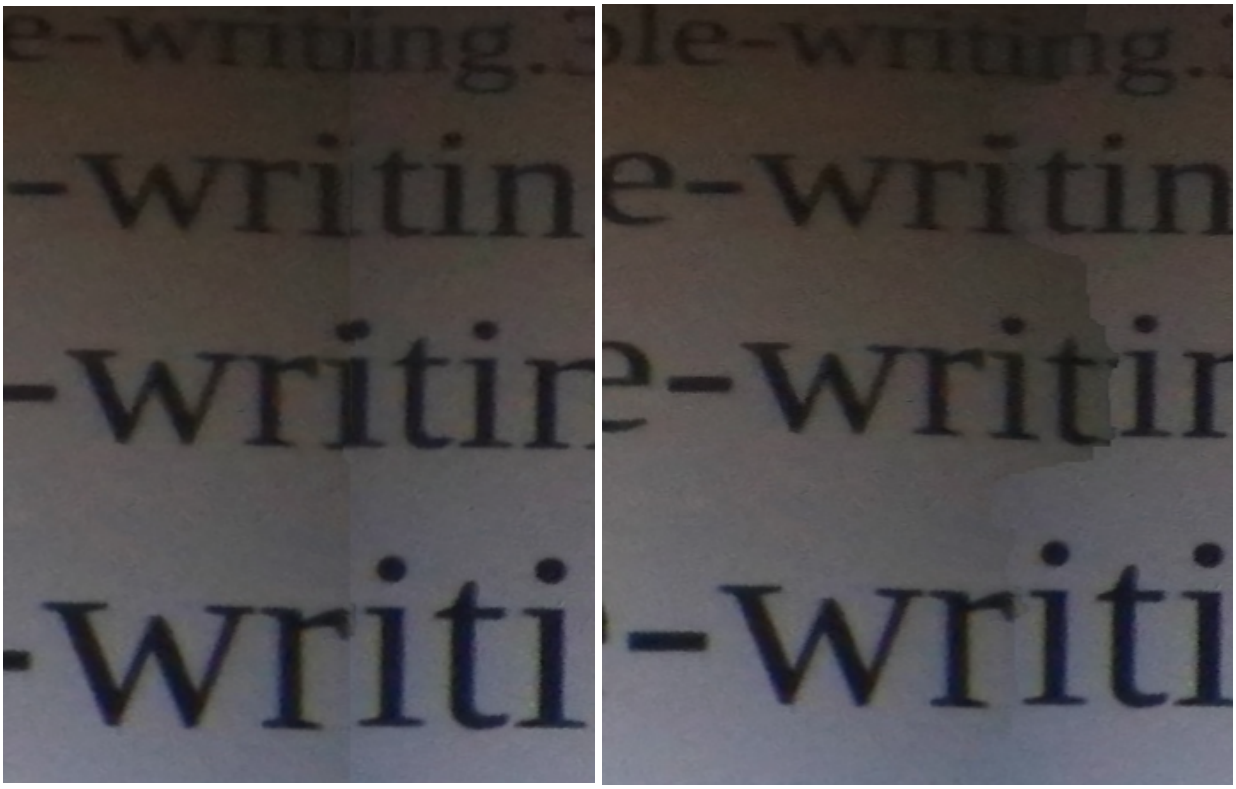
(c) RGB highlight, head



(d) HSV highlight, head

Figure 8.6: Different scenes for comparison RGB and HSV weights

Figure 8.6 contains two comparisons between different regions in the result. In Figure 8.6a and 8.6b similar performance can be seen between RGB and HSV based stitching. However in more complex scenes, see figure 8.6c and 8.6d more drastic improvements can be seen. The border can still be seen because of the contrast difference but notice how the transition in the image avoids the area with the hair in the HSV highlight. The shortest path tends to avoid big shifts in color so objects with large contrast against the background are avoided as a result.



(a) (b)

Figure 8.7

The effect of avoiding objects with large contrast is especially evident when stitching text. See Figure 8.7. Further discussion and comparison continues in chapter 9.

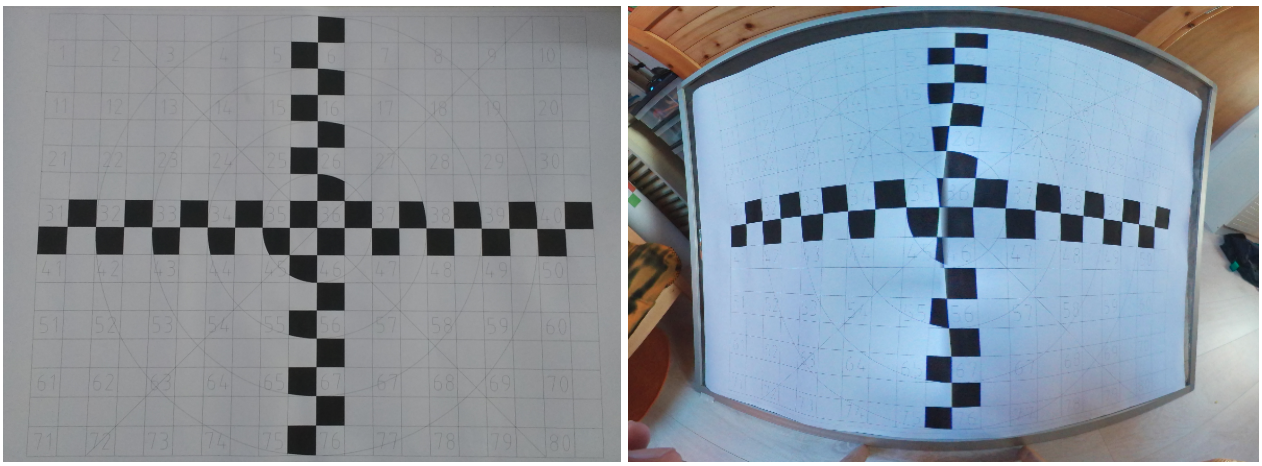
Chapter 9

Comparison

The three different stitching techniques all have their own benefits and drawbacks. In this chapter some highlights are shown with different source images to compare them to Samsung's implementation of the stitching.

Figure 9.1: Sample output of Samsung's implementation

In Figure 9.1 Samsung's result can be seen. This is the same image we have used throughout the paper for testing. A quick observation shows that Samsung is most likely using some form of blending to join the images. Notice the places where sharp transitions exist in Figure 9.1 and how the color fades over here. Also the vignetting effect is successfully removed either by this blending or an extra step reducing the effect as seen in the work of Chu et al [LCC+15].



(a) Original (b) Output from Samsung

Figure 9.2: Failure to align images.

Sometimes Samsung's alignment process fails to recognize and match the right feature points. This leads to image content being lost near transitions as in Figure 9.2. Another result of this can be ghosting, the opposite of the before mentioned effect. Ghosting is the repeat of content near image transition due to misalignment. See Figure 9.3. The image is a close-up of a photo containing a railroad. The tree and the overhead line holding pole are repeated here.

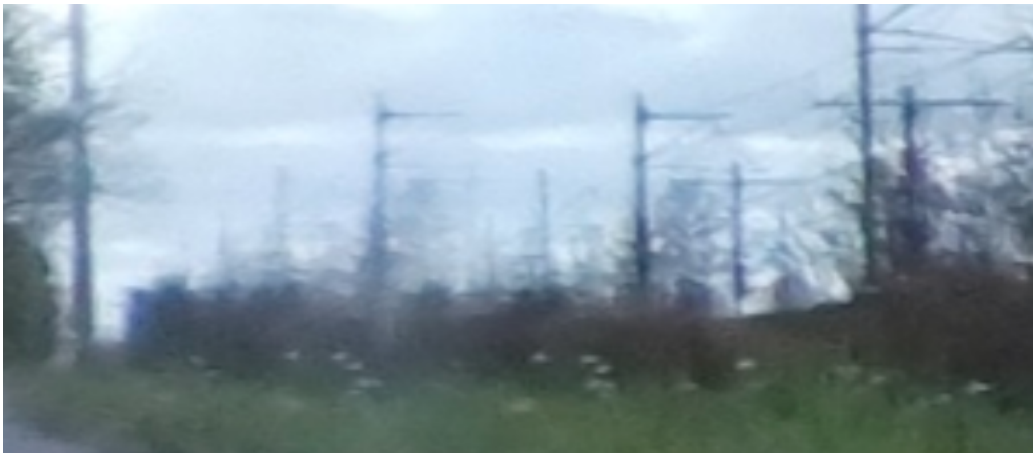


Figure 9.3: Failure to align images, output from Samsung

First we will show some examples of Samsung's output versus the blurring technique of chapter 6. Then a comparison between Samsung's blending and the Poisson blending shown in chapter 7. Lastly we compare with results of the dynamic stitching technique of chapter 8.

9.1 Blurring

Blurring is a very basic method and does not compare well to the output of Samsung's algorithm. The stitching edge is fairly obvious. However in the case of heavy misalignment blurring can help to cover up big semantic inconsistencies.

(a) Result of Samsung

(b) 400 pixel wide gradual blur.

Figure 9.4

In Figure 9.4 the wrong alignment in the chairs back support is more obscured by the blur. Outside of the heavy misalignment Samsung's algorithm performs better across the board.

9.2 Blending

Samsung uses a type of blending as well. However the process is much faster than the implementation with only a few seconds per image needed.

(a) Result of Samsung.

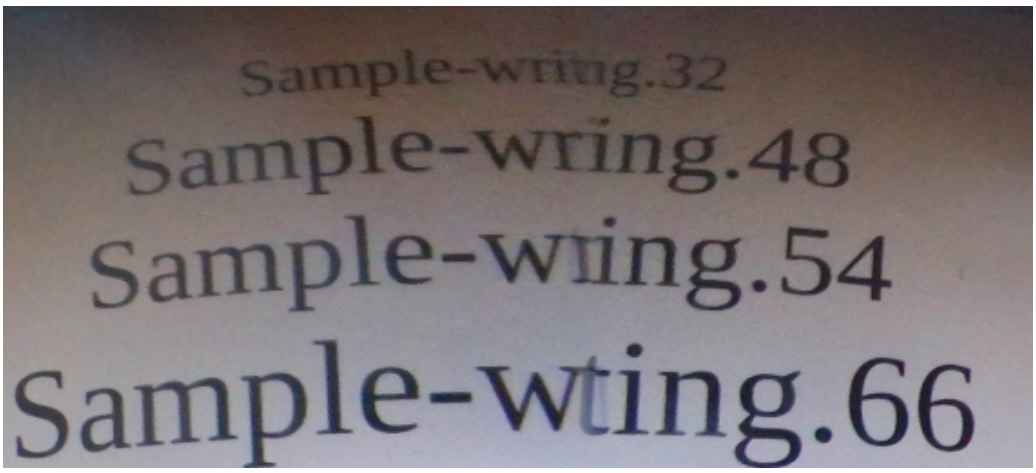
(b) 50 pixel overlap Poisson blending.

Figure 9.5: Comparison between Samsung's blending and the blending of the implementation.

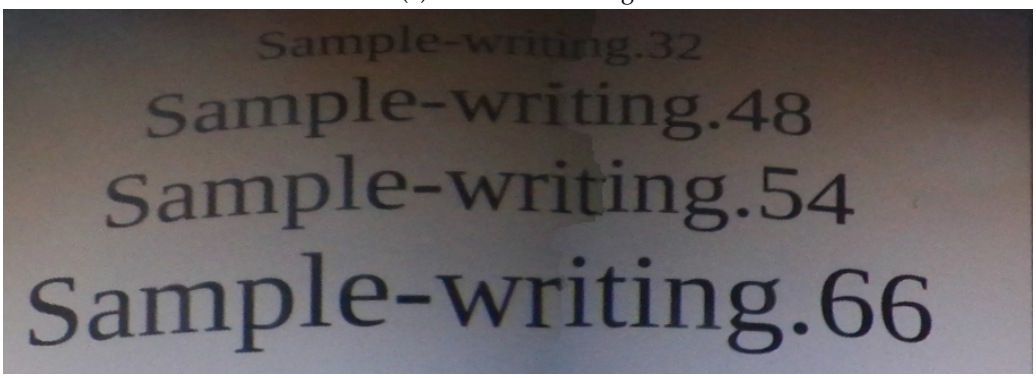
With good manual positioning the proposed blending performs similar to the blending implemented by Samsung. See Figure 9.5. The clouds are blended very similarly as well as the foreground. The same problem with aligning the images was also present in our method.

9.3 Dynamic Stitching

A problem with Samsung's stitching is that it does not take into account image content when deciding the stitching line. There is a generous overlap, however the stitching line is always straight down the middle. In most scenes this produces a visually pleasing stitching line regardless, though when a semantically strict content such as text is near image borders blending can give a bad result.



(a) Result of Samsung



(b) Dynamic stitch based on HSV color space.

Figure 9.6: Comparison between Samsung's result and the dynamic stitching.

In Figure 9.6 a text reading "Sample-writing" is near the transition. In the results of Samsung the text is cut off due to incorrect alignment and the letters are also blended with the background. In the result of the dynamic stitch proposed letters are avoided by the stitching line. The letters retain their sharp contours.

Overall some interesting proposals have been done to improve the stitching for the Samsung Gear 360. Most promising is the dynamic stitching. Combining this with a light blending to remove the contrast difference would be a great improvement. Alternatively a method as proposed in Ward [War06] is also a great method to both retain edges and blend textures in scenes.

Chapter 10

Evaluation

I can honestly say now that the project is finished that I learned a lot throughout the entire course of it. Before I started I had very little knowledge of computer graphics and computer vision. I learned how by reading papers and surveys you can quickly get familiar in a new field. Also the OpenCV library was a great choice for its extensive tutorials and plenty support existing online to get started and find solutions to technical problems.

The bachelor class sets clear deadlines for progress in the project throughout the year. This gives a good framework to pace yourself around when writing and developing. The tutorials given in the course were helpful and for the most part timely to the progress in the project.

I am content with my achievements in the project. I managed to achieve a result rivalling the results from Samsung. The dynamic stitching is especially interesting with its ability to avoid make transitions less complex and therefore improve the stitching.

If I was given the opportunity to expand more on the program and paper I would try again to implement automatic positioning. The process of feature point detection extraction and matching is an interesting but complex one. Also combining the techniques of dynamic stitching and blending would definitely give even better results. Lastly a working implementation of a devignetting process would greatly help the overall result.

Chapter 11

Conclusions

We have compared different approaches to stitching to improve the quality of the raw output of the Samsung Gear 360. A 360° panorama camera. The tests in the Chapters 6, 7 and 8 show how the different techniques give definite improvement over simply splicing. In particular cases the output of one of our methods outperforms the output of Samsung's software. Most notably the dynamic stitching method shown in Chapter 8 shows significant improvements when stitching complex scenes.

Bibliography

[BA83] Peter J. Burt and Edward H. Adelson. A multiresolution spline with application to image mosaics. *ACM Trans. Graph.*, 2(4):217–236, October 1983.

[BL07] Matthew Brown and David G. Lowe. Automatic panoramic image stitching using invariant features. *Int. J. Comput. Vision*, 74(1):59–73, August 2007.

[BXXL12] Chunxiao Bian, Chuangbai Xiao, Huiqin Xi, and Yi Liu. Stitching line and deformation propagation for seamless image stitching. In *Proceedings of the 27th Conference on Image and Vision Computing New Zealand*, IVCNZ '12, pages 262–267, New York, NY, USA, 2012. ACM.

[Fon15] Chamberlain Fong. Analytical methods for squaring the disc. *arXiv preprint arXiv:1509.06344*, 2015.

[HKL+07] S. J. Ha, H. I. Koo, S. H. Lee, N. I. Cho, and S. K. Kim. Panorama mosaic optimization for mobile camera systems. *IEEE Transactions on Consumer Electronics*, 53(4):1217–1225, Nov 2007.

[KH10] Michael Kazhdan and Hugues Hoppe. Metric-aware processing of spherical imagery. *ACM Trans. Graph.*, 29(6):149:1–149:10, December 2010.

[LCC+15] Yu-Hsin Lin, Yu-Mei Chen, Lun-Cheng Chu, Andre Chen, Scott Chien-Hung Liao, and Edward Y. Chang. Pan360: Ins assisted 360-degree panorama (demo description). In *Proceedings of the 23rd ACM International Conference on Multimedia*, MM '15, pages 795–796, New York, NY, USA, 2015. ACM.

[MB00] Laurent Meunier and Moritz Borgmann. High-resolution panoramas using image mosaicing. *Final Project*, 2000.

[MJYD01] Zhang Maojun, Xie Jingni, Li Yunhao, and Wu Defeng. Color histogram correction for panoramic images. In *Proceedings Seventh International Conference on Virtual Systems and Multimedia*, pages 328–331, 2001.

[OKK12] Tomohiro Ozawa, Kris M. Kitani, and Hideki Koike. Human-centric panoramic imaging stitching. In *Proceedings of the 3rd Augmented Human International Conference*, AH '12, pages 20:1–20:6, New York, NY, USA, 2012. ACM.

[PEH+08] Peter Pesti, Jeremy Elson, Jon Howell, Drew Steedly, and Matt Uyttendaele. Low-cost orthographic imagery. In *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '08, pages 24:1–24:8, New York, NY, USA, 2008. ACM.

[PG16] Daniel Pohl and Oliver Grau. Concept for content-aware, automatic shifting for spherical panoramas. In *Proceedings of the 22Nd ACM Conference on Virtual Reality Software and Technology*, VRST '16, pages 321–321, New York, NY, USA, 2016. ACM.

[PGB03] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. In *ACM Transactions on graphics (TOG)*, volume 22, pages 313–318. ACM, 2003.

[PKT+09] Sylvain Paris, Pierre Kornprobst, Jack Tumblin, Frédo Durand, et al. Bilateral filtering: Theory and applications. *Foundations and Trends® in Computer Graphics and Vision*, 4(1):1–73, 2009.

[PN11] Yulka Petkova and Nuri Nuri. An algorithm for fast image registration and feature matching for the purposes of image stitching. In *Proceedings of the 12th International Conference on Computer Systems and Technologies*, CompSysTech '11, pages 228–233, New York, NY, USA, 2011. ACM.

[PP95] B. Pham and G. Pringle. Color correction for an image sequence. *IEEE Computer Graphics and Applications*, 15(3):38–42, May 1995.

[TGTC02] Gui Yun Tian, D. Gledhill, D. Taylor, and D. Clarke. Colour correction for panoramic imaging. In *Proceedings Sixth International Conference on Information Visualisation*, pages 483–488, 2002.

[War06] Greg Ward. Hiding seams in high dynamic range panoramas. In *ACM SIGGRAPH 2006 Research Posters*, SIGGRAPH '06, New York, NY, USA, 2006. ACM.

[XP09] Yingen Xiong and Kari Pulli. Color correction for mobile panorama imaging. In *Proceedings of the First International Conference on Internet Multimedia Computing and Service*, ICIMCS '09, pages 219–226, New York, NY, USA, 2009. ACM.

[YHLX13] Tao Yan, Zhe Huang, Rynson W. H. Lau, and Yun Xu. Seamless stitching of stereo images for generating infinite panoramas. In *Proceedings of the 19th ACM Symposium on Virtual Reality Software and Technology*, VRST '13, pages 251–258, New York, NY, USA, 2013. ACM.