FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

All-in-Focus Image Generation using Automatic Depth-Based Focus Selection

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Michael Pointner
Matrikelnummer 01427791

an der Fakultät für Informatik
der Technischen Universität Wien
Betreuung: PhD Stefan Ohrhallinger

Wien, 8. August 2017

Michael Pointner	Stefan Ohrhallinger

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

All-in-Focus Image Generation using Automatic Depth-Based Focus Selection

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Michael Pointner
Registration Number 01427791

to the Faculty of Informatics
at the TU Wien
Advisor: PhD Stefan Ohrhallinger

Vienna, 8th August, 2017

Michael Pointner

Stefan Ohrhallinger

Erklärung zur Verfassung der Arbeit

Michael Pointner
Vorgartenstraße 129-143/1/17

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 8. August 2017

Michael Pointner

Danksagung

Mein besonderer Dank gilt meinem Betreuer bei dieser Arbeit PhD Stefan Ohrhallinger für die tolle, richtungsweisende, aber gleichzeitig auch offene und kompromissbereite Begleitung während der Implementierung und dem Schreiben dieser Arbeit.

Weiters möchte ich mich bei ihm und dem Institut für Computergraphik und Algorithmen an der technischen Universität Wien für die Bereitstellung des Testgerätes sowie der Räumlichkeiten für die Aufnahme von Bildmaterial für die Evaluierung bedanken.

Und zu guter letzt möchte ich mich noch bei meinen KorrekturleserInnen Herbert Pointner, Nathalie Pleidl und Eva Rotky herzlich bedanken.

Acknowledgements

My special thanks go to my advisor for this work PhD Stefan Ohrhallinger, for the great and directional but at the same time also open and compromise-oriented accompaniment during the implementation and the writing of this work.

Furthermore, I would like to thank him and the Institute of Computer Graphics and Algorithms at the Vienna University of Technology for the provision of the test device as well as the rooms for the recording of photographic material needed for the evaluation.

And last but not least, I would like to thank my reviewers Herbert Pointner, Nathalie Pleidl and Eva Rotky.

Kurzfassung

Eines der wichtigsten Ziele beim Fotografieren ist die Schärfe des ganzen Bildes, was jedoch aufgrund der physikalischen Eigenschaften einer Kameralinse nicht so einfach möglich ist. Anlässlich Google's neuer Tango API, welche das Wahrnehmen von Tiefeninformationen auf Smartphones ermöglicht, haben wir die Nutzbarkeit dieser neuen Technologie zur Generierung von ganzheitlich scharfen Bildern durch Mehrfach-Focus mit dem Lenovo Phab 2 Pro als erstes Smartphone, dass diese Technologie unterstützt, evaluiert.

Abstract

One of the most important objectives in photography is the sharpness of the whole image, which is not so easy to achieve because of the physical properties of a camera lens. Due to Google's new Tango API, which enables the perception of depth information on smartphones, we have evaluated the usability of this new technology for the generation of all-sharp images through multi-focus with the Lenovo Phab 2 Pro as the first smartphone to support this technology.

Contents

CHAPTER 1

Introduction

Embedded cameras in Smartphones not only open a wide range of possibilities for taking advantage of this feature in applications like video calls, but also enable us to take photos wherever we are without the necessity to carry a heavy DSLR or a pocket camera. Nevertheless, there still exist some challenges concerning the image acquisition due to physical characteristics of the lens system as in traditional cameras. One example is the struggle to capture a wide range of different illuminated scene elements with the same brightness called HDR. Some of the latest generation smartphones already have an integrated feature to automatically capture HDR images.

However, there are still no build-in solutions to the topic of all-in-focus or multi-focus image generation. Because of the point spread function cameras focus at a certain distance which therefore is in-focus. Also, an interval around the focus distance called depth of field is acceptably sharp, while the rest of the image is blurry.

There exists a variety of approaches trying to solve this issue, from extending the depth of field by lowering the lens aperture to capturing multiple images with different focal distances and stitching them together. The problem of stitching is that it requires depth information to select an image in which a pixel is in-focus. However, hardly any camera or smartphone has a depth sensor integrated to this day.

This gap is now closed by Google's new Tango API which enables smartphones like the Lenovo Phab 2 Pro, the first smartphone to support the API, to perceive depth information and use it in applications. The Tango API supports a smartphone's built-in depth sensor and fish-eye camera to perceive depth information and perform motion tracking.

For this thesis we obtained a Lenovo Phab 2 Pro (Figure 1.1) and created an application to evaluate Tango's usability for all-in-focus image generation. We use the terms multi-focus and all-in-focus interchangeably since we capture images by focusing at multiple distances,

Figure 1.1: Lenovo Phab 2 Pro - the first smartphone supporting Google's Tango API. On the back from top to bottom: Camera, Infra-red depth sensor, Fish-eye camera for motion tracking, fingerprint sensor [Source: www3.lenovo.com/at/de/tango]

but the resulting image after stitching is an all-in-focus image due to overlapping depth of fields.

The content of this thesis is structured as follows: First, we will talk about related work in this research area, in order for you to get an impression of the state of the art. Then we will explain the theoretical basics and principles in the Methodology section and talk about the Implementation of our approach. We conclude with the evaluation of our approach as well as with a resume and future work still open to this topic.

CHAPTER 2

Related Work

There exist two main approaches to achieve all-in-focus images: single-image and multi-image approaches. Single-image approaches try to estimate the defocus blur based on just one image, sometimes assisted by other resources such as depth images or disparity images. In contrast, multi-image approaches like ours achieve all-in-focus images by stitching together multiple images captured with different focal settings.

One of the most relevant single-image approaches by Jung et al. [JPJ16] is to reconstruct all-in-focus images and refocused images using depth images. We will briefly describe the idea behind their approach as well as their essential equations, for detailed derivation of the formula refer to [JPJ16]. Jung et al. use the thin lens model to derive formula 2.1 to estimate the standard deviation of the Gaussian defocus blur σ on the pixels coordinate (i, j) based on a pixel's depth value $u(i, j)$.

$$\sigma(i,j) = | - \frac{\alpha^*}{u(i,j)} + \beta^* | \tag{2.1}$$

α^* and β^* are estimated values for $\alpha = \frac{kFv_0}{f}$ and $\beta = \frac{k(v_0-F)}{f}$, since the camera parameters focal length F, f-number of the lens $f = F/r$ with r being the radius of the lens and the distance between the lens, image plane v_0 as well as the linear factor k are usually not known. k is the linear relationship between the radius of the circle of confusion c and the standard deviation of the Gaussian defocus blur, i.e. $\sigma = kc$.

Hence, Jung et al. [JPJ16] try to estimate the parameters α^* and β^* as a fitting problem based on the edges in the image. The problem is that inherently blurry edges and defocus-blurred edges cannot be easily distinguished. Their strategy in order to deal with this problem is to group the edge pixels obtained by the canny filter ($\Omega_{x,y} = 1$) based on their depth value $u(x, y)$. Furthermore, they take the median of the estimated standard deviations $\sigma(x, y)$ of all edges of a depth value k as shown in equation 2.2 to get

an estimation of the standard deviation of the Gaussian defocus blur $\bar{\sigma}_k$ for the depth k robust to outliers.

$$\bar{\sigma}_k = median\{\sigma(x,y)|u(x,y) = k, \Omega_{x,y} = 1\} \tag{2.2}$$

Now, based on $\bar{\sigma}_k$ for all distances k, the parameters α^* and β^* can be estimated by solving the fitting problem with equation 2.3:

$$\{\alpha^*, \beta^*\} = \underset{\{\alpha,\beta\}}{arg\,min} \sum_k \omega_k \cdot (|-\frac{\alpha}{k} + \beta| - \bar{\sigma}_k)^2 \tag{2.3}$$

In this equation ω_k is a weight, which take the reliability of $\bar{\sigma}_k$ into account. It is empirically chosen as $\omega_k = \frac{1}{\upsilon_k + \varepsilon}$, where $\upsilon_k = var\{\sigma(x,y)|u(x,y) = k, \Omega_{x,y} = 1\}$ and the constant ε used to restrict the maximum value of ω_k is set to 1.

After determining α^* and β^*, a blur map can be calculated with the initial formula 2.1 for the whole image. Using the blur map and the defocused image, the all-in-focus image can be reconstructed via pixel-wise mapping. Refocusing at distance u_F works by calculating a new blur map that can be obtained by using formala 2.1, but instead of α^* and β^* α' and β' are used respectively, where $\alpha'/\beta' = u_F$ and β' can be choosen freely to apply strong/weak refocusing effects. [JPJ16]

Unlike to our approach, Jung et al. [JPJ16] require dense depth images like those returned by stereo systems, which calculate the depth data from the disparity of the objects in two images, in order to have precise depth points at the edges of objects. Infra-red systems like those used on the Lenovo Phab 2 Pro return sparse depth clouds of points detected on surfaces, and consequently the depth at the corners can just be estimated.

Another approach for single-image all-in-focus image generation is the one by Salahieh et al. [SRSL17]. They claim, that in contrast to traditional extended depth-of-field approaches, their prosed depth-based deconvolution technique takes into account the depth-variant nature of the point spread function. Unlike Jung et al. [JPJ16], they bring a single blurred image to focus at different depth planes and stitch those images together based on a depth map. Furthermore, they implement strategies to suppress the devolution's ringing artifacts on 3 levels: block tiling to eliminate boundary artifacts, reference maps to reduce ringing initiated by sharp edges, and depth-based masking to mitigate artifacts raised by neighboring depth-transition surfaces.

According to Salahieh et al. [SRSL17], most software-based deblurring approaches imprecisely approximate the point spread function as a two-dimensional depth-invariant entity, which leads to suboptimal restorations with ringing artifacts especially with included objects in the captured scene that belong to a wide range of depth planes.

Salahieh et al. [SRSL17] use a single blurred image, a multidimensional point spread function and a depth map as input to output a fully focused image. Briefly explained, their approach works as follows: First, they pad the blurred image by interpolating

smooth blocks to eliminate the boundary artifacts. Afterwards, the point spread function volume is sliced axially into various depth planes and fed along with the padded blurred image into an iterative adaptive nonblind deconvolution stage.

The algorithm of Salahieh et al. [SRSL17] iterates to get a better estimate of a reference map employed to locally regularize a smoothing prior, which balances the suppressible ringing and the resolvable details in the deconvolved images associated with the applied axial point spread function slice. They only bring details in each axial deconvolved image to focus if they are located at the exact depth plane correspondent to the applied deblurring point spread function slice. Finally, after an adequate number of iterations, these focused points are extracted from the deconvolved images and stitched together according to the depth map to output an all-in-focus image. [SRSL17] Apart from their computational depth-based deconvolution technique, Salahieh et al. [SRSL17] also describe their strategies to tackle Boundary, Edge and Depth-Transition Artifacts as mentioned above, but for description of those refer to their paper [SRSL17].

In summary, the difference between their approach and the one by Jung et al. [JPJ16] is that they deblur the images layer-wise and stitch them together afterwards.

Another approach which uses stitching of multiple images is from Hasinoff et al. [HK11], but unlike the two discussed approaches they expand the depth of field by capturing multiple images like our approach. Their main observation is that the exposure level of a photo L is linearly proportional to the exposure time τ, but square proportional to the aperture diameter D as shown in equation 2.4.

$$L \propto \tau D^2 \tag{2.4}$$

This equation has some limitations, especially for $D \longrightarrow \infty$ one is limited by the finite radiant power in the scene. The exposure level L of a photo is the total radian energy integrated by the camera's sensor while the shutter is open. This means that a key property of the depth of field is, that it increases when the aperture diameter decreases, following the longer needs to be the exposure time to reach the chosen exposure level. Hence, their approach is to capture and combine multiple photos with different focal lengths, while the camera settings are carefully chosen to minimize total exposure time for a desired depth of field and exposure level.

Suppose that we want to capture a single photo with a specific exposure level L and specific depth of field $[\alpha, \beta]$, there is only one aperture diameter D that can span the given depth of field and only one exposure time τ that can achieve the given exposure level L with that diameter. Therefore, their key idea is that, while lens optics do not permit to reduce this exposure time without compromising the depth of field or the exposure level, it can be reduced by taking multiple photos. Hasinoff et al. [HK11] use tuples of $<D, \tau, \upsilon>$ that specify a photo's aperture diameter, exposure time and focus setting respectively.

To get an optimal capture sequence consisting of a finite ordered sequence of photo tuples, they use a simple branch-and-bound method to solve this integer problem. Since optimal capture sequence only depends on the relative depth of field size α/β, it can be precomputed exactly for all relative sizes and stored in a lookup table. α and β are the boundaries of the depth of field in the focal length domain.

Other than our proposed approach, Hasinoff et al. [HK11] use an existing depth-from-focus and compositing technique for the image stitching without needing a depth sensor. Their depth of field composition operation produces a coarse depth map as an intermediate step, based on maximizing a simple focus measure that evaluates the local contrast according to the Difference-of-Gaussian filter. Each pixel in the composite obtains a label that indicates the input photo in which the pixel is in-focus. These labels are optimized using a Markov random field network, which is biased towards piecewise smoothness. The optimization reduces artifacts at label boundaries, including those due to misregistrations. For the stitching, rather than selecting pixels at in-focus depths from the input sequence, they use the recovered Markov random field optimized depth map to select pixels with appropriate level of defocus and simple linear cross-fading to interpolate between the input photos.

Their results show that they can successfully lower the total exposure time from for example a single-image acquisition with an aperture of $f/16$ and an exposure time of $800ms$ to a total exposure time of $70ms$ using 14 images with an aperture of $f/1.2$ and an exposure time of $5ms$ each. A combination of our approach with Hasinoff et al. [HK11] causing the advantage of lower total exposure time and a more precise depth map at unstructured areas was not possible because the aperture on the Lenovo Phab 2 Pro is fixed to $f/2.2$.

The final two approaches that will be reviewed briefly are focal sweep and plenoptic foto stack capturing. Zhou et al. [ZMN12] propose an imaging system, called focal sweep camera, that directly captures a focal stack for refocusing by physically sweeping its focal plane across a scene. Their object is to capture a focal stack those aggregated depth of fields cover the desired depth range and that no two depth of fields should overlap for a complete and efficient coverage. Furthermore, they found out that the sensor should move by a constant distance between consecutive image captures for an efficient and complete sampling. Therefore, assuming a constant camera's frame-rate, the ideal strategy is to sweep the sensor at a constant speed. The result is a focal stack in which every point of the scene is in focus in at least one image of the focal stack. From this focal stack, an all-in-focus image can be computed using a focus measure to seek out the sharpest image of the focal stack for each pixel.

Another technique to generate a focal stack is to compute it from a light field. A plenoptic camera uses a lens array in the imaging pipeline to capture a 4-D light field. The captured 4-D light field can be used to render a focal stack for image refocusing. However, this results in a significant sacrifice regarding the image resolution since the light field is 4-D and the resulting focal stack is only 3-D, therefore light field cameras capture more information than needed. [ZMN12]

CHAPTER 3

Methodology

In this chapter we will discourse the theoretical concepts and technologies we used. First of all, we present the technology Tango (section 3.1), which we use to acquire depth data, then we describe the concept of focal planes and depth of field (section 3.2) and lastly, we discuss the theoretical concepts concerning Image Registration (section 3.3).

3.1 Tango API

Tango is a new technology developed by Google Inc. to present smartphone devices with the ability to understand their position relative to the world around them. Tango consists of three core technologies: Motion Tracking, Area Learning and Depth Perception. Motion Tracking enables Tango devices to track their own movement and orientation through 3D space, Area Learning allows the devices remember to the locations previously seen and further use them to improve the accuracy of Motion Tracking. Finally, through Depth Perception Tango devices can understand the shape of their surroundings. [Source: `https://developers.google.com/tango/overview/concepts`]

For our application, we only need Depth Perception to calculate focal planes based on the perceived depth data and therefore we will take a closer look at that. Manufacturers of Tango devices can choose among common depth technologies including Structured Light, Time of Flight, and Stereo to implement Depth Perception. [Source: `https://developers.google.com/tango/overview/depth-perception`] By observing the sensors, we could figure out that the Lenovo Phab 2 Pro uses infra-red Light to perceive depth. However, none of the online available resources like manuals and technical specifications provide the information if Structured Light or Time of Flight as Infra-Red measurement technology are used. As both technologies have similar advantages and limitations (we will talk about them in section 5.1), knowing which of the two is used is of little relevance to us.

3.2 Focal Planes and Depth of Field

This chapter deals with the theoretical concept of focal planes and depth of field. By setting a focal length which changes the distance between focus lens and image sensor, we focus at a certain distance d_F called focus distance (Figure 3.1). We call the plane at focus distance and parallel to the image sensor, focal plane and all objects on this focal plane are therefore in-focus, because a point of these objects is projected onto a point on the image sensor.

If an object is closer or further away than the focal plane, the image formed on the sensor of an object's point is not a point, but a blurred circle of diameter c called circle of confusion. The diameter of the largest circle of confusion, which does not result in unacceptable blurring of the image, determines the closest/furthest object distance D_n/D_f from the camera lens, for which the image meets the stated condition. The distance range between D_n and D_f is called depth of field (Figure 3.2). Therefore all objects in the range of the depth of field show acceptable sharpness. [Gre50] The stated distances D_n and D_f can be calculated by the following formulas 3.1 and 3.2 according to Greenleaf [Gre50]:

$$D_n = \frac{s(H-f)}{H+s-2f} \tag{3.1}$$

$$D_f = \frac{s(H-f)}{H-s} \tag{3.2}$$

The hyperfocal distance H is the minimum distance where the depth of field reaches infinity and has the value $0.431m$ on the Lenovo Phab 2 Pro. s in the formulas is the distance between the lens and the objects and f is the lens focal length. In case of the Lenovo Phab 2 Pro, f has the value $3mm$ according to `http://www.devicespecifications.com/en/model/444a3c89`. Hence, by setting s to the hyperfocal distance H we get $D_n = 0.216m$ and $D_f = Infinity$, which is already almost the whole possible focus distance range [$0.1m$ - $Infinity$] of the Lenovo Phab 2 Pro. Nevertheless, we are not satisfied with this and want to see if we can achieve better results with our approach than we would get at the border of the depth of field range.

3.3 Image Registration - Alignment Methods and Motion Models

As the movement of the focal lens changes the field of view and therefore the section that can be seen of the scene, we need to register the images using a common reference image. The two terms image registration and image alignment are used interchangeably in the following. While the term image registration is typically used for the process of transforming different images into a common coordinate system, image alignment

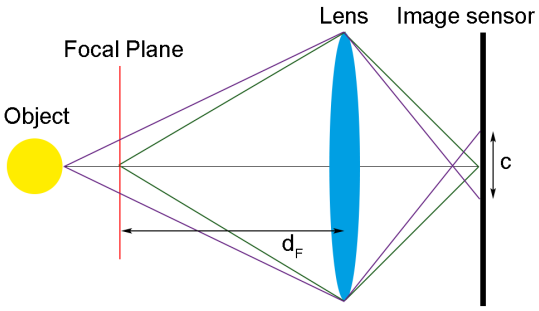


Figure 3.1: Abstract representation of the focus concept based on the thin lens model. Objects exactly on a focal place are the sharpest due to their projection onto exactly one point on the image sensor. A point on an object before or after a focal plane is scattered onto a circle consisting of multiple pixel with a diameter c called circle of confusion [according to [Gre50]]. This scattering of color information of a single point onto multiple pixels causes what we see as blur.

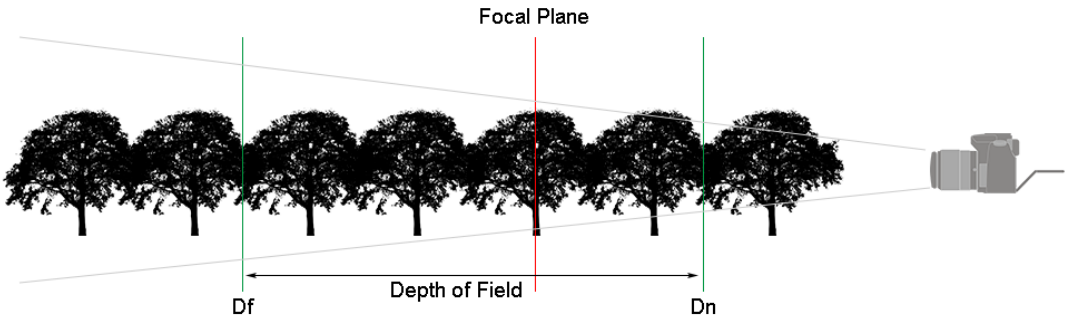


Figure 3.2: Depth of Field: In an image taken with focus at the focal plane, objects from D_n to D_f are acceptable sharp.

is used for the alignment of an image based on a reference image. We will talk about the implementation details of image registration in section 4.5, but at first we need to talk about the different alignment algorithms and motion models and which of those are suited for our purpose.

There are two main alignment algorithms, namely direct (pixel-based) and feature-based alignment. Direct alignment works by directly minimizing pixel-to-pixel dissimilarities while feature-based approaches work by extracting features like corners or edges and matching them. Feature-based approaches come with the advantage of being more robust against scene movement. They also function successfully with images that only partly overlap as in panoramas, however, feature-based alignments are less precise than direct alignment. Direct alignment uses an error metric to compare images on a pixel-based

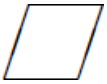
Name	Transformation	Preserves	Icon
Translation	Translation	Orientation + ...	
Rigid (Euclidean)	... + Rotation	Lengths + ...	
Similarity	... + Scale	Angles + ...	
Affine	... + Affine	Parallelism + ...	
Projective	... + Perspective	Streight lines	

Table 3.1: 2D Motion Models used for image alignment. [Sze04]

level. In direct image alignment, one image is usually fixed and the other is moving around while being compared to the fixed image. The simplest technique is to try all possible alignments but this is to slow in practice and therefore image pyramids are used to speed up the process. In order to also get sub-pixel precision, incremental methods based on a Taylor series expansion of the image function are often used. [Sze04] We use direct alignment because we need at least pixel-precise alignment since we want to stitch the aligned images pixel-wise together.

Furthermore, we also need to declare the 2D (planar) motion model necessary for our purpose. According to Szeliski [Sze04], there exist the following motion models: Translate, Rigid, Similarity, Affine and Projective (Table 3.1). Translation just covers a movement of the camera in the direction of the image x- and y-axis. Rigid can also handle a rotation of the image and additionally Similarity takes into account the scaling of the moving image. Affine considers linear distortion as well and Projective can even handle perspective transformations. [Sze04]

Since the movement of the focal lens causes small changes in the horizontal and vertical viewing angle of the camera (field of view), the image section of the images focused at different focal distances might not be the same. This causes the following observed effect: In two images taken on a tripod from the same position with just a different focus distance were in the outer approximate 12/22 pixels of each side (long/short side) details of the scene visible in the image taken at a focus distance of $7.2m$, that were not visible in the image taken at a focus distance of $0.4m$. This is the reason why we at least need Similarity as Motion Model for aligning the images in our application to handle this scaling effect.

CHAPTER 4

Implementation

4.1 Overview

Our proposed approach was implemented as a pipeline consisting of multiple stages (Figure 4.1). The non-trivial stages will be explained in the following sections. A part of the pipeline was implemented in Java for Android and the rest in Matlab.

1. **Depth Clustering:** The constantly received depth cloud from the Tango API is clustered into clusters based on the point's depth (z-distance) for image capture at the centroid distances.

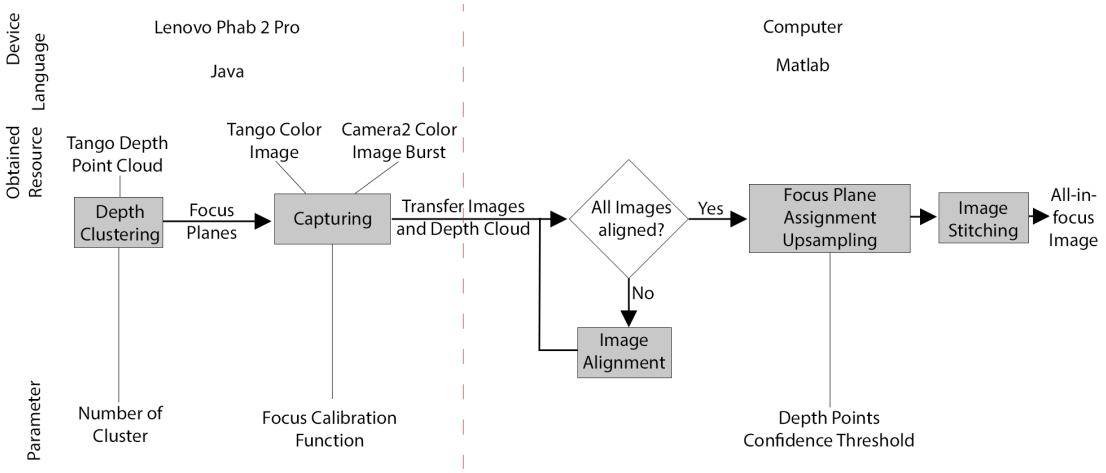


Figure 4.1: Processing Pipeline

2. **Capturing:** After saving the last transformed and clustered depth cloud, the next color image from the Tango API is saved. Afterwards, the camera accessing API is switched from the Tango API to the Camera2 API due to the fact that Tango does not support manual focus to date. Next, the image burst with the cluster centroid distances as focus distances is captured and stored. We have to save the color image of the Tango API to know the field of view of the depth cloud, those distortion parameters were used to transfer the depth points from depth space into color space. The transformation is necessary because both cameras have a different coordinate system due to their different location on the smartphone.

3. **Image Alignment:** As the movement of the focal lens also changes the field of view, we have to align the Camera2 API's images to a common reference image. We took the Tango API's color image as reference image so that the color images would match with the transformed depth points. This procedure is conducted for every Camera2 API's color image.

4. **Focal Plane Assignment Upsampling:** The focal plane assignments are the cluster assignments from the depth clustering stage in our approach and can be viewed as labels. Nevertheless, we will call them focal plane assignments from now on to keep it more general. As we receive a spare depth cloud with average 20,000 depth point depending on the scene, we have to upsample the focal plane assignments of the depth cloud to the resolution of the color images (5312x2988).

5. **Image Stitching:** Lastly, we stitch the images from the Camera2 image burst together based on the upsampled focal plane assignment.

4.2 Depth Clustering

To consider the scene's geometry and to put the scene's objects best in focus, we have to cluster the depth points based on their depth value to determine the best focal planes for the image capturing.

In order to select the most relevant focal planes for the image acquisition, k-Means clustering is applied to obtain normal distribution shaped clusters at best. In most cases the image scene consists of nearly planar objects (e.g. people, tables, walls) represented by a set of depth points of similar z-distance to the image plane which we want to have in the best focus. Therefore, we would see bumps representing those objects we want to have in focus in the z-histogram of detected depth points (Figure 4.2) and we want these bumps to be represented by exactly one single cluster in the best-case scenario. In cases of a flat scene (e.g. capture of a wall from the side like in figure 4.3), we can at best achieve non-linear distributed clusters for the fact that the depth of field has a non-linear behaviour like also Hasinoff et al. [HK11] found out.

This means that the depth of field is shorter when close to the camera and for that reason a higher amount of images with focus distance close to the camera and just a few to

Cluster Identifier	1	2	3	4	5	6	7
Initial Cluster Centroid Distance	0.5m	0.59m	0.71m	0.89m	1.2m	1.85m	4.0m

Table 4.1: Example for non-linear k-Means clustering: The mentioned distances are the calculated initial seed points for the cluster centroids for 7 clusters using formula 4.1 for depth points in the range [0.5m, 4.0m].

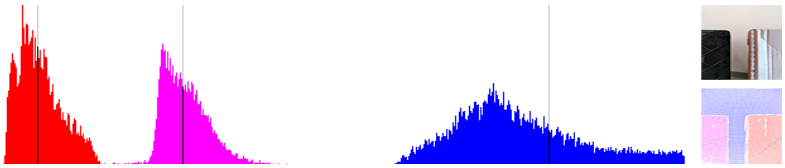


Figure 4.2: Example of k-Means Clustering applied on z-Values of Depth Points. The black bins with gray fade out represent the cluster centroids. Adjacent equally coloured bins represent the depth points assigned to the clusters. The scale is linear and goes from left to right from minimum to maximum z value.

cover the background are necessary to have the whole scene in-focus. We achieve this by setting non-linear initial cluster centroids and let them move in the direction of the bumps until the sum of changes in the centroid positions falls under a threshold. For us, the value 0.1m resulted in an acceptable runtime with minimal difference to full converge.

The initial seed points are determined by formula 4.1 that gives a non-linear distribution of the seed points (Table 4.1).

Also a non-linear metric is applied for the assignment of each depth point to its nearest cluster centroid, the calculation of the new cluster centroids and the summation of the changes. Formula 4.2 serves as an example for the distance metric between a depth point and a cluster centroid. The number of clusters can be set by the user, for the theses we determined the use of 7 clusters yields in good results and will mainly use this number of clusters, which is low enough to not run into the risk of an image processing memory overflow, that results in non-reception of images and/or green filled images.

Now that we have determined the photo acquisition focal planes, where we will capture the burst of images and afterwards extract the sharpest representative for each pixel of the resulting image from the burst. But first, we have to up-sample the depth point cloud to the resolution of the resulting image, at which we take a closer look in the next section.

$$seedpoint_i = \frac{1}{\frac{1}{maxZ} + \frac{numClusters-i}{numCluster-1} * (\frac{1}{minZ} - \frac{1}{maxZ})} \mid i \in [1, numClusters] \qquad (4.1)$$

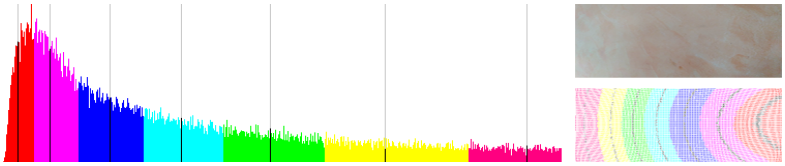


Figure 4.3: Example of k-Means Clustering applied on the depth cloud of a wall captured from a 45 degree angle. Although the wall is flat the histogram is not due to the linear axis, but still we see the monotonic of the curve without significant bumps like in the previous figure 4.2.

$$distance_{p-c} = \frac{1}{|\frac{1}{z_p} - \frac{1}{z_c}|} \,|\, p \in Points, c \in Clusters \tag{4.2}$$

4.3 Calibration

Although the Camera2 API returned us the manual focus of our Lenovo Phab 2 Pro device would be calibrated, we stumbled across blurry results despite correctly found focal planes by k-Means clustering. According to the API's documentation, the relationship between focus distance and focal length is $focallength = \frac{1}{focusdistance}$ which seems to be incorrect. Therefore, we decided to verify and correct the calibration of the device.

At first, we conducted a binary search on the full interval [0, 10] of the focal length for a few object distances based on a sharpness maximization (Table 4.2). As metric we used the sum of 4-neighbourhood Laplace of the central 101x101 pixels aligned to a normal plane containing only a single black letter on white background at a physically measured distance. There is no particular reason for this size. We simply empirically determined that it would be big enough to fit our letters and small enough to have an acceptable runtime for the focal measurement calculation. In a sharper image, the intensity changes from one pixel to a neighbouring pixel at the edges of the letter are higher, therefore resulting in a higher metric result.

We repeated the routine 3 times for each of the 7 distances to make sure it converges correctly (Figure 4.4) and we used this calibration with non-linear interpolation in between as a starting point for the refinement. From now on, we use the cluster centroids distance of the k-Means algorithm determined for one cluster for the central 101x101 pixel as distance measurement instead of the physical measurement since the same distance measurement is used for the focal plane calculation and therefore free of any hard to determine offsets between the physical and the depth sensors measurement.

In the next step, we refined the results by measuring the sharpness of more points and visually comparing the calibration with 3 images before and after in a 0.1 focal length units' step size. The optimum of these 7 resulting images was taken as refined calibration.

Iteration	Focal length	Sharpness	Iteration	Focal length	Sharpness
1	0.0	51886	6	1.875	102586
2	10.0	23985	7	1.5625	107437
3	5.0	31355	8	1.71875	104511
4	2.5	83733	9	1.640625	107806
5	1.25	97041	10	1.6015625	104239

Table 4.2: Example of the binary search for the sharpness optimum for the focal length calibration. As you can see, the sharpest image is not always found in the last iteration. As result, we take the focal length with the highest sharpness value which, in this case, is in iteration number 9. The amount of iterations of the algorithm is hardware-limited to 9-12 iteration depending on the image content due the size limit of the devices' image processor.

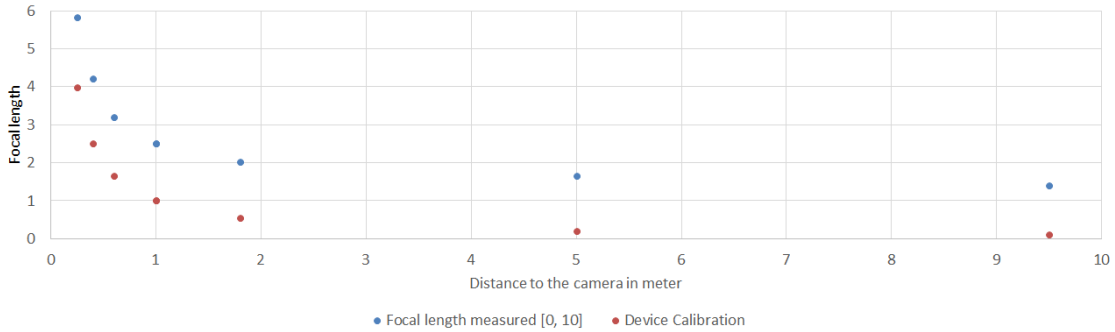


Figure 4.4: Results of the binary search for the focal length calibration for 7 distance (median value of 3 passes). As you can see there seams to be an offset of 1 to 2 focal length units of the actual focal length to the pre-calibrated focal lengths in the device.

We repeated this routine for the bulk of the distances and carried out a power regression with the results of both iterations combined (Figure 4.5).

From now on, we use the resulting formula 4.3 from the power regression for the mapping of focus distance to focal length.

$$focallength = 2.8974 * focusdistance^{-0.339} \tag{4.3}$$

You can see our calibration compared to the calibration of the device in figure 4.6 and a comparison of our calibration with 6 neighbouring images in the range of $[-0.3, +0.3]$ in figure 4.7.

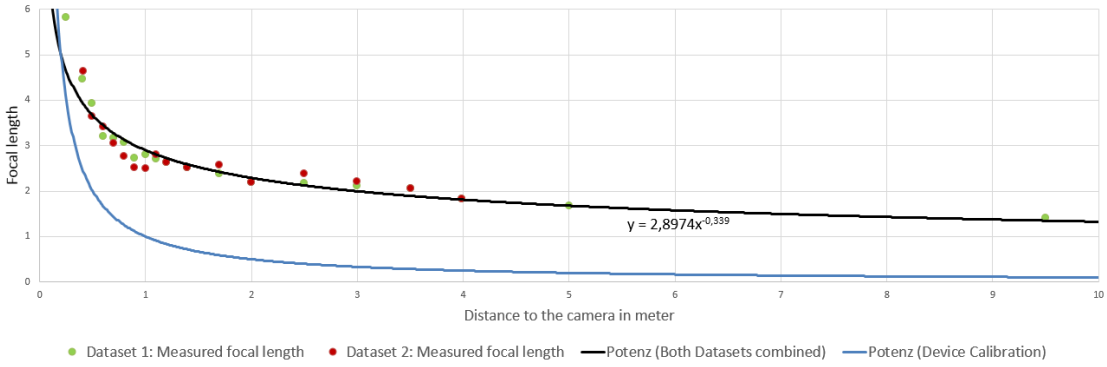


Figure 4.5: Power regression of both datasets combined (black curve) compared to the device calibration (blue curve).

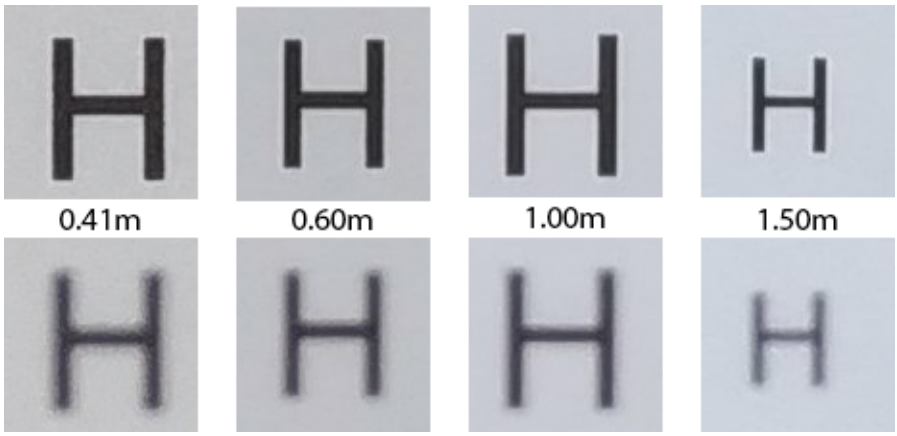


Figure 4.6: Visual comparison of our calibration (first row) with the device calibration (second row) for 4 exemplary distances.

4.4 Capturing

To this day, the Tango API does not provide the feature of manual focus and only one higher level camera framework API is permitted by Android's camera hardware abstraction layer (HAL) to connect to the cameras at the same time. Therefore, we need the Tango API to capture the depth cloud but also the Camera2 API to capture the image burst at different focus distances. As we need the Tango API to perform the Depth Clustering, we start by capturing and saving the resources we receive from the Tango API.

First of all, we start by capturing the Tango API's color image, upsample it to the same resolution of the Camera2 API's images (1920x1080 → 5312x2988) and save it to the memory. Then we serialize the transformed depth cloud into a file. We proceed by

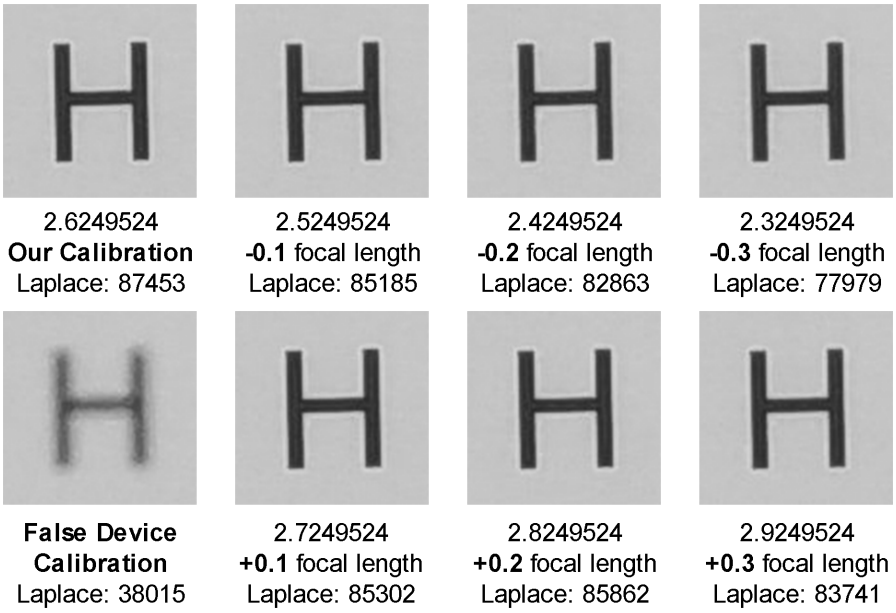


Figure 4.7: Visual comparison of our calibration and 3 images before and after in 0.1 focal length units. We have chosen the maximum of the sum of 4-neighbourhood Laplace over all shown 101x101 pixels as calibration criteria. As becomes apparent, the Laplace value and the visual result do not change much in the range of $[-0.3, +0.3]$ Because of that, we choose to use a power regression as shown in figure 4.5 to have a monotonic function approximating the focal length, since the optical differences hardly vary and the maximum of Laplacian does not result in a monotonic step function as you can see in figure 4.5.

switching from Tango API to Camera2 API to capture the color image burst. The cluster centroid distances of the depth clustering stage are used as focus distances for the burst capture.

As the number of clusters is the same as the number of captured images by the burst, we have to take care of its value in order to not violate hardware limits. The burst images are cached and processed in a image sensor processor with a memory size limitation whose size we could not figure out in the data sheet of the device. According to personal experience, 9 to 12 images can fit into its memory depending on the JPEG compression which depends on the image content. The challenging part about this is that we do not know the limitation in advance and only notice a memory overflow because we receive fewer images than asked for. According to this, the best way is to set the amount of clusters ≤ 9, since this is the minimum amount we have always received. The images are saved to the memory with their focus distance coded into the filename for the stitching stage.

4.5 Image Registration

In this section we describe our usage of image registration, which is necessary in order to have all images show the same part of the scene. Regarding image registration, we did not develop a proper implementation but rather used a demo application and added a small modification to fit into our pipeline because the purpose of this thesis is not to show new methods of image registration. The aim of this section is to describe the parameters we used to get our results.

First of all, we will describe the alternatives we tried under Android with which we were not successful and which forced us to change to an implementation in Matlab. Our first attempt was to use OpenCV (github.com/opencv) with the registration module in the extra modules package (github.com/opencv/opencv_contrib/tree/master/modules/reg) because its runtime is expected to be good due to its implementation in C++. We spent several weeks trying to build OpenCV with the registration module for the process architecture (arm64-v8a) of the Lenovo Phab Pro 2 but failed.

We managed to build it for a different process architecture (armeabi-v7a) but as far as we could guess from the source code, the process architecture of the Lenovo Phab Pro 2 had not yet been taken into account in the development of the build script. The modification of the build script in order to support our process architecture was far beyond the purpose of this thesis and so we searched for an alternative.

There also exist a full java implementation of image registration called BoofCV (boofcv.org), but the fact that it uses feature-based alignment as alignment method is problematic - in contrast to OpenCV which uses direct (pixel-based) alignment - because it is not suited for our purpose.

In the end, we found an acceptable alternative in Matlab's imregister method. We used the image registration application of Brett Shoelson (mathworks.com/matlabcentral/fileexchange/34510-image-registration-app), which is just a user interface environment for the Matlab's imregister method. We modified it a little to fit into our pipeline and have our empirically determined default parameters pre-set.

As described in section 3.3, we use Similarity as motion model, because we usually have to deal with scale transformation due to the movement of the focal lens. Translation and rotation can also pose a minimal issue depending on whether a good tripod and smartphone grip is used or not.

We applied "Difference" as visualization method because we found it is most suited to see minimal errors in the alignment. We chose to use the registration type "Monomodal" due to the fact, that the images were captured from the same device and due to auto exposure with similar brightness. "Multimodal" would be more flexible and work for different lightning conditions, but in our case its usage led to an extensive increase of incorrectly aligned images, so we remained with "Monomodal". We achieved the best results with 2 Pyramid Levels because a higher amount often led to misalignment due to false converging at a higher pyramid level.

At last, the optimizer/metric parameters worked well for us in the standard configuration which is: Gradient Magnitude Tolerance=0.0001, Maximum Iterations=100, Relaxation Factor=0.5, Minimum Step Length=0.00001 and Maximum Step Length=0.0625.

The image registration routine typically had a runtime between 30 seconds and 5 minutes per image depending on the scene and a success rate of approximately 95%. The remaining 5 percent exist due to scenes with well-known image alignment issues such as repeating structures (e.g. stairs) or movement (e.g. grass) in the scene and can therefore be avoided.

4.6 Focal Plane Assignment Upsampling

As we typically receive only around 20,000 depth points from the Tango API but have to assign 15,872,256 (5312x2988) pixels to a focus layer, we need to upsample the focal plane assignment to the density of the pixel information, keep in mind that the focal plane assignments in our approach are the cluster assignments from the depth clustering stage. However, we need to do some pre-processing first.

The depth image in the clustering stage still showed some outliers with depth values at positions in the image which are not possible, e.g. a depth value of $0.3m$ on the wall in the background at approximately $4m$. Due to their small amount, however, we decided to keep them in the point cloud for the clustering and exclude them in the post-processing pipeline with a threshold based on their confidence value, since it is easier to re-run the post-processing pipeline with a different threshold than recapturing the images.

The depth points obtained by the Tango API also include a confidence value, that tells us how precisely the depth point could be detected by the depth sensor. We use a threshold for the confidence value to decide which depth points to include for the following steps and decided empirically to use the value 0.5. This value reduces the amount of outliers sufficiently and at the same time maintains enough depth points in the background to upsample it correctly (first row in figure 4.8). But there are still too many correct depth points removed in the background and some outliers remain in the foreground. Hence, we raised the threshold to 0.8 for the nearest point and let it linearly fall to zero for the furthest point (second row in figure 4.8).

Since we now have successfully removed outlier, we can upsample the focal plane assignments. In order to do this, we use the nearest neighbour method on the focal plane assignments, because we empirically found out that real edges like the edge of a table are most likely to be in the middle of two depth points of different clusters. We decided to use the closest neighbour as metric to draw the separation line in the middle of 2 points of different focal plane assignments, because we do not have any information about the empty space between the two points in order to come to a better decision on the location of the separation border.

Furthermore, we tried linear interpolation of the depth values and focal plane assignment to the nearest focal plane using non-linear distance but in most cases, it resulted in worse overlap with real edges in the scene than with simple nearest neighbour of cluster

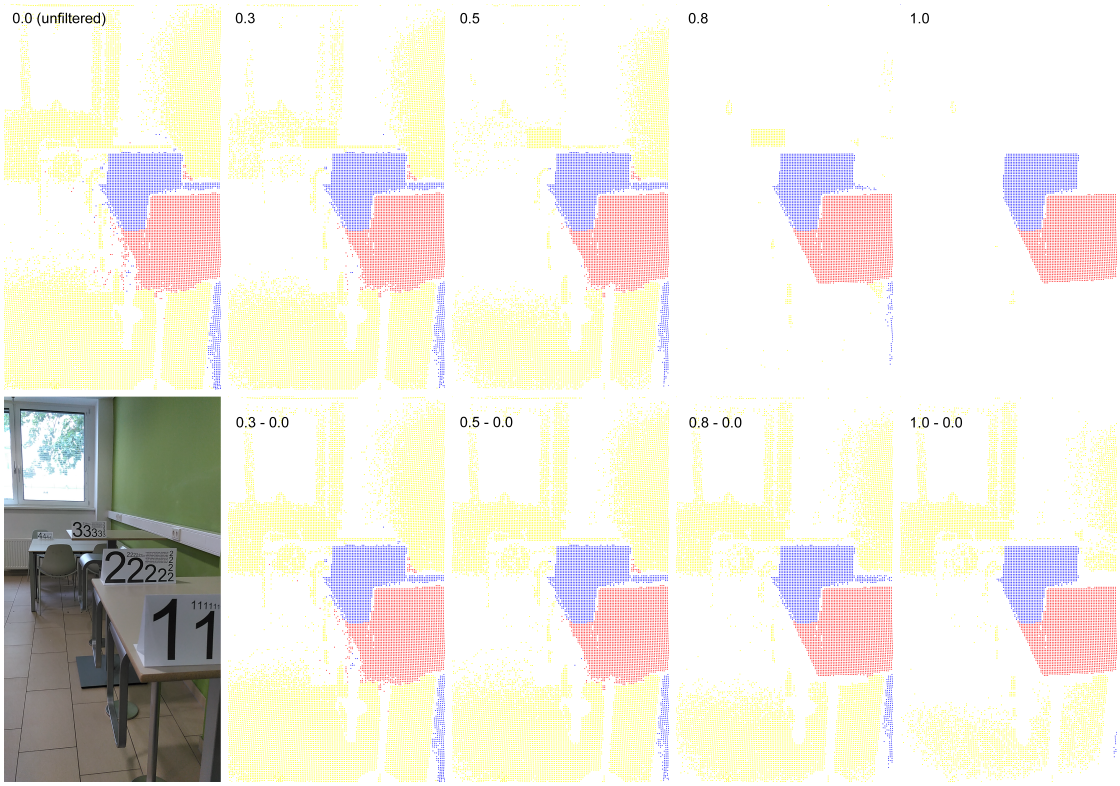


Figure 4.8: Depth points filtering based on a confidence value threshold: The image 0.0 shows the detected depth points without filtering and 1.0 shows just depth points those the depth sensor has detected with full confidence. We found that 0.5 is a good compromise between the amount outliers and the amount of the depth points in the background (first row). With letting this value linearly fall down to zero from the front to the back, we could raise this value to 0.8 (second row). So our finally used threshold is shown in the image with the label $0.8 - 0.0$.

assignments. Moreover, we tried the same procedure using nearest neighbour of the depth values with non-linear assignment again, but this provided the same results as directly applying nearest neighbour on the cluster assignments. Non-linear distances are applied so that the depth of field has a non-linear behavior.

Upsampling the cluster assignments directly instead of upsampling the depth data poses the advantage, that we reuse given calculations of the clustering state and do not need to repeat this assignment of the nearest neighbour upsampled depth points to the nearest focal plane.

Initially, we tried using the upsampleImageNearestNeighbor method of the Tango Support Class, but as it is still in experimental state it only worked partially in a few cases and did not work at all resulting in an empty image most of the time. Therefore we had to

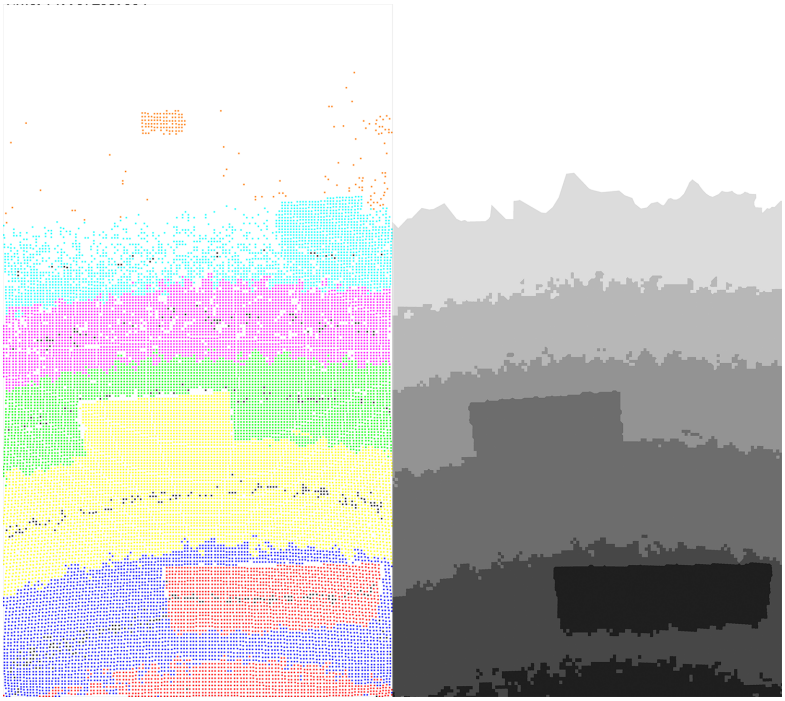


Figure 4.9: Focal Plane Assignment Upsampling with the nearest neighbour method. Based on the grid-like scanning of Tango's depth laser the assignment result has also pixel-like or stair-like edges between the regions, except in the back region where tango could only locate a few points. The black points in the left image visualize depth points nearest to a focal plane.

search for an alternative.

Self implementation in Java was far too slow, but we found an acceptable alternative in the griddata method of Matlab to upsample the focal plane assignments to the resolution of the color pixels using the nearest neighbour method. There are ways to run Matlab code on Android to integrate it into the rest of the application and receive the resulting image already on the smartphone but this is of no importance for this thesis and therefore, all processing steps requiring Matlab are performed on a computer. The result of this upsampling is shown in figure 4.9.

4.7 Image Stitching

Now that we have registered all images to the same viewport and the focal plane assignment has been upsampled to the same resolution, only the stitching of the result remains.

We use trivial pixel-wise assignment of the color values based on the focal plane assignment

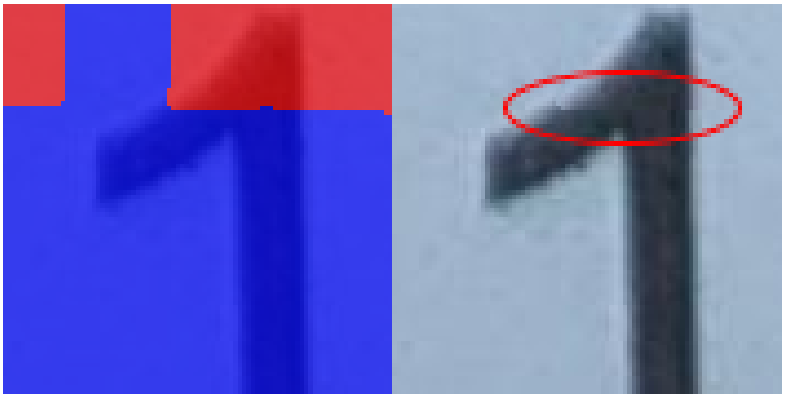


Figure 4.10: Visual evaluation of the stitched result. The left image is color-coded to show which regions come from the same image source. The marked areas show visual jumps of 1-2 pixels at the edge of the letters due to small image registration errors.

without any handling of seam artifacts.

There are several approaches like Birsak et al. [BMAW13] dealing with differences in color intensities due to different lighting conditions. Birsak et al. uses a leveling function which deals with the discontinuity of color intensity due to jumps of the color intensity at adjacent faces of different labels. Each label stands for a specific image source. To ensure smooth transitions, the leveling function has negative jumps with the same value as the jumps of the original function and minimal magnitude of the gradients elsewhere. [BMAW13]

As our pictures are captured from the same location and angle just seconds from each other the lighting conditions are almost the same. However, there remain some rarely occurring and barely noticeable visual jumps due to remaining small image alignment errors (Figure 4.10). Blending might help here but one runs at risk to reintroduce blur and so we left as it was because it is a rarely occurring phenomenon. Since we took account of the non-linearity of the depth of field in the focal plane assignment, there are hardly no differences in the remaining amount of blur noticeable in adjacent regions of a border between two focal plane regions. For results of the described pipeline see the following chapter 5.

CHAPTER 5

Evaluation

First of all, we want to review the objective of this work. As Google's new Tango API with the Lenovo Phab Pro 2 as the first device supporting it opens a wide range of new possibilities, the objective of this work is to evaluate the API's usability for generating all-in-focus images.

In contrary to single-image approaches, the resulting average sharpness highly depends on the used amount of images, but there are also some limitations in the amount of images due to the hardware as we discussed in section 4.2 and 4.4. So we take advantage of the depth of field to also have areas in the image in focus that do not lie directly on a focal plane. But as the depth of field on smartphones due to the small lenses is already wide, a focus setting at hyperfocal distances (in case of the Lenovo Phab 2 Pro: 0.431m) nearly covers the whole distance range with just one image.

So the results we obtain are usually sharper at every pixel of the image than it would be at the border of the depth of field range of a single image, because of overlapping depth of fields. Notice the difference in the following images between blur and sharp images (Figure 5.1), especially at zoomed in details.

In the following chapters we will evaluate different aspects and limitations of the results, but also of the limitations of the capturing and clustering of the depth cloud.

5.1 Capturing

As the Lenovo Phab 2 Pro uses infra-red light to measure depth there are some limitations in capturing the depth cloud. According to Google Inc., Infra-red systems like Structured-light or Time-of-flight have the benefit of working particability well indoors, in low light and where there is no texture on the wall. On the other hand infra-red light is less reliable in environments with light sources with high amount of infra-red like direct sunlight or

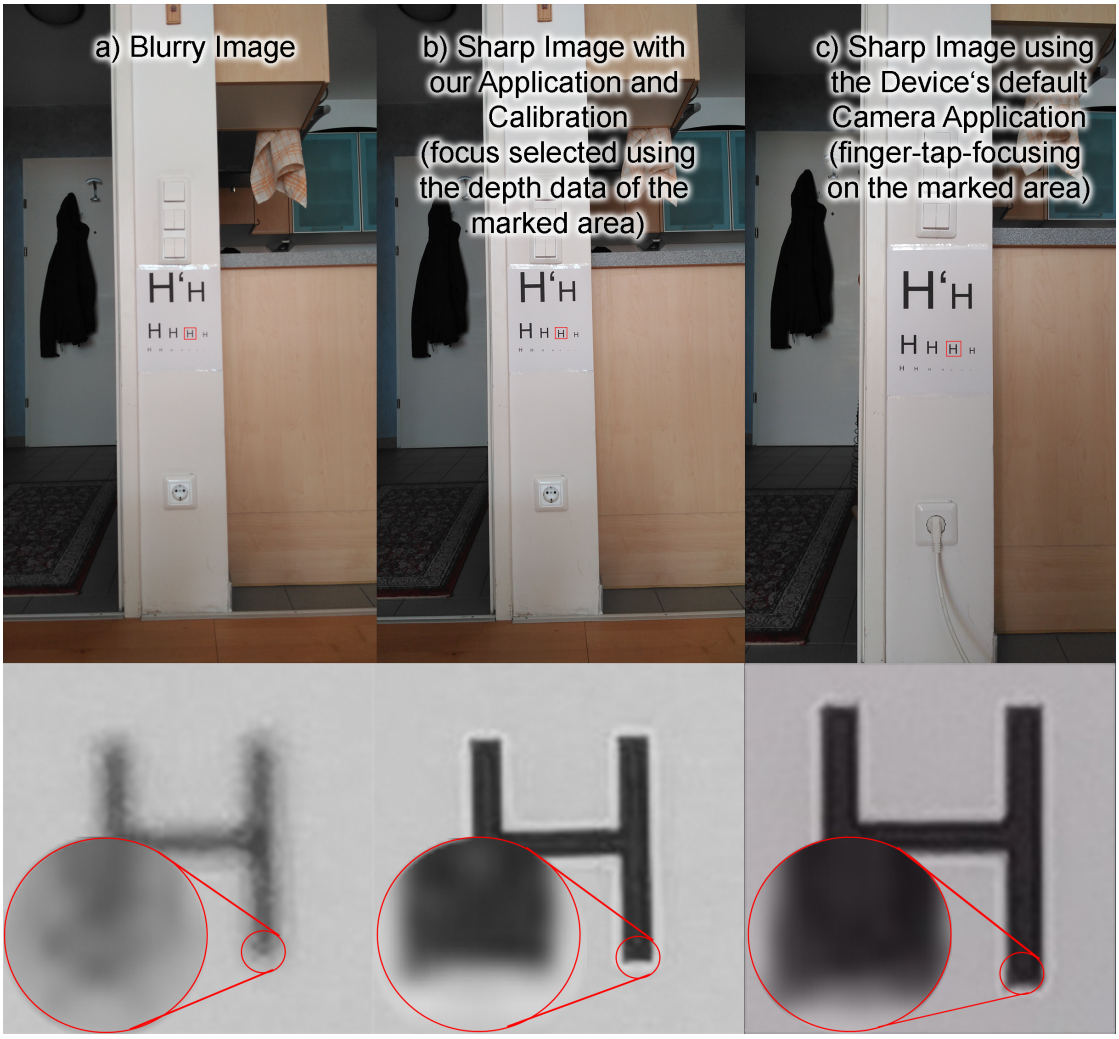


Figure 5.1: Comparison between a blurry (a) and a sharp image (b) using our application. Just comparing the upper image one could say that all 3 images look sharp, but if you zoom in into the rectangle area you see the blur (a). Notice that in the round detail area of the sharp image using our application (b) you can still see that the letter has no sharp edges but comparing it with the sharp image using the device's default camera application (c) this is more or less the sharpest image quality that could be obtained with the device's camera. Also take care that the excerpt of the scene and the exposure time of the device's default camera application (c) was different.

incandescent bulbs and on very dark surfaces who tend to absorb infra-red light, which causes a reduction of the amount of depth data acquired.

The sensors of Tango devices are optimized to scan room-size and static environments which makes it well suited for detecting surfaces, large objects like walls, doors, tables and chairs. Current devices are designed to work best at moderate distances (0.5 to 4 meters). [Source: developers.google.com/tango/overview/depth-perception]

We could also observe the described problems and got depth data in a distance range from approximate $0.4m$ to $7.4m$ (Figure 5.2). Following from this, we empirically determined that good capturing conditions are indoor locations with good ambient light like incoming light from big windows on a cloudy day or not in sun direction orientated windows on a sunny day. Outdoor it needs to be a place in the shadow to at least get depth data in the near part of the scene.

5.2 Clustering

In figure 5.3 you can see how well our version of the kMeans algorithm works.

5.3 Results

In this section we compare the results and evaluate the usability of Tango for multi-focus image generation. Since the images are to big to notice any differences in the sharpness in full size, we zoom in to some details where the difference between the input images can be noticed. Our goal is to evaluate if the kMeans algorithm could calculate a sufficient sharp focal plane for a given detail and if the sharpest images was chosen out of all input images for the result of the detail based on the depth cloud.

The syntax of the following result comparison images (Figure **??** to **??**) is as follows:

First row from left to right: 1) Detail result image, 2) Detail color coded result image to show which pixel originates from which image of the second row, 3) Detail depth cloud image to show which depth points were detected from the depth sensor in this detail section, 4) a map showing the whole result image and its detail section from 1), 5) a map showing the whole color coded result image and its detail section from 2), 6) a map showing the whole depth cloud and its detail section from 3).

Second row: All input images from the nearest (left, red) to the farest (right) for the selected amount of clusters. In some comparisons not all input images are displayed as you can notice on the missing numbers in the numeration, but the extremes (first and last images) are always shown. The background color of the label is the same as in the color coded images and indicates the regions in which the image was used.

As shown in the images the sharpness of the images could be improved in each case compared to a single image taken advantage of the depth of field (Figure 5.5, 5.8). In those mentioned comparisons the first image was taken at approximately hyperfocal

Figure 5.2: (a) In the left image you can see the leak of depth data due to the absorption of the infra-red light on the black doors of the furniture. (b) Due to the high amount of infra-red in direct sunlight Tango could not obtain depth points in the back of the park scene. (c) In the right image you can see the near and far limitations of the Tango depth sensor. The closest detected points are at $0.4m$ and the farest at $7.4m$. Further, it can be noticed that the industrial floor in this room was not suited for depth perception due to high amount of reflection.

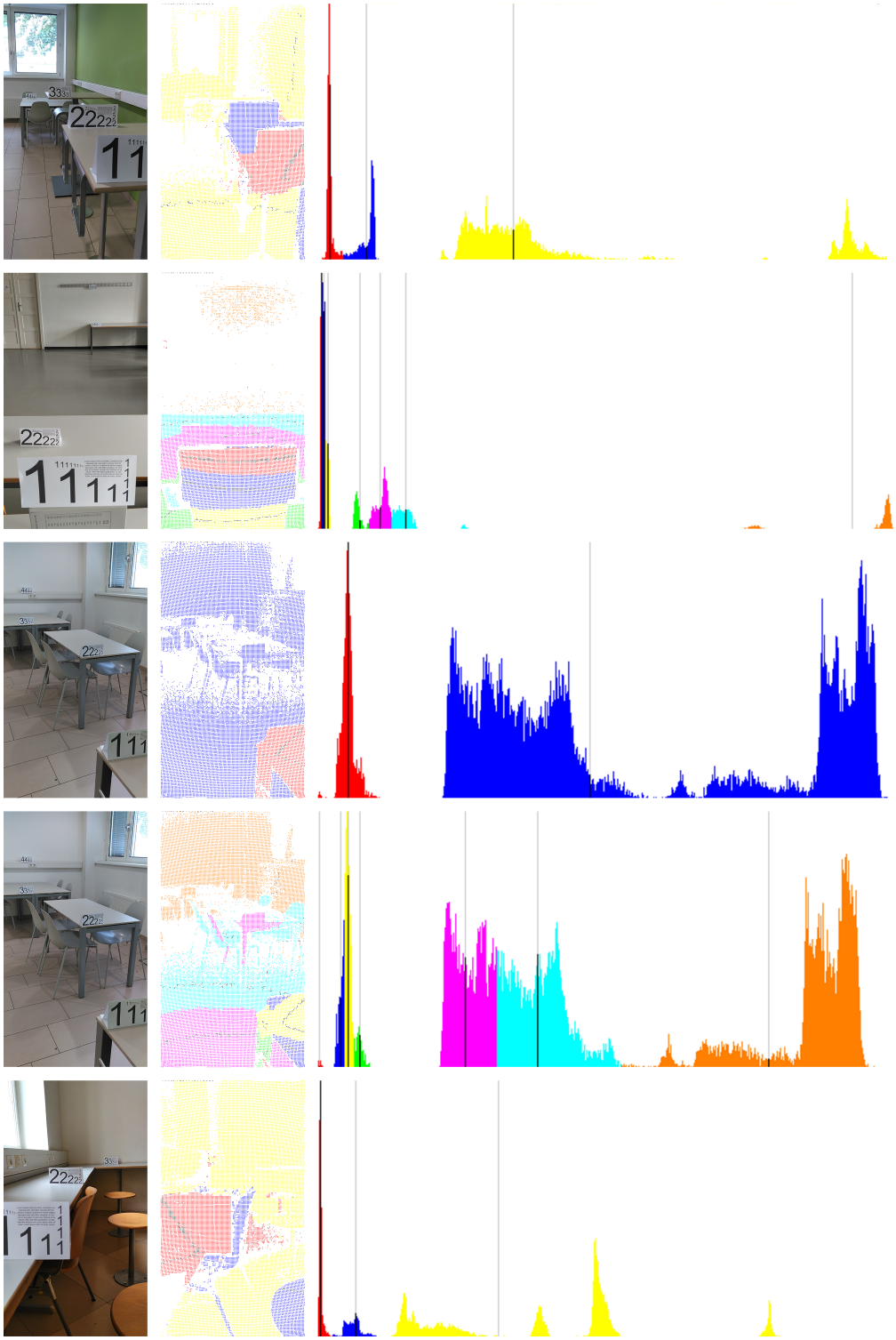


Figure 5.3: Examples for k-Means clustering: The horizontal axis in the histograms goes from minimum depth (left) to maximum depth (right) linearly. The colors represent different clusters and the vertical black/gray lines represent the distances of the cluster centroids. Image 3 and 4 are the same scene, but with different number of clusters. In most cases work 2-4 clusters well, but in image number 2 it had to be a higher number of clusters (7), so that the last cluster (orange) remains in the background. In image number 5 were in our opinion not the ideal cluster centroids found, since the last two centroids could be at the location of higher peaks behind them.

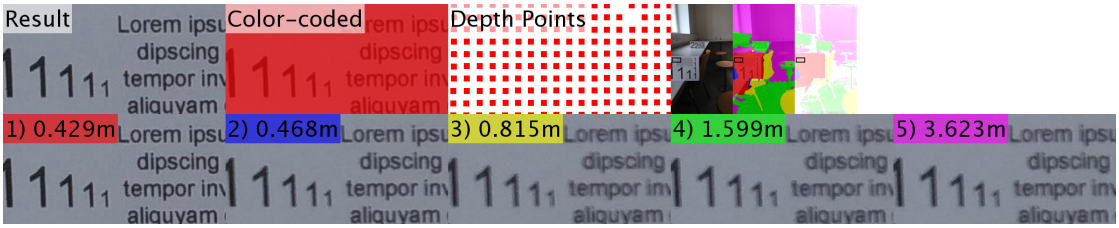


Figure 5.4: Scene Stool (front card): Correctly assigned image number 1 (red, 0.42898m) as the sharpest image for the displayed detail.

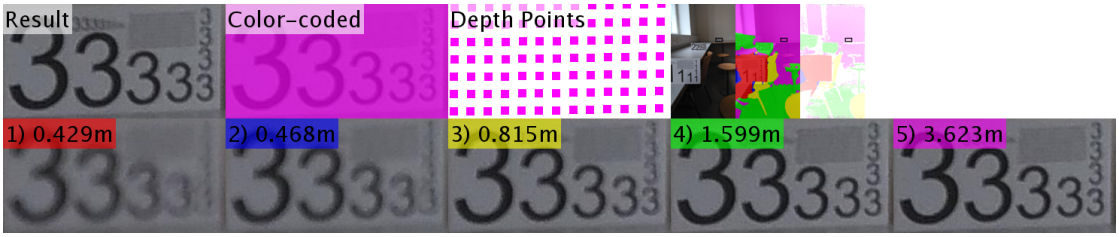


Figure 5.5: Scene Stool continuation (back card): Correctly assigned image number 5 (violet, 3.6233m) as the sharpest image for the displayed detail.

distance of the Lenovo Phab 2 Pro (0.431m) and therefore as a single image taking advantage of the depth of field, but still the visible detail of the background shows a significant amount of blur compared to our result. If the best focal planes could be found, highly depends on the detected depth cloud and the amount of clusters.

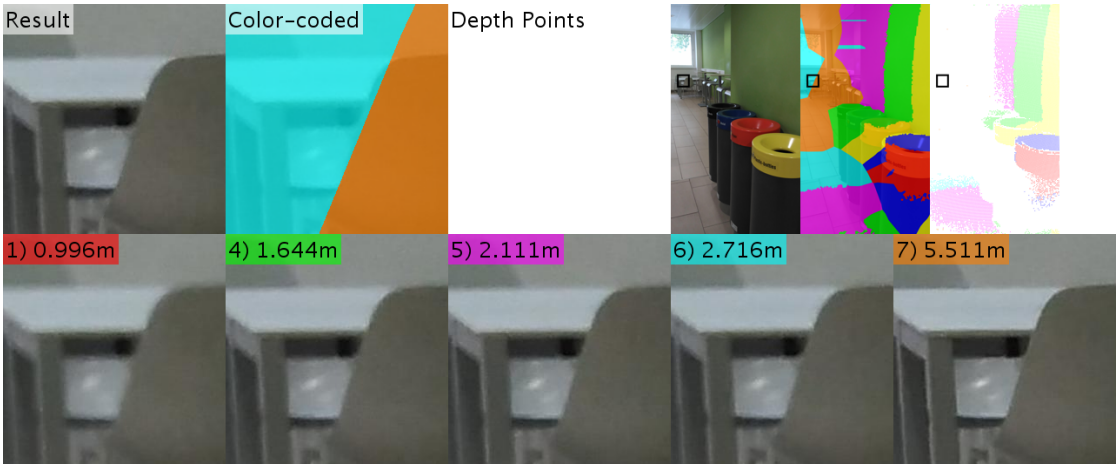


Figure 5.6: Scene Chairs: Due to wrong detected depth points at the window in the sparsely detected background, for the left chair image number 6 is used, although image number 7 would be more suited.

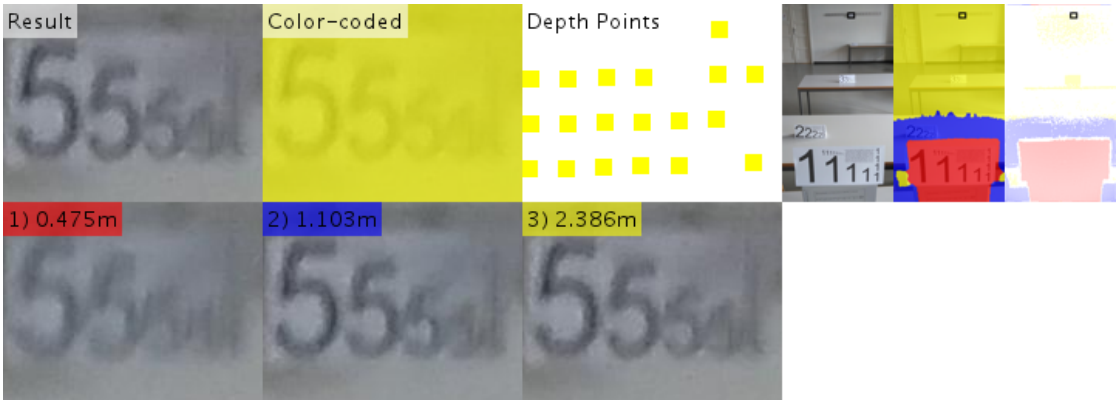


Figure 5.7: For this deep room were 3 clusters not enough to get the back sufficient sharp. Here are more clusters necessary, like shown in figure 5.8.

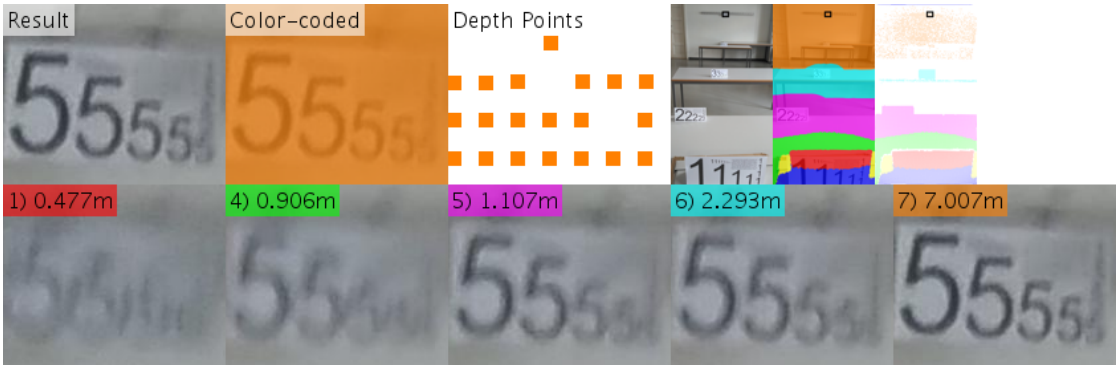


Figure 5.8: Sharper version of figure 5.7 with 7 clusters.

CHAPTER 6

Resume and Future Work

In summary, the goal was to evaluate the usability of the Tango API to generate all-in-focus images. As shown in the evaluation chapter, we could improve the sharpness of the image compared to a narrow single image. Although there some exceptions: In some cases the optimal image could not be found for a detail due to a too sparse depth cloud and for some cases it was necessary to use a higher amount of cluster than objects in the scene to cover the whole scene sufficiently due to not optimal converge of the kMeans clustering algorithm. Those problems are due to limitations of the depth sensor or respectively could be fixed by using a higher amount of clusters, since hardly any metric could led to a perfect converge for every possible scene.

But still a big issue is the high amount of time it takes to process those images, which makes it impractical for the daily usage. It takes approximately average 1 second to save the depth cloud and capture the Tango color image, 4 seconds to switch from Tango API to Camera2 API, 1-8 seconds to capture the image burst depending on the light conditions and amount of clusters, 3 minutes to register each image, 40 seconds to upsample the focal plane assignment and 40 seconds to stitch the images together. This sums up to a total run-time of average 15 minutes for a singe all-in-focus image depending on the amount of captured images. And due to the long capture time it is necessary to capture the images with a tripod and there may not be motion in the scene.

There some improvement for future work, that can be done concerning the run-time:

But as Hasinoff et al. [HK11] showed it is possible to capture multiple images with the same total depth of field under the time of a single image by magnifying the aperture diameter and lowering the exposure time. But this is not possible on the Lenovo Phab 2 Pro due to the fact, that the aperture is fixed to $f/2.2$. Concerning this, we have to hope, that smartphone manufactures soon use configurable apertures in their devices.

Further, it would be possible to capture the images directly with the Tango API if Google Inc. would add manual focus support to omit the time for API changing.

It would be also possible to pre-calibrate the scale of the image section compared to the reference image like Hasinoff et al. [HK11] did as a future work to omit the time for the image registration, but for this it needs a faster capturing to prevent motion between the captured images even without a tripod.

Although the Camera2 API returned, that the focus was calibrated, it was not as you could see in figure 4.4 to 4.7. Therefore, if this wrong calibration varies between devices, our re-calibration would not be transferable onto other devices.

A different approach for future work, that does not depend on the focus calibration would be a different focus assignment strategy, that clusters the depth cloud into x-y areas of depth points with similar depth, where the automatic focal mechanism of the camera could search for the right focal value. But due to our observation of the finger-tap focus in the default camera application, the camera of the Lenovo Phab 2 Pro uses a blur metric to search for the sharpest setting, which also takes approximately one second for each image and therefore would be slower than our approach.

In conclusion, we have shown that our approach works fine and outputs good results if the limitations of the depth sensor are not violated, but still there are many improvements necessary on hardware, Tango API and application side to make it practical for daily usage.

List of Figures

List of Tables

Glossary

deconvolution Image deconvolution is a computational technique that undoes the optical blurring to retrieve the true scene. Deconvolution, in its simplest implementation, can be carried out by a division between the captured image and the blurring kernel in the Fourier domain [SRSL17]. 4, 5

depth of field is the interval of distances in the scene $[d_1, d_2]$, whose blur diameter is below a maximum acceptable size c [HK11]. 1, 2, 5–8, 12, 20, 21, 25, 33

point spread function describes the response of an imaging system to a point source or point object [Source: Wikipedia]. 1, 4

ringing artifacts In digital image processing ringing artifacts are artifacts that appear as spurious signals near sharp transitions in a signal. They appear as bands or "ghosts" near edges. The term ringing is because the output signal oscillates similar to a bell after being struck [Source: Wikipedia]. 4

Bibliography

[BMAW13] M. Birsak, P. Musialski, M. Arikan, and M. Wimmer. Seamless texturing of archaeological data. In *2013 Digital Heritage International Congress (DigitalHeritage)*, volume 1, pages 265–272, Oct 2013.

[Gre50] Allen R. Greenleaf. *Photographic optics / Allen R. Greenleaf.* Macmillan New York, 1950.

[HK11] S. W. Hasinoff and K. N. Kutulakos. Light-efficient photography. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(11):2203–2214, Nov 2011.

[JPJ16] Seung-Won Jung, Jong Hyuk Park, and Young-Sik Jeong. All-in-focus and multi-focus color image reconstruction from a database of color and depth image pairs. *Multimedia Tools Appl.*, 75(23):15493–15507, December 2016.

[SRSL17] Basel Salahieh, Jeffrey J. Rodriguez, Sean Stetson, and Rongguang Liang. Single-image full-focus reconstruction using depth-based deconvolution. *Optical Engineering*, 56(4), 4 2017.

[Sze04] Rick Szeliski. Image alignment and stitching: A tutorial. Technical report, October 2004.

[ZMN12] Changyin Zhou, Daniel Miau, and Shree K Nayar. Focal sweep camera for space-time refocusing. *Technical Report, Department of Computer Science*, 2012.