Universität Karlsruhe (TH)
Fakultät für Informatik
Institut für Betriebs– und Dialogsysteme
Prof. Dr. H. Prautzsch

Reconstruction Using Structured Light

Hao Li
`hao@ira.uka.de`
`http://i33www.ira.uka.de/~hao`

Undergraduate Research Project
(Studienarbeit)
February 23, 2004

Supervisors: Prof. Dr. H. Prautzsch, Dipl. Inform. R. Straub

Statement

I hereby certify that I have completed the following work myself without the help of any resources other than those listed under references.

Karlsruhe February 23, 2004

Abstract

Extracting range data through active optical triangulation requires robust stripe edge detection w. r. t. the emitted pattern. On the other hand, due to ambiguous subpattern in the projection caused by the limitations of the windowed uniqueness property, corresponding a pattern consisting of multiple stripes, in attempts to reduce the number of input images, often leads to mislabeling. Moreover, high frequency shape and shading variations in the scanned object introduce significant noise in the resulting range map. We propose several extensions and an implementation of a subpixel-accurate shape acquisition system using color structured light. Assuming that stripes have a certain width, our edge detection criterion is achieved by restricting the number of consecutive edge pixels and exploiting the positive metric for multi-spectral edge detection. We fix the multi-pass dynamic programmed labeling by considering a background plane behind the object and masking it during the triangulation. The subpixel accuracy from one single input image can be obtained by approximating the gradient of the square local contrast with a piecewise linear curve. Finally, we present a noise reduction technique on the range map through meshing followed by a parametrized face orientation culling. Experimental results have demonstrated major improvements in terms of robustness against acquisition noise, shading variations and complex shapes in the multi-pass dynamic programming approach originally proposed by Zhang et al. [ZCS02].

Keywords

Photogrammetry, 3D scanning, range acquisition, structured light system, sub-stripe accuracy, optical triangulation, reverse engineering.

Contents

Chapter 1

Introduction

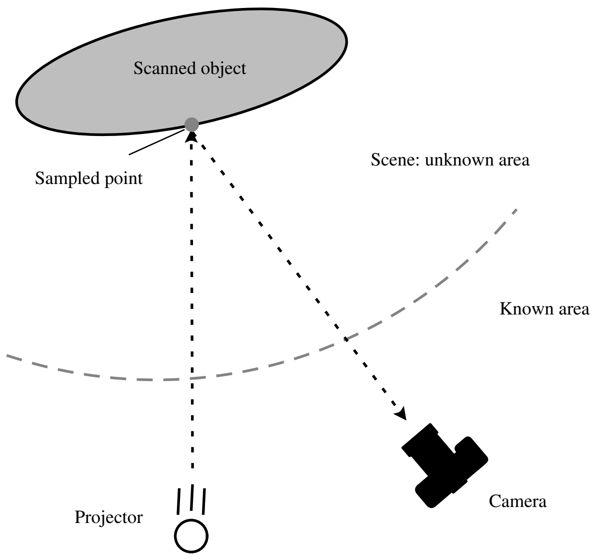


Figure 1.1: A scheme of an optical triangulation in a 2D plane. We differ between known and unknown area, whereas the known area comprise the calibrated projector and camera, and the unknown area is the scene with the target object. Knowing the camera's and projector's line of sight, we can extract the position of the sampled point in space by intersection.

Being able to reconstruct the shape of an object from a set of images has a tremendous impact on the field of computer graphics. As part of the CAD and entertainment industries, the modeling process of highly complex models, which requires a huge number of CG modelers and great artistry, could be reduced to a simple automated scanning process. Animators could benefit from extremely accurate models instantly acquired from the real world. Low cost range acquisition devices, which have attracted the manufacturing community, offer the possibility of remotely visualize NC fabricated products in 3D. They also accelerate the evolution of large

3D online catalogs that are becoming increasingly popular on the Internet. Furthermore, robotics laboratories have proven the importance of such range scanning systems in terms of robot navigation, object recognition and vision-assisted tools, such as augmented reality devices.

Since the idea of constructing a 3D fax machine a decade ago, researchers have continued to examine new applications in order to advance the 3D scanning technology and to evaluate their findings. Examples include, in the context of the acquisition of cultural heritage artifacts, the Digital Michelangelo Project of the Stanford University and the University of Washington [LPC+00], The Pieta Project of the IBM T. J. Watson Research Center [RBMT] and the works of Rocchini et al. at the ISTI [RCM+01]. All these systems are based on a classical method called the optical triangulation, see Figure 1.1. This approach has undergone numerous drastic improvements in recent years, due to the ever-evolving and more economical hardware, for instance, better CCD imaging qualities and of course new algorithms. Figure 1.2 shows a complete overview of different proposed optical shape acquisition systems.

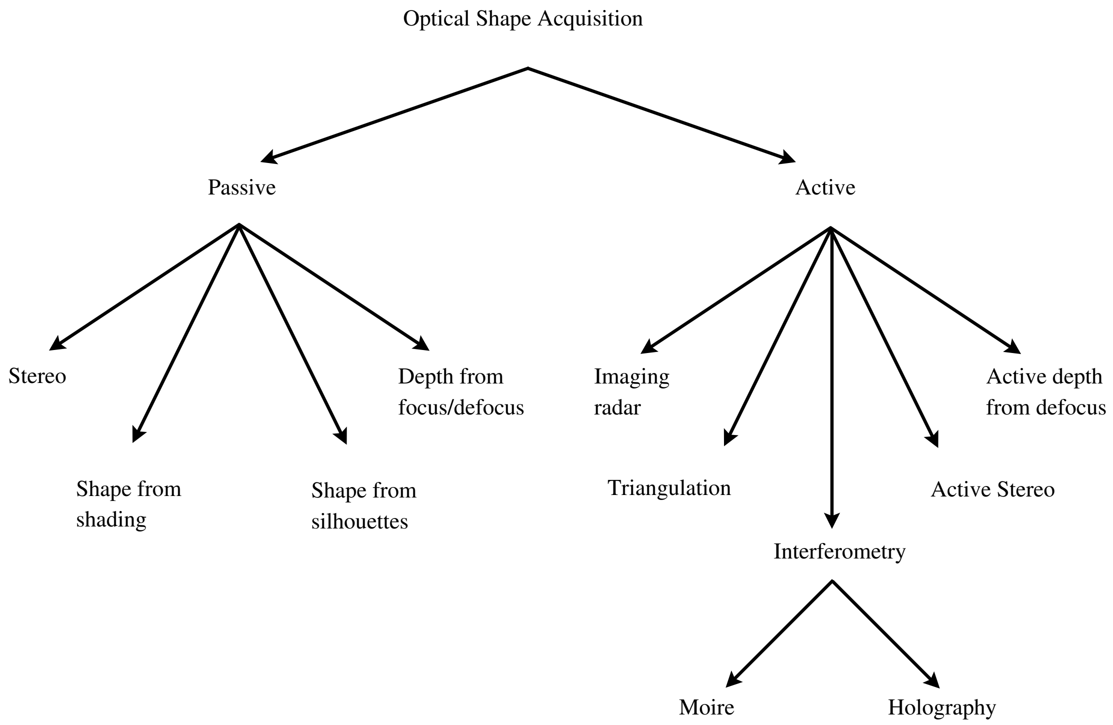


Figure 1.2: A taxonomy of different optical shape acquisitions from [CS00]

Although a large assortment of commercially available 3D scanning devices can be employed for our system, we limit ourselves to using devices which are relatively economical. We also face a limitation on flexibility in terms of the degree of freedom to experiment with new approaches. With these considerations in mind, we decide to restrict the hardware components of our triangulation system to more affordable consumer-level technology, consisting of an ASK M2 DLP video projector for the emitter and a Fuji Finepix S2 Pro professional digital camera for the sensor. The projector produces a specific structured light pattern, in this case a set of edges, which is then projected onto the target object. The digital camera acquires the necessary images from a shifted position w. r. t. the emitter. From the camera's point of view, the structured light appears to be distorted if it is not a plane object. Using these images combined with the relative positions and orientations of the sensor-emitter pair, we are then able to extract the depth information using optical triangulation. Unlike conventional methods, such as using unique coherent light beams (lasers), this method of generating structured light pattern has the advantage of being able to speed up the scanning process by triangulating multiple ranges at a time. In some cases, it is even possible to reduce the number of input images to just one, as shown by Zhang [ZCS02], by using high frequency color encoded patterns and multi-pass dynamic programming to solve the correspondence problem. If the subsampled ranges reach a certain density, zooming in on the object over a particular value will begin to cause aliasing effects, which is manifested as steps. This is due to the pixel-accurate edge detection in the sensored image which leads to a pixel-accurate triangulation. Typically subpixel-accurate edge detection requires multiple input images. A remarkable solution using a sequence of a shifting illumination has been proposed by Curless and Levoy [CL95], namely the space-time analysis approach. Moreover, using dynamic-programming for corresponding still yields to mislabeled edges as a result of undetected edges and the often inappropriate shape of the object.

To solve these problems, we propose a few modifications of the color encoded multi-pass dynamic programming approach. More specifically, we are seeking to design a structured light system that is able to reach subpixel accuracy using one single input image, increase the robustness against mislabeling and achieve better edge detection. We present this technical report as follows. In section 2, we formulate the complete architecture of our acquisition system as a pipeline, where each stage depends on the previous one. Section 3 gives an overview of the structure of the implemented experimental software environment. In section 4, we briefly show some results of our color encoded triangulation system and discuss some remaining problems. Finally, a summary and propositions for future work is suggested in section 5.

Chapter 2

A Shape Reconstruction Architecture

The pre-requisites to the design of a structured light system are an in-depth formal specification of the problem and a well-defined intention. Recall the stages required for a complete scanning process. We need primarily to obtain the camera and video projector extrinsic parameters (orientations and positions in space), as well as the intrinsic parameters, such as the focal length, before any triangulation can be performed. A noticeable solution to this non-trivial calibration problem has been proposed by Zhang [Zha00]. The next step is the range acquisition stage, followed by the registration process which merges the set of range maps together. To finally obtain the mesh of the object, we need to reconstruct a surface from the point-cloud as well as extract view dependent textures in order to effectively visualize material properties. This technical report focuses on the task of range acquisition using a video projector and a digital camera to obtain subpixel accurate range maps, assuming a calibrated emitter-sensor pair.

2.1 Problem Statement

The problem should be stated as illustrated in figure 2.1. The process of triangulation can be modeled by a function f. We input a projected pattern image p, the emitter-sensor parameters θ and an unknown object x to the function, such that $f := f_{\theta,p}(x)$. In the ideal case, f outputs an image i, which is acquired by the digital camera, i. e. $f_{\theta,p}(x) = i$. We extract the shape of the unknown input x from this partially known function and from its output.While f^{-1} refers to an ideal reverse engineering process, we can only afford to find an approximation of it: $x = \tilde{f}^{-1}_{\theta,p}(i)$. This is because the image i has noise and its CCD resolution is limited (subsampling). Moreover the triangulation function f depends on environmental

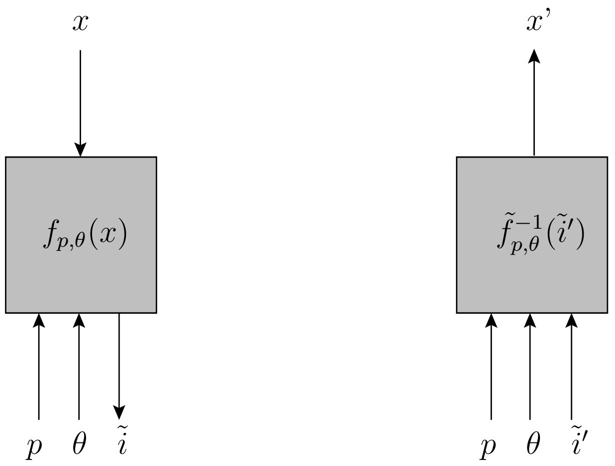


Figure 2.1: A model of the problem statement. On the left, f represents the scene transformation, which distorts the pattern p according to the unknown object x and outputs $\tilde{i}$. On the right, $\tilde{f}^{-1}$ determines our acquisition procedure to recover x'.

factors, such as ambient light, which are also assumed to be unknown. This results in additional sources of errors. We therefore consider the noised image $\tilde{i}$.

To get around these difficulties, we must design carefully an appropriate pattern, which can be clearly distinguished from the object's shape and texture. The acquired image $\tilde{i}$ must be pre-processed to remove high frequency noise. In addition, it must be color corrected and the object must be extracted from the background and purged from pixels under shadows. From this repaired image $\tilde{i}'$, we perform an edge-detection to locate the distorted stripe edges of the pattern p on the object x. The multi-pass dynamic programming algorithm labels each detected edge of $\tilde{i}'$ with its corresponding projected edge of p. As the edges of $\tilde{i}'$ are detected on a CCD, we need to reposition them to obtain subpixel accuracy. This is done by approximating the gradient of the square local contrast of each edge transition by a piecewise linear function. From this set of labeled stripe edges, we compute the optical triangulation on each of them. Finally, we post-process the obtained range map via meshing and face orientation culling to obtain an approximated discretization of the object x'. Figure 2.2 describes this proposed architecture in form of a pipeline.

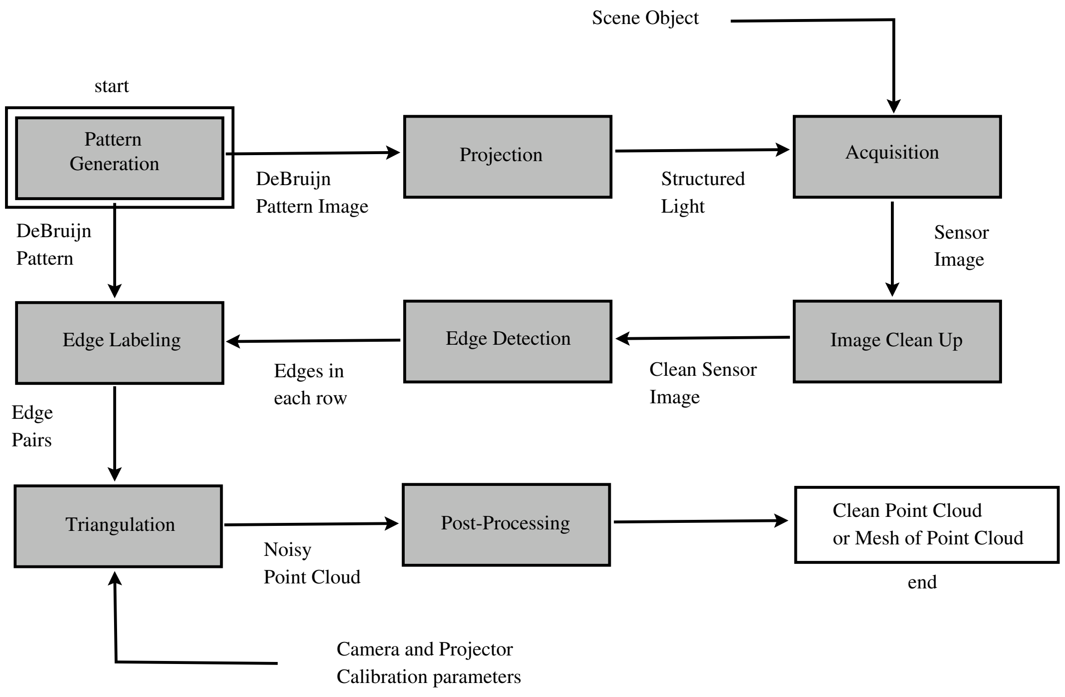


Figure 2.2: An overview of the shape reconstruction architecture.

2.2 De Bruijn Illumination Pattern

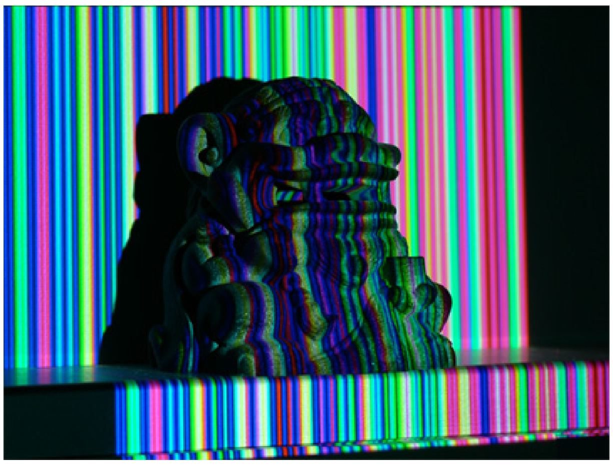


Figure 2.3: A De Bruijn pattern illuminated object.

To accelerate the acquisition process, as mentioned previously, we project multiple edges on the object in a single step. This allows us to triangulate multiple detected edges from one single image. Problems arise when the projected edge is to be distinguished from the object. The object's texture and shape might provoke undesired shadings which can falsify the edge-detection. Moreover, to be able to triangulate, the considered projected edge must be labeled with the right detected edge. This requirement forces us to encode each edge uniquely. In the case of using only one captured image, we are restricted on using color encoding. For robustness against textures of the object, we limit the use of colors to the extreme values of each RGB channel 0 and 1. Unfortunately, this will create only 8 different encoding possibilities, which means only 8 different stripes can be used. To overcome this, we project color stripes and consider the color transitions as edges. In addition, the latter allows us to encode the decreasing -1 for each channel. These three possible values in three channels encodes 27 values but as $(0, 0, 0)$ is equivalent to no transition at all, we subtract 1 from it. For large objects, these 26 possible edge values can be repeated to achieve higher resolution. Assuming that the ensuing correspondence algorithm is computed using a dynamic programming approach, the order of the sequence will be taken into account. A single connected surface will have more priority than scattered ones. But a simple repetition of the sequence will also lead to ambiguity. We therefore need to find a sequence with a good windowed uniqueness property, which is a sequence with small subsequences of unique consecutive stripe edges. A large window size will provoke the possibility of many repetitions within the window, which is not desired. Any subpattern larger than or equal to the size of the window is assured to be unique within the entire stripe edge sequence.

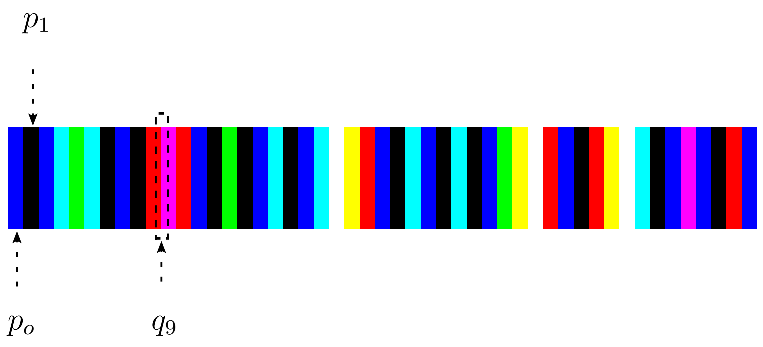


Figure 2.4: A example of the De Bruijn stripe pattern. p_0 and p_1 are stripes and q_9 is an edge, encoded by the transition between p_9 and p_{10}.

Our task can be formally described as follows. Let P be a pattern of colored stripes sequence projected onto the object. P represents a tuple of colors:

$$P = (p_0, p_1, \ldots, p_N)$$

As we are using the transitions to encode the edges, P yields to a sequence:

$$Q = (q_0, q_1, \ldots, q_{N-1})$$

Each element $q_j = (q_j^r, q_j^g, g_j^b)$ has for each of its 3 color channels 3 possible values -1, 0 and 1, as mentioned before. For instance, the edge between magenta $p_{j+1} =$

$(1, 0, 1)$ and yellow $p_j = (1, 1, 0)$ would become $q_j = (0, 1, -1)$. To simplify matters, we refer q to be any q_j value and q^c to be any of (q^r, q^g, q^b). It is clear that any p_j and p_{j+1} have to differ in at least one of their 3 channels so that q_j could become a valid edge. To construct such a sequence P, we choose an initial value p_0 of P and inductively compute the XOR operation on p_j with one element of the following set $\{(0, 0, 1), (0, 1, 0), \ldots, (0, 1, 1), (1, 1, 1)\}$ to obtain the p_{j+1} elements. We now return to the binary coding. The XOR operator has the property of flipping bit values of one operand if the other one is equal to 1. For example, if $p_j = (1, 0, 0)$, we compute $p_j\texttt{XOR}\ (0, 1, 0)$. This operation flips the green channel value of p_j and we obtain $(1, 1, 0)$, which then defines the value of p_{j+1}. Finding a sequence P such that Q has a good windowed uniqueness property of window size n is the same as choosing a sequence of XOR operands of size n that are unique within the whole sequence. Such a sequence can be obtained with the De Bruijn sequences from the field of combinatorics. A k-ary De Bruijn sequence of order n defines a circular sequence

$$D = (d_0, d_1, \ldots, d_{k^n-1}) \quad .$$

Any subsequence of D of length n or greater is assured to appear once. k represents the number of possible values d_j can hold, in this case 7. We then generate one of 7^n possible sequences D of size n. The sequence $(p_0, ..., p_{7^n})$ can now be generated from an initial color p_0, which is an element of $\{(0, 0, 0), ..., (1, 1, 1)\}$, and by iterating through the inductive operation

$$p_{j+1} = p_j \,\texttt{XOR}\, d_j \quad .$$

This implies that a maximum length of 7^n for the sequence P can be reached with a fixed window size n. As noted by Zhang [ZCS02], $(1, 1, 0)$ and $(1, 1, 1)$ are removed from the De Bruijn sequence D because of strong crosstalk errors which resulted during color correction later in the acquisition process. Nevertheless, it is sufficient to work with $k = 5$ and a window size up to $n = 3$ to reach a sequence P of 125 stripes. We have explained how to generate P such that Q is suitable in terms of windowed uniqueness using a De Bruijn sequence D. To generate D, we used algorithm 2.1 available online [Rus00], which is precomputed for different values of n as our projector resolution is limited.

2.3 Input Image Clean Up

Before we start the edge detection, we assume that the images we are dealing with are noisy. Such noise is due to subsampling, unfavorable lighting circumstances, object textures, camera CCD capturing noise and especially the low pixel density of the projector's LCD grid as shown in figure 2.5. Moreover, the shape of the

```
function condition(p)
if (n mod p = 0)
  for j = 1 to p
  print(a[j])

function generate(t,k)
if (t>n)
  condition(p)
else
  generate(t+1)
  for j = a[t-p]+1 to k-1
  a[t] = j

procedure deBruijn(n,k)
a[0] := 0
generate(1,1)
```

Algorithm 2.1: De Bruijn sequence generator.

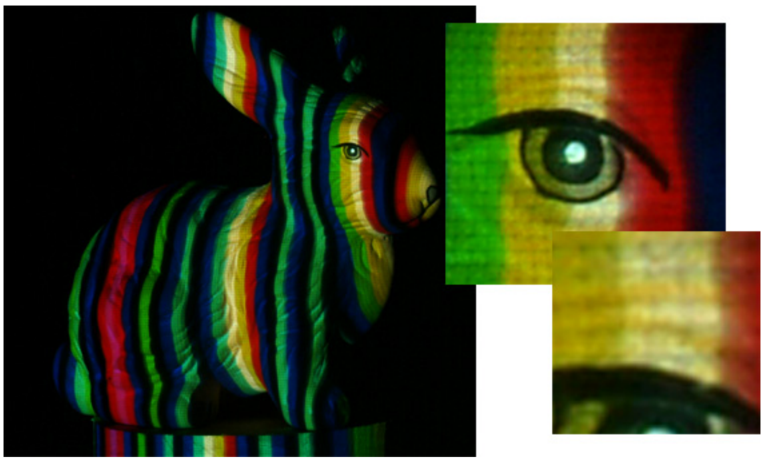


Figure 2.5: The low pixel density is visible in the close up of the capture. The spaces between the projector's LCD grid yield to dark lines. This can confuse the stripe edge detection.

object might be self-occluded and the color of the projected pattern might not correspond exactly to the acquired one. All these factors make consistent edge detection difficult, especially when setting a threshold for a pixel to be recognized as an edge. For this reason, we propose the use of three stages to purge the images. The first stage is responsible for eliminating the pixels that are visible to the sensor and are hidden by the emitter which are manifested as shadows. In the same manner, we detect the background areas and ignore them for subsequent considerations. This has the advantage of speeding up subsequent scanline processes as well as minimizing wrong detections in invalid areas. The second stage takes care of the color crosstalk phenomenon, which is often due to uncalibrated colors projected by the emitter and acquired by the sensor as well as to the surface that modifies the projected color spectrum in unknown ways. For this problem, we use an approximated model proposed by Caspi et al. [CKS98], assuming that the scene reflectance is constant in each color channel and the emitter-sensor coupling matrix are nearly the identity. An optional final step can be applied if the acquired image remains too noisy for further processing. Assuming an additive white gaussian noise over the image, we filter out the high frequency noise with a simple approximated low-pass filter, resulting in a slightly smoothed image. This noise reduction pass should be used carefully as excessive smoothing can make the stripe edge detection even more difficult.

2.3.1 Background and Pixels Under Shadow Masking

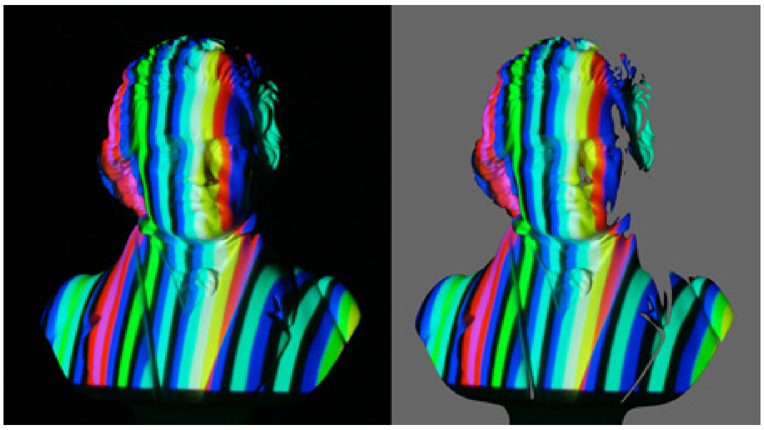


Figure 2.6: Before (left) and after (right) background and pixels under shadow purging. The masked areas are visualized in gray.

We choose to perform the complete acquisition process of the image in a completely dark room. This reduces the intervention from light sources other than the projector light. Moreover, this makes it much easier to extract the relevant objects from its background as well as purging the pixels in shadow. Firstly, we observe that the shadowed pixels and the background are the dark areas in the image. To be more precise with the definition of dark, we normalize each pixel to detect "dark" areas in a relative way. A white light projection is done for this purpose by sending the corresponding plain white image to the video projector. Let $p(u, v)$ be the pixel of the white image projected camera acquisition at position (u, v). Considering the brightness of the image, we desaturate the pixels p to obtain a grayscale value. We then compute by scanlining the image, the brightest pixel p_{max} and darkest pixel p_{min}. We are then able to normalize each $p(u, v)$ to obtain

$$p_n(u, v) = \frac{(p(u, v) - p_{min}(u, v))}{(p_{max}(u, v) - p_{min}(u, v))} \quad .$$

A parametrizable hard threshold is then applied to the normalized image to remove the "dark" areas from the image. In practice, setting all p_n under 0.25 as dark has shown to be a quite suitable value. We save this bitmask of "dark" areas and use it to distinguish between relevant and non relevant pixels. The scanlining process thus speeds up with the amount of unconsidered pixels.

2.3.2 Color Cross-talk Correction

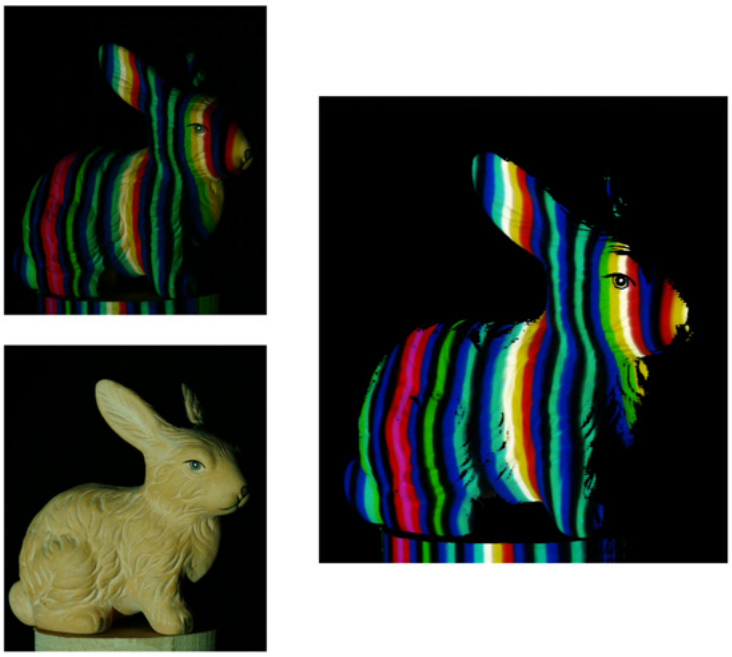


Figure 2.7: Before (upper and lower left) and after (right) color cross-talk correction. We obtain an image with pure colors, by normalizing the stripe capture with the white light capture. The black (ambient) light illumination was not necessary as the capture was completely black. We also note the disappearance of the colored textures after color correction.

As part of the illumination pattern, multiple colored stripes are projected simultaneously onto the object. It is therefore necessary to identify the correct color from the acquisition to be able to label it subsequently. Unfortunately, the acquired color usually differs from the projected one, because of unknown material reflection properties (albedo), the uncalibrated colorimetry of the projector-camera pair and an over complex lighting environment. The color crosstalk has to be corrected in order to overcome this problem. Although the object is assumed having a non neutral color surface (i. e. textures are possible), we restrict ourselves to identifying only pure colors, which are those given by 0 or 1 in each RGB channel to be more reliable. We use the projector-camera coupling model described in [CKS98] by Caspi et al, which relies on the camera and projector properties. While they used a classical LCD based projector, we experimented with the DLP (digital light processing) based projector ASK M2. Instead of splitting the white source into the three RGB beams and by modulating with LCD arrays separately as for the LCD technology, the DLP based projector uses, in our case, a single DMD (Digital Micromirror Device) to substitute the LCD arrays. White light is filtered out by a fast spinning filter wheel so that we can obtain a red, green and blue beam separately in a loop. These beams are projected onto the micro-mirrors, which are controlled by the input image. By setting an appropriate shutter speed, our acquisition device is then able to blend these rapidly alternating switches between the color channels. This time-multiplexed technique ends up to be the same as the classical wavelength-multiplexed output beam of the projector, as the camera CCD receives the signals from all three channels. Because of this, we are able to reformulate the camera-projector coupling model in the same manner as for LCD based projectors.

We choose a reference point on the colored pattern image which is decomposed into the three channels red, green and blue. Let (r, g, b) be its pure color with $r, g, b \in \{0, 1\}$. Recall that r,g and b are input values of the projector. The illumination in the red channel at the reference point is given by $I^t_R(\lambda) \cdot P(r)$, with $I^t_R(\lambda)$ the normalized projected red beam spectrum, $P(r)$,$P(g)$ and $P(b)$ are monotone non-linear factors (due to color cross-talk) and λ the considered wavelength in the color spectrum. A beam spectrum is normalized so as to correspond to a light spectrum for which the maximum value of 1 is reached, for example $\lambda = 590$ nm for the red beam case. The same beam spectrum functions $I^t_G(\lambda)$ and $I^t_B(\lambda)$ apply for those in the remaining channels. The illumination in each channel are summed up during the acquisition by alternating through all the three colors (caused by spinning wheel of the DLP), which yields to the following illumination function:

$$I^t(\lambda) = P(r) \cdot I^t_R(\lambda) + P(g) \cdot I^t_G(\lambda) + P(b) \cdot I^t_B(\lambda) \quad .$$

Assuming no presence of mutual illumination or fluorescence in the object s surface, a linear relation of the projected illumination light and the light reflected from the object would be

$$I^r(\lambda) = I^t \cdot k_i \quad .$$

k_i is a constant factor within a channel. k_i has to be considered differently in each channel because of different reflection properties (albedo) of surfaces. $I^r(\lambda)$ is the light acquired by the camera's CCD array. Finally, the response of a pixel on the CCD array would be formulated as the tuple (R, G, B), where

$$\begin{aligned} R &= \int_0^\infty f_R(\lambda) \cdot I^r(\lambda)\, d\lambda \\ B &= \int_0^\infty f_B(\lambda) \cdot I^r(\lambda)\, d\lambda \\ G &= \int_0^\infty f_G(\lambda) \cdot I^r(\lambda)\, d\lambda \end{aligned}$$

$f_R(\lambda)$, $f_G(\lambda)$ and $f_B(\lambda)$ are the spectral responses of the camera filter in each channel. Thus, we can bring the equations in all three channels, by taking into account an ambient environment illumination, to an equation system in matrix form as follows:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = A \cdot K \cdot \begin{bmatrix} P(r) \\ P(g) \\ P(b) \end{bmatrix} + \mathbf{c} \quad , \tag{2.1}$$

with illumination acquisition matrix (projector-camera coupling matrix)

$$A = \begin{bmatrix} \int f_R(\lambda)\, I_R^t(\lambda) d\lambda & \int f_R(\lambda)\, I_G^t(\lambda) d\lambda & \int f_R(\lambda)\, I_B^t(\lambda) d\lambda \\ \int f_B(\lambda)\, I_R^t(\lambda) d\lambda & \int f_B(\lambda)\, I_G^t(\lambda) d\lambda & \int f_B(\lambda)\, I_B^t(\lambda) d\lambda \\ \int f_G(\lambda)\, I_R^t(\lambda) d\lambda & \int f_G(\lambda)\, I_G^t(\lambda) d\lambda & \int f_G(\lambda)\, I_B^t(\lambda) d\lambda \end{bmatrix}$$

and albedo matrix

$$K = \begin{bmatrix} k_R & 0 & 0 \\ 0 & k_G & 0 \\ 0 & 0 & k_B \end{bmatrix}$$

and ambient illumination vector

$$\mathbf{c} = \begin{bmatrix} R_0 \\ G_0 \\ B_0 \end{bmatrix} \quad .$$

Our task now is to recover the correct (r, g, b) vector from the acquired biased (R, G, B) color vector. (R_0, G_0, B_0) can be recovered under black light illumination, which acts as the ambient environment illumination. P is as mentioned before a monotone non-linear function, hence invertible. P is usually computed using a look-up table, that is pre-computed as part of an offline colorimetric calibration.

To simplify matters, we can avoid the computation of A, K and P by performing an online calibration and by assuming a perfectly color calibrated projector. A projector is considered color calibrated when P is the identity, which means that the instruction color perfectly matches the output illumination at the considered wavelengths. This is done as follows:

We compute a white illumination using a white pattern

$$(P(r), P(g), P(b)) = (r, g, b) = (1, 1, 1) = \mathbf{e} \quad .$$

This yields to the following CCD color response

$$\begin{bmatrix} R_W \\ G_W \\ B_W \end{bmatrix} = A \cdot K \cdot \mathbf{e} + \mathbf{c} \quad . \tag{2.2}$$

We obtain the following result from the equations 2.1 and 2.2:

$$\begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} \frac{R-R_0}{R_W-R_0} \\ \frac{G-G_0}{G_W-G_0} \\ \frac{B-B_0}{B_W-B_0} \end{bmatrix} \quad .$$

The recovery of the input (r, g, b) assuming a perfect projector side colorimetric calibration turns out to be a simple normalization using a white pattern light (maximum light) and an ambient black pattern light (minimum light).

In contrast to Zhang et al. [ZCS02], we have chosen not to compute this stage off-line in our experiments due to the loss of robustness against surface texture and for the sake of simplicity. However, the drawback of requiring 2 more images has to be taken into account, namely the white light projected image and the black light projected image. As we previously used the white light projected image to perform the background and pixels under shadow elimination and assuming the weak influence of the ambient light generated by a black illumination, we can reduce the scanning process to two images. The black projected image would be replaced by an image with all pixel values equal 0 in all channels. We leave it up to the developer to decide whether or not to compute the colorimetry calibration off-line in the final implementation.

2.3.3 Optional Low-pass Filtering

As we are using uncompressed raw data from the camera, the quality of the acquired image is usually good enough for the edge detection pass. However, the scanned object might still contain high-frequency and high-contrast areas due to its possible complex shape and surface texture although it has already been relaxed through color correction. This leads to local contrasts which result in wrongly identified edges. We assume that each pixel is affected by an additive white gaussian noise (AWGN), as it is supposed to be statistically independent for each pixel. Thus, the pixel rows of the image are typically modeled as a stationary process. This noise manifests itself as non-zero amplitudes at higher frequencies in the power spectrum of the Fourier domain of the considered image signal. In this particular case, we simplify the image processing by restricting ourselves to 1-dimensional image signals, i. e. the rows of the image, if and only if the projected stripes are perfectly vertical. If only vertical stripes w. r. t. the camera's point of view are projected, then only vertical edges should be detected. This detection is done separately in each color channel via a horizontal scanline process so that we are able to perform multi-spectral edge detection further. To reduce the noise by lowering the undesired high frequency amplitudes, we experimented with different filter kernels which are ideally suited for discrete signal processing and easy to implement. Let

$$F(w) = \int_{-\infty}^{\infty} f(t)\, e^{-i\,w\,t}\, dt$$

be the Fourier Transform of the image signal $f(t)$ and $H(w)$ be the low-pass filter which reduces the undesired high amplitudes when multiplied by $F(w)$. We obtain a cleaned up $G(w) = F(w) \cdot H(w)$ in the Fourier domain. We also know that $g(t)$, which is the inverse Fourier transform of $G(w)$, is obtained by convolving $f(t)$ and $h(t)$, with $f(t)$ being the horizontal 1-dimensional image signal and $h(t)$ being the filter in the space domain, which is also called the impulse response in systems theory. We obtain:

$$g(t) = f \otimes h(t) = \int_{-\infty}^{\infty} f(s)\, h(t-s) ds \quad .$$

Because we are using CCD captured pixel arrays, we approximate the considered functions in space domain by a subsampled discrete signal. $f(t)$, $h(t)$ and $g(t)$ becomes $f(n)$, $h(n)$ and $g(n)$ respectively. The implementation of the discrete convolution is straightforward by computing

$$g(n) = \sum_{m=min}^{max} f(n)\, h(m-n)$$

min is the lowest n with $h(n) \neq 0$ and max is the highest n with $h(n) \neq 0$. Different filter kernels $h(n)$ have been tested. In practice, the rather small normalized kernel

$$h = \frac{1}{4}\,[1\;2\;1] = \begin{cases} 1/4 & , n = -1 \\ 1/2 & , n = 0 \\ 1/4 & , n = 1 \\ 0 & , else \end{cases}$$

was able to reduce most local noise and keep the stripe edges visible enough for the following edge detection stage.

2.4 Color Edge Detection

To be able to extract range data through optical triangulation, the rays of the projected stripes and the rays reflected from the object to the camera must be known in order to compute the corresponding intersection, which is the localization of the triangulated point in 3-space. As the projected stripe pattern image is given, we only need to determine the rays reflected from the object to the sensor. These rays pass through the camera's CCD array and its central point. We must therefore locate the ray's intersection with the camera's CCD, which is the same as finding the pixels where a stripe transition occurs. Again, we assume that the vertical oriented stripes w. r. t. the camera form the edges we are looking for. This implies that only a 1-dimensional edge detection in the rows of the image is required. To be consistent with our previously generated De Bruijn pattern, we have to detect edges in colored images, which need an additional consideration of the local contrast in all three channels.

2.4.1 Multi-spectral Definition of Edges

For edge detection to be as consistent and as accurate as possible, it is crucial to have a formal definition of edges in multi-spectral images. If, for example, a colored image is converted into grayscale (same as the intensity channel of the HSI color model), no edges will be detected between a pure red stripe and a pure blue stripe if their light intensities are equal. A survey by Novak and Shafer has shown that 10 percent of the edges are left undetected using this method. In our case, this is even worse after color normalization. One way to overcome this problem will be to find the edges in all three color channels independently and combine the results using a logical `OR` operator in the same reference point, as mentioned by Koschan in [Kos95]. Using this method is problematic for edges, that are detected in more than one color channel and they don't occur at the same position. To be more rigorous, we choose to use Cumani's definition of an edge [Cum91]in color images, which has proven to be a good choice shown in many colored edge detection studies. In contrast to the general Cumani's approach, our special case allows us to work with, easier to handle, 1-dimensional tri-band image signals. A directional gradient analysis can therefore be neglected.

Let the row of a 3-band image of length N be a function

$$\begin{array}{rccc} f: & \{0,\dots,N-1\} & \Rightarrow & [0,1]\times[0,1]\times[0,1] \\ & n & \mapsto & F=(f_r(n),f_g(n),f_b(n)) \end{array} .$$

To ensure a more robust detection against high frequency noise, we first smooth the image by convolving with a simple averaging

$$\frac{1}{2}[1\ 1] \tag{2.3}$$

filter kernel. In discrete signal processing, it is common to approximate infinitesimal length to the neighboring pixel distance, which is a displacement of length one pixel. We obtain the following discrete derivative of F in its only direction n:

$$\begin{aligned} \triangle F(n) := F(n+1) - F(n) &= \begin{bmatrix} f_r(n+1) - f_r(n) \\ f_g(n+1) - f_g(n) \\ f_b(n+1) - f_b(n) \end{bmatrix} \\ &= \begin{bmatrix} f_r \otimes [1\ -1]\,(n) \\ f_g \otimes [1\ -1]\,(n) \\ f_b \otimes [1\ -1]\,(n) \end{bmatrix} \quad . \end{aligned}$$

Convolving with $\frac{1}{2}[1\ 1]$ followed by $[1\ -1]$ is equivalent to convolving directly with $\frac{1}{2}[1\ 0\ -1]$, which is also known as the Prewitt kernel. Intuitively, an edge seems to be a local contrast in light intensity or a drastic color variation or both together. Using the Euclidean metric, we are able to measure these two properties as a movement in a 3-space with (R, G, B) as basis vectors. This implies that the definition of the square local contrast of F at n is the squared norm of $\triangle F$

$$S(n) = \triangle F \cdot \triangle F = \triangle f_r(n)^2 + \triangle f_g(n)^2 + \triangle f_b(n)^2 \quad .$$

This yields to $S(n) = df(n)^2$ in the monochromatic case, which is also a valid definition. Finally, we define an edge point in a multi-spectral image as a point, for which the square local contrast $S(n)$ exhibits a local maximum at the pixel n. In connection with zero-crossing methods, a stationary contrast edge point of a tri-band image is a pixel n where the derivative of the local square contrast $S(n)$ is zero. To ensure a local maximum of the contrast, we must check if the zero-crossing is transversal or not. We note that only maxima can occur as we are deriving a norm. The edge is then detected at the pixel n for which the derivative of $S(n)$

$$\triangle S(n) = S(n+1) - S(n) \tag{2.4}$$

has a transversal zero-crossing and if, $S(n)$ and $S(n+1)$ exceed a certain threshold. This can be verified rapidly by checking if

$$|S(n) + S(n+1)| = |S(n)| + |S(n+1)| \tag{2.5}$$

as both sides are equal if and only if the signs of $S(n)$ and $S(n+1)$ are equal. We observe that the $\frac{1}{2}$ factor is useless and can be omitted in 2.3. At this point we are able to detect an edge in a multi-spectral image at pixel accuracy, using a single hard threshold. To be able to detect edges that are caused by single channel variations, we must choose a threshold, that is smaller than 1. We also note that, in addition to thresholding, the transversal zero-crossing check permits reduce wrongly detected noise pixels, that occurs even when there is no local maximum. This is often caused by a small contrast, which appears in more than one channel at a given position, near a real stripe edge.

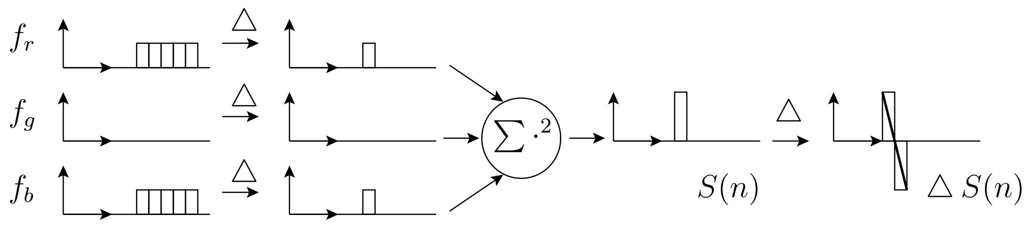


Figure 2.8: An illustration of the 1 dim. edge detection process. In this example, we observe the ideal case of a black to magenta transition. A transversal zero-crossing occurs at the gradient of the square local contrast of the input signal.

2.4.2 Consecutive Edge Pixels Restriction

The edge detection is a decisive stage in the acquisition process, especially when labeling based on colored stripe transitions is heavily dependent on the local contrast of the edge point. A wrongly detected edge will have a different contrast which might confuse the dynamic programming approach of the labeling. Moreover, a wrongly localized edge will also yield to an erroneous optical triangulation. The previous semi-automatic edge detection, which has been deduced from a more general framework, might not suffice for our application. It detects shading contrasts as well as color transition contrasts, which isn't always desired because our main task is to extract the stripe transitions edges and not all edges. Using the fact that stripes have, in most cases, a certain width on the captured image, we temporarily buffer consecutive detected edges during the scanlining process until a certain number of consecutive pixels M has been reached. The buffer only contains the edge with the maximum contrast of the consecutive detected edges. If no edge is detected before M is reached, we choose the buffer content to be our detected

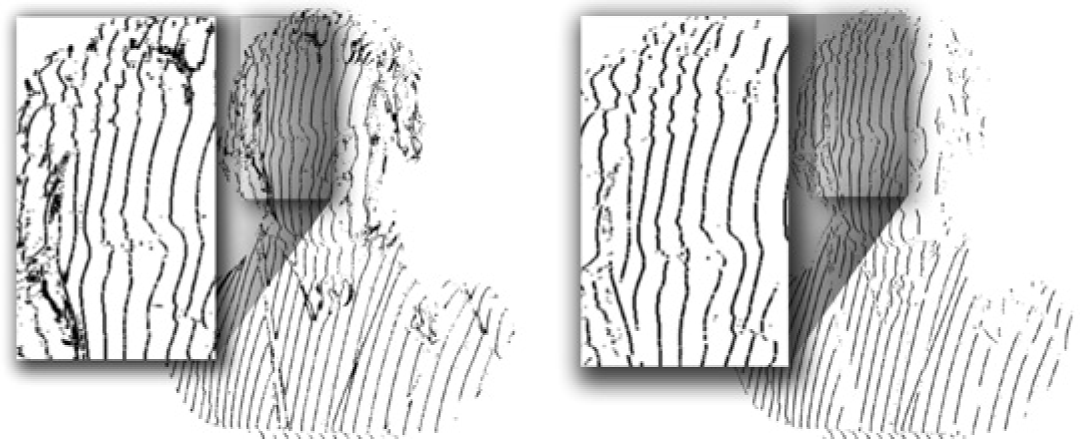


Figure 2.9: While on the left, a simple color edge detection (using Cumani's definition of colored edge) has been performed, we observe a noticeable improvement in terms of stripe edge detection by restricting 5 consecutive edge pixel ($M = 5$).

edge and restart the procedure. Thus, we are able to use context knowledge about the stripe width to modify our previous definition of local contrast for this particular application. We define M as the number of consecutive edges (in pixels) restriction, which is an additional parameter. In practice, we set M to be slightly smaller than the average stripe width on the acquired image so that the advantage of detecting the correct edges outweighs the drawbacks of not detecting any edges. The procedure is described in algorithm 2.2:

```
procedure edgeDetection(imageRow,threshold,M)
lastPosition = 0
edgeCandidate = empty
for  x = 0 to imageRowLength-1
  s(x) = squareLocalContrast(x)
  if (transversalZerocrossing(s(x)) AND s(x) > threshold)
    if (edgeCandidate isEmpty OR x < lastPosition+M)
      if (edgeCandidate isEmpty OR detectedEdge has greater contrast than
      edgeCandidate)
        edgeCandidate = detectedEdge(x)
        lastPosition = x
    else
      return edgeCandidate
      edgeCandidate = empty
end for
```

Algorithm 2.2: Color edge detector using consecutive pixels restriction.

We note that M is always > 1, which means that two edge pixels must at least be separated by 2 pixels. This is due to our previous convolution of the image with $\frac{1}{2}[1\,1]$ and because the square local contrast is always positive. It is therefore possible to always achieve better results than the edge detection without the consecutive pixels restriction by setting $M = 2$, while assuring correctness. W.l.o.g. we consider a single color channel (monochromatic case). For $M = 1$: two edges are separated by 1 pixel. Assuming that we have detected two consecutive edges using the 1-dimensional Prewitt kernel, the gradient of the square local contrast must have consecutively decreasing and increasing contrast values to produce two consecutive transversal zero-crossing, which is impossible as the square local contrast, being positive, can only yield to decreasing transversal zero-crossings. In other words, the derivative of a positive increasing and decreasing signal only yields to a decreasing transversal zero-crossings and this is not feasible within 3 pixels (which is the increasing and decreasing sequence), containing the single pixel between two detected edges.

2.5 Edge Correspondence

The labeling process is responsible for the re-identification of the projected edges from the observed image, which is distorted as we are projecting multiple edges at one go, to accelerate the acquisition process. In our case, in connection with the use of De Bruijn colored stripes pattern, we must be able to tell which detected edge, represented by a stripe transition, corresponds to a projected edge. A wrongly labeled projected and captured edge pair will consequently produce an erroneous triangulated range. The method based on multi-pass dynamic programming proposed by [ZCS02] is very well suited for solving the problem of labeling multiple color-encoded edges, especially in the case of holes and self-occlusions in the scanned surface, which yields to undetected edges. However, it has shown some weakness as the extreme edges of our projected pattern are wrongly or simply not detected. Although the dynamic programming approach tends to generate a single connected surface rather than a scattered one, we often observe mislabeled edges at the object s boundary. Based on this, we propose a modified usage of the suggested corresponding by considering a background plane behind the object and masking it during the triangulation, processed in a later stage. Thus, we are able to fix the wrongly labeled edges around the boundary of the object, as demonstrated by our results, figure 4.9 to 4.12.

2.5.1 Multiple Hypothesis Code Matching

Obtaining the correspondence between the edges in the projected pattern and the detected edges in the acquired image is equivalent to determining the correspondence between the rows of the projected pattern and the rows of the captured image. Thus, our problem is now reduced to a 1 dimensional labeling problem.

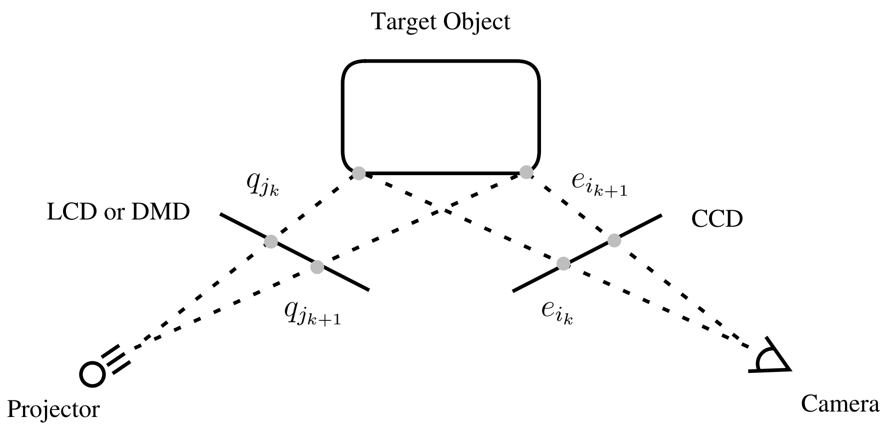


Figure 2.10: Top view of the optical triangulation with the stripe and capture enumeration.

Using the multi-stripe technique, our challenge is to distinguish between the edges w. r. t. their color encoding and their order. Ambiguity can occur due to the limited colors available. This is relaxed with the windowed uniqueness property of the De Bruijn pattern. We use the same notations as in section 2.2 to describe the projected string of edges $Q = (q_0, ..., q_{N-1})$, which is the same in all rows. After computing the edge detection in the considered row of the captured image, we obtain a sequence of color edges:

$$E = (e_0, ..., e_{M-1})$$

Using the same notation as before, the color edge

$$e = (e^r, e^g, e^b) \in [-1, 1] \times [-1, 1] \times [-1, 1]$$

defines the intensity gradients in each channel. These intensity gradients are the same as local contrasts between two colored stripes, which implies a value between -1 and 1. Our aim is to pairwise find the corresponding edges between the sequence Q and E. Thus, the correspondence can be seen as a function with input (Q, E) and output a pair sequence. Unfortunately, this faces the following difficulties that can lead to misclassification. The edge detection, that provides the input sequence E, might not provide sufficiently consistent edge colors, not to mention wrong ones, due to surface reflectance and shading, device color cross-talk and sensor noise. Moreover, missing edges, that are not visible from the camera's point of view, due to surface occlusions, shadows and surface discontinuities, blurs the knowledge about the order of the sequence which is necessary for labeling.

A technique, also referred to as the multiple hypotheses evaluation, is used to solve this problem. Instead of assigning a unique label to the edges in the image, all labeling combinations are considered with their corresponding matching probability. The objective is then to find the optimal labeling among all the considered ones with the highest probability of matching. This yields to a classical global optimization problem. The problem can be formulated as follows:

Let q_j be the j-th projected edge in the stripe edge transition Q and e_i the i-th edge in the detected edge sequence E of a structured light pattern row.

$$\Phi = \{(j_1, i_1), ..., (j_k, i_k), ..., (j_P, i_P)\}$$

defines the global match hypothesis which depicts the probability of matching. P is the number of integer pairs contained in Φ. To simplify matters, we set

$$j_1 < j_2 < \ldots < j_P$$

The integer j should not be confused with q_j and i should not be confused with e_i. q_j and e_i contain their corresponding triple of color edge values and, for the implementation, the position so that triangulation can be performed on them. To clarify our notation, (j_k, i_k) is equivalent to the statement that e_i and q_j are the k-th corresponded pair and Φ represents the set of all the corresponded pairs. Further, we observe that corresponding q_j with e_i, which is done by Φ, can be seen as an injective function that maps the set $\{j\}$ of cardinality N onto $\{i\}$ of cardinality M. Φ turns out to be a special case of a relation. It is injective because each detected edge e can only be affected by at most one projected edge q and it is a function because each q can only be affected by at most one e. Let

$$\begin{array}{llll} f_\Phi : & \{j\} \times \{i\} & \rightarrow & \{0,1\} \\ & (j,i) & \mapsto & \begin{cases} 0 & , (j,i) \in \Phi \\ 1 & , (j,i) \notin \Phi \end{cases} \end{array}$$

be the characteristic function of the set Φ. Accordingly, f_Φ is a reasonable representation of Φ and can also be interpreted as a 2D matrix array as illustrated in 2.11, with columns $j \in [0, N-1]$ and rows $i \in [0, M-1]$. If a match occurs between q_j and e_i, $f_\Phi(j,i)$ is equal to 1. Eventually, all the 1 elements in the matrix f_Φ can be connected to represent a path from the left to the right in the array. Φ being injective implies that no more than one 1 element can occur in each row and column. Our task of finding the optimal global match hypothesis Φ has been turned to finding an optimal path in a 2D matrix array f_Φ.

We recall that our criterion of an optimal match is the match with the most consistent projected and captured edge pairs. We therefore need to define the quality of a match in terms of a correspondence consistency. Consistency can be measured with a scoring function score(q_j, e_i), which is examined more in detail in section 2.5.4. The score for the entire match is defined as

$$\sigma(\Phi) = \sum_{k=1}^{P} \{\, \text{score}(q_{j_k}, e_{i_k})\}$$

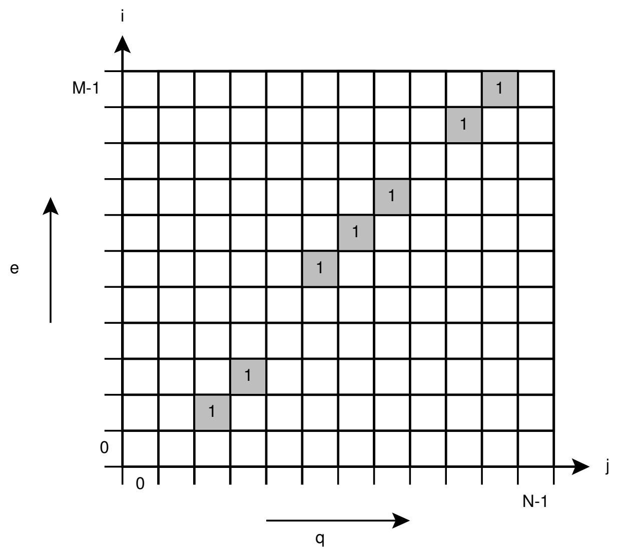


Figure 2.11: 2D matrix array representation of f_Φ. The gray elements marked 1 represent monotonic paths in this array.

This is reasonable, because each component of the sum can be geometrically interpreted as a translation vector in a 1D space which moves further than others if its score is higher. In this case, an energy based squared sum would be inappropriate because negative scores will also move the vector forward which doesn't make sense. Thus, the optimal match is

$$\Phi^* = \arg\max_{\Phi}\{\sigma(\Phi)\} \quad .$$

2.5.2 Dynamic Programming For Solving Correspondence

Considering the paths from left to right, a brute force approach to find Φ^* would be an overkill as it yields to $O(M^N)$ possible matches. To relax this intractability, we introduce an assumption of depth ordering (monotonicity) which is $i_1 < \ldots < i_P$. This assumption is violated when a possible occlusion in the object occurs and leads to a wrong triangulation or, in most cases, holes in the reconstructed scene. This problem can be overcome with a multi-pass technique described in section 2.5.3. Let $path1$ be shorter than $path2$, both starting at column and row zero. By assuming monotonicity, the array that forms $path1$ is always kept within the array that forms $path2$. W. l. o. g. we assume that there is only one optimal path in f_Φ. Using this strict ordering assumption, we are able to identify a path by a subarray, as it must uniquely lie inside it. Typically for dynamic programming approaches, a substructure of the problem can be observed. In order to find the final optimal path Φ^* in f_Φ, we need to find an optimal subpath in the subarray of

f_Φ. We note a dependency between the subproblems. Let us formally characterize the substructure.

Let $G_{j,i}$ be a subarray of f_Φ

$$G_{j,i} := [0, j] \times [0, i]$$

and $\Phi^*_{j,i}$ the optimal path in $G_{j,i}$.

We define 3 possible configurations, which characterizes the possible directions the path can go, namely to the right, upper right and up:

1. a match is found, which is interpreted as incrementing the subarray in horizontal and vertical direction as the injectivity of Φ is not violated. The ideal case would be that this direction is always taken, which is prevented by occlusions and shadows. This is shown in figure 2.12.

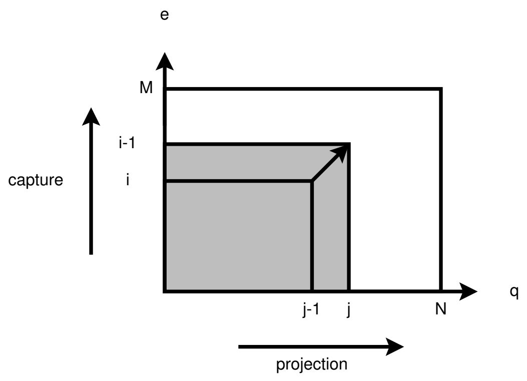


Figure 2.12: $\Phi^*_{j,i} = \{(j,i)^t\} \cup \Phi^*_{j-1,i-1}$

2. We increment the subarray in the vertical direction and no match is found because the injectivity of Φ is violated. This is shown in figure 2.13.

3. We increment the subarray in the horizontal direction and no match is found because the injectivity of Φ is violated. This is shown in figure 2.14.

The substructure characterization is then followed by a recursive definition of the value of an optimal solution. This is equivalent to recursively defining the score of the entire match Φ using the previously defined subarrays:

$$\sigma(\Phi^*_{j,i}) = \begin{cases} 0 & \text{, if } j = 0 \text{ or } i = 0 \\ \max & \begin{cases} \sigma(\Phi^*_{j-1,i-1}) + \text{score}(q_j, e_i) \\ \sigma(\Phi^*_{j-1,i}) \\ \sigma(\Phi^*_{j,i-1}) \end{cases} \end{cases} . \tag{2.6}$$

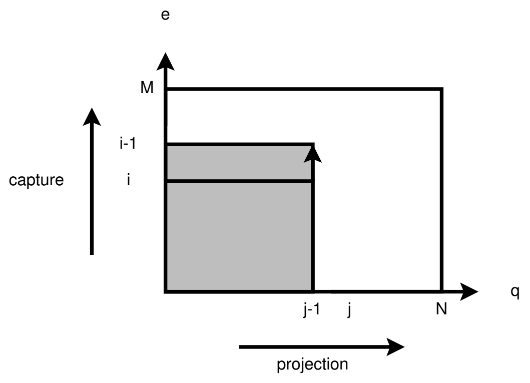


Figure 2.13: $\Phi^*_{j,i} = \Phi^*_{j-1,i} \subseteq G_{j-1,i}$

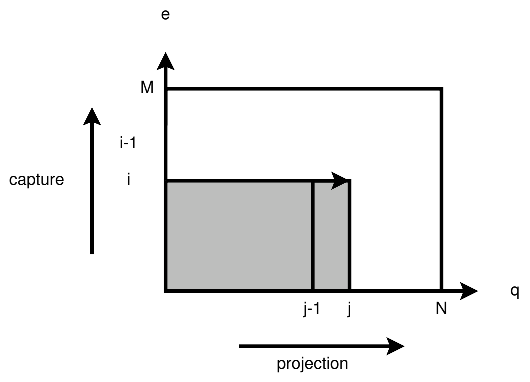


Figure 2.14: $\Phi^*_{j,i} = \Phi^*_{j,i-1} \subseteq G_{j,i-1}$

The recursive definition allows us to compute the $O(M \cdot N)$ different scores of the corresponding paths in a bottom up fashion, by filling up an array containing all the scores, yielding a cost matrix. Finally, we backtrack the cost matrix in a greedy manner. From right to left, we move through the matches with the highest scores and obtain the final optimal path which yields to the demanded optimal global match hypothesis in $O(M + N)$. This is justified by the substructure property of the problem described previously.

2.5.3 Multi-pass Procedure To Avoid The Monotonicity Assumption

As seen before, using the dynamic programming approach is limited to strict monotonicity assumption. Any projected edge index j_{k-1} that is smaller than another edge j_k can only be assigned to a captured edge index i_{k-1} that is smaller than the assigned i_k. Thus we are only able to scan objects that do not contain any occlusion, which tends to influence the order of the acquired edges as illustrated in figure 2.15.

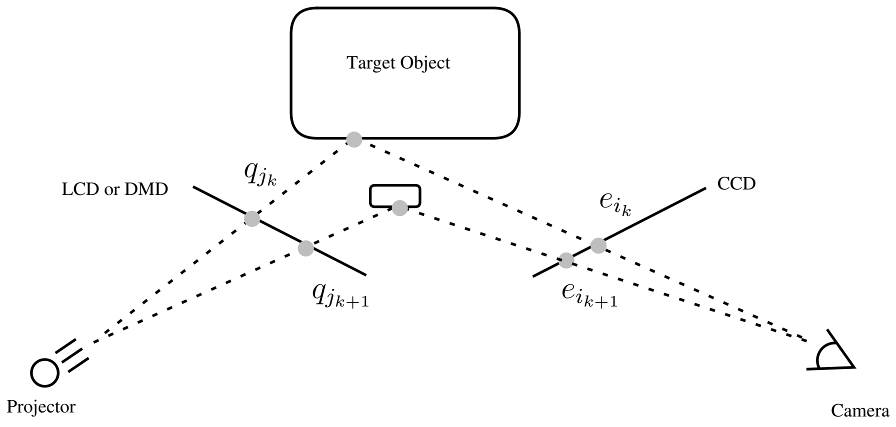


Figure 2.15: Occlusions in object cannot be solved using the simple dynamic programming approach. We have $j_k < j_{k+1}$ but $i_k > i_{k+1}$, that violates the monotonicity assumption. The multi-pass procedure is therefor required.

In order to be able to scan realistic objects that potentially contain occlusions, we adapt a multi-pass procedure proposed in [ZCS02]. In practice we observed that the computed monotonic optimal path corresponds to a correct subpath of the optimal solution. Whenever an occlusion occurs it will appear as a hole. The way to overcome this problem is therefore to fill these holes by iteratively recomputing the dynamic programming until no matches can be found and by taking into account previously computed paths. In this way, algorithm 2.3 describes this multi-pass procedure that wraps over an underlying dynamic programming procedure and we can obtain a match such as the one in figure 2.16.

```
procedure Multipass(scoreArray)
finalPath = clear
temporaryPath = clear
repeat
  temporaryPath = labeling(scoreArray)
  add temporaryPath to finalPath
  remove columns and rows in temporaryPath from scoreArray
until  temporaryPath is an emptyPath
return finalPath
```

Algorithm 2.3: Multi-pass Procedure.

In contrast to the pure monotonic dynamic programming approach, this easy to implement procedure solves the occlusion problem with an additional $O(N)$ factor. In practice objects that contain too many occlusions might already fail the edge detection pass due to high shading frequencies that increase with the objects shape complexity.

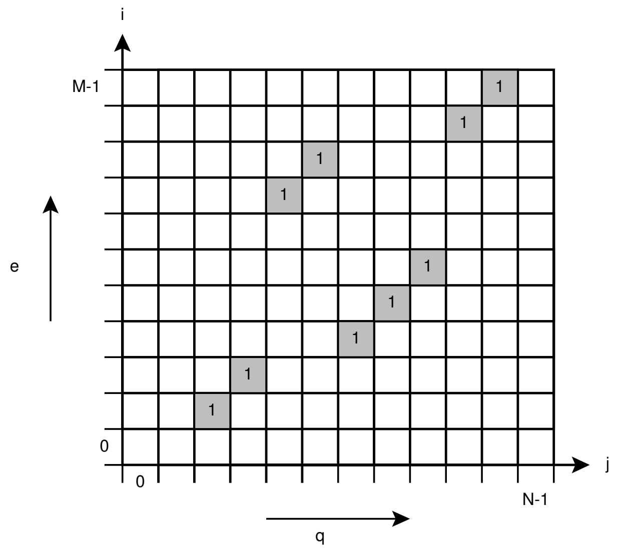


Figure 2.16: A non monotonic match. This can be found using the multi-pass overlying procedure.

2.5.4 Quality Of A Match: The Scoring Function

In relation with our previously described dynamic programming, we have recursively defined the value of the optimal solution (equation 2.6). This definition contains a scoring function score(q, e) which is the value that measures the consistency between a projected edge q and a captured edge. In this manner, we compute all values of the potential solutions by summing up the scores of the considered match, as described in 2.5.1. We now introduce score(q, e) as a specific definition of consistency. We recall that the projected edge value is given by $e = (e^r, e^g, e^b)$, with $e^c \in [-1, 1]$, while the captured edge value is given by $q = (q^r, q^g, q^b)$, with $q^c \in \{-1, 0, 1\}$. Thus, two edges are consistent if their values match in all three channels. However, for comparison purposes, score(q, e) must return a scalar value. Therefore, we compute the consistency of a pair of edges in each channel independently and set the scoring function to be the smallest value among them for a pessimistic estimation. We obtain:

$$\text{score}(q, e) = \min_{c \in \{r,g,b\}} \{\text{consistency}(q^c, e^c)\} \quad .$$

By just taking the minimum value we can be certain that the return value of score(q, e) won't be falsified by values that are too consistent in the other channels, which would have happened if we consider all channels at the same time.

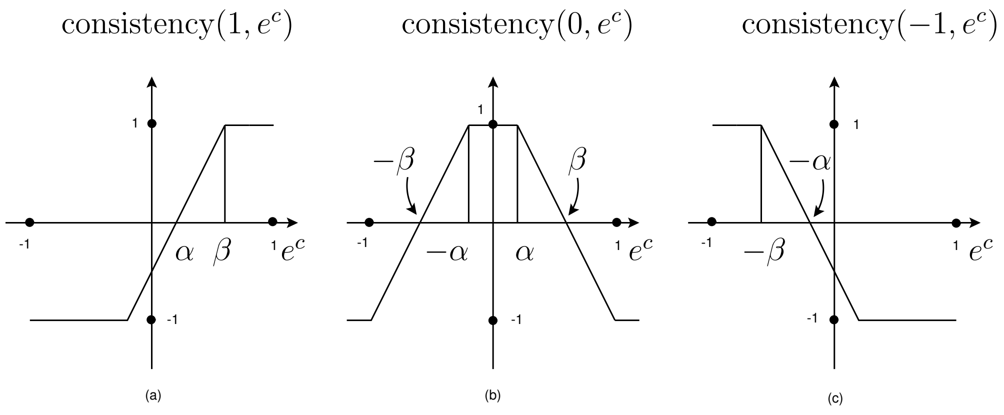


Figure 2.17: The consistency(q^c, e^c) function defined for $q^c = 1$ in (a), $q^c = 0$ in (b) and $q^c = -1$ in (c)

consistency(q^c, e^c) is a function that measures the agreement between q and e in a single channel by returning normalized scalar values. In this manner, we define for instance, consistency$(1, e^c)$ to be equal to 1 if e^c exceeds a certain threshold β. Every edge pair that has a value greater than β is considered to have a valid consistency. When e^c is below β, consistency$(1, e^c)$ will have a linear decreasing value that is equal to 0 for a small value of e^c in α. For inconsistent edge pairs, consistency$(1, e^c)$ carries a negative value. A more comprehensible Figure 2.17 illustrates the definition of consistency$(1, e^c)$, as well as for the cases of $q^c = 0$ and -1. Thus, a formal definition can deduced with the following equations:

$$\begin{aligned}\text{consistency}(1, e^c) &= \text{clamp}(\tfrac{e^c-\alpha}{\beta-\alpha}; -1, 1)\\ \text{consistency}(0, e^c) &= \text{clamp}(1 - \tfrac{|e^c|-\alpha}{\beta-\alpha}; -1, 1)\\ \text{consistency}(-1, e^c) &= \text{consistency}(1, -e^c)\end{aligned}$$

where

$$\text{clamp}(x; x_0, x_1) = \begin{cases} x_0 & \text{, if } x < x_0 \\ x & \text{, if } x_0 < x \leq x_1 \\ x_1 & \text{, if } x_1 < x \end{cases} .$$

$0 \leq \alpha < \beta \leq 1$ are soft thresholds as they represent the linear decreasing consistency measurement, which implies a certain notion of uncertainty when e^c is situated between α and β. The uncertainty of consistency linearly increases with $\beta - \alpha$. In the particular case of $\alpha = \beta$ we would obtain a hard threshold measurement. In practice, we achieve best results by maximizing the uncertainty with $\alpha = 1$ and $\beta = 0$. This is mainly due to the shadings in the object.

We also note as mentioned in [ZCS02], that when using the summation based optimal solution value, only matches with a positive consistency value are taken. If it is negative, a greater optimal solution is achieved by not taking it into account. This implies that negative consistency values are actually not required, but they help to stabilize possible numerical problems for very large negative values when α and β are too close.

2.5.5 Additional Ambiguity Avoidance Through Masking

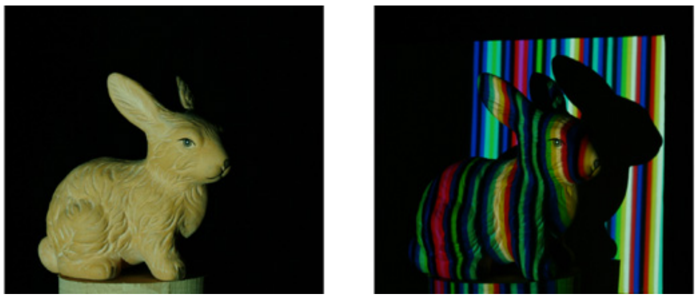


Figure 2.18: The structured light illuminated object image is shot with a plane wall behind it (right). We reuse the white light projected image taken during color correction without the wall (left) to distinguish the object from its background. The wall contains the missing colors that cannot be reflected by the object. This stabilizes the matching using dynamic programming.

Because of the limited windowed uniqueness of the De Bruijn pattern (of window size n), we must expect repeating subpatterns of size smaller than n within a large sequence. Another observation is that in many cases the silhouette of the scanned object does not cover the complete stripe pattern. Some areas of the stripes do not appear on the object. Moreover, our dynamic programming approach evaluates the cost matrix in a bottom up fashion, which is in this case a scanline from left to right. These three situations often cause wrongly matched edge correspondence. For example, a black to blue transition q_{15}, which yields to an edge value of $(0, 0, 1)$, occurs again at q_{10}. If the shape of the object ends at the projection q_{10}, the backtracking procedure of the dynamic programming would automatically match with the wrong q_{15}, which also maximizes the optimal solution. A match is found between q_{15} and e_i instead of q_{10} and e_i. This ambiguity leads to an uncontrollable bad triangulation. This is due to the greedy path backtracking of the cost matrix. We could have overcome this particular case by not choosing a match when ambiguity occurs. But the problem would reappear on the other side of the object as shown in figure 4.10 and 4.12. The problem lies in the lack of stripe information caused by the empty background of the shape and of potential shadows. We therefore propose a solution that reduces the effect of the empty background, by putting a plane wall behind the object during the stripe projection and caption. The wall is removed during the white light projection in order to produce a bitmask that identifies the object from the background and purges the image from pixels under shadow. In this way, the previously missing stripes can be recovered on the wall and will fix the ambiguity problem. Now the labeling process is now no longer limited to unsuitable bounding shapes of the object. This method requires an additional scan with the white light projection, and fortunately, we can reuse the scan performed previously during the color correction stage.

2.6 Subpixel Accurate Repositioning

For the optical triangulation pass, we use our previously labeled edges that were detected on the input images. These input images are pixel arrays recovered from the CCD of the camera. It is obvious that from a pixel accurate edge detection, we are only able to triangulate the image at pixel accuracy. This imprecision is manifested by jagged aliasing effects when the final object is zoomed in. In order to produce high precision scanned results, many approaches have been suggested. Zhang et al. [ZCS02] adapted the space time analysis technique on this colored structured light approach with the disadvantage of requiring multiple input images of the shifted De Bruijn pattern. We will explain in this technical report how it is possible to detect colored edges with subpixel accuracy from a single structured light projected image. This will consequently produce a subpixel accurate triangulated range map.

An edge that has been detected between two pixels will typically be positioned exactly in the middle of the center of these two pixels. This is a subsampling approximation, but in reality, the edge is not always located exactly in the middle. In the monochromatic case for example, let us consider an edge of a transition with intensity from 0 to 1. If the edge is situated nearer to the center of the left pixel, we can assume that the left pixel will appear slightly brighter as compared to when the edge is located exactly in the middle. Similar to the linear boundary interpolation approach described in [MV97], we use this observation to approximate a subpixel accurate position for the edges, by exploiting the light intensity of these pixels. Let $f(n)$ be the intensity of the pixel n in an one-band image and an edge being detected between $f(n_e)$ and $f(n_e+1)$. $\triangle f(n) = f(n+1) - f(n)$ is the local contrast between these two pixels. We have defined in section 2.4 that an edge is detected when this value is a local maximum of the signal $f(n)$. The laplacian of $f(n)$ would have a transversal zero-crossing at n_e. Comparing with an imaginary non-discretized image $f(x)$, the edge would be located at the exact zero-crossing of the laplacian of $f(x)$, which is not the pixel position n_e. We therefore approximate this subpixel position by considering the gradient of the contrast function $\triangle f(x)$ to be a piecewise linear polynomial that is achieved by linearly interpolating the subsampled pixels from $f(n)$. The subpixel position of the edge can now be computed by calculating the intersection of the piecewise linearly approximated function of $\triangle^2 f(x)$ with the x-axis $g(x) = 0$. Finally, we apply this approximation method to the multi-spectral case by approximating the gradient of the square local contrast of each channel of the captured image $\triangle S(n) = S(n+1) - S(n)$ with a piecewise linear function. In the interval $[n, n+1]$, the approximated curve is given by

$$c(x-n) = S(n) + (S(n+1) - S(n))(x-n) \quad .$$

The intersection of this approximated curve with the abscissae x, yields to the following subpixel accurate position:

$$x = n - \frac{S(n)}{(S(n+1) - S(n))} \quad .$$

2.7 Optical Triangulation

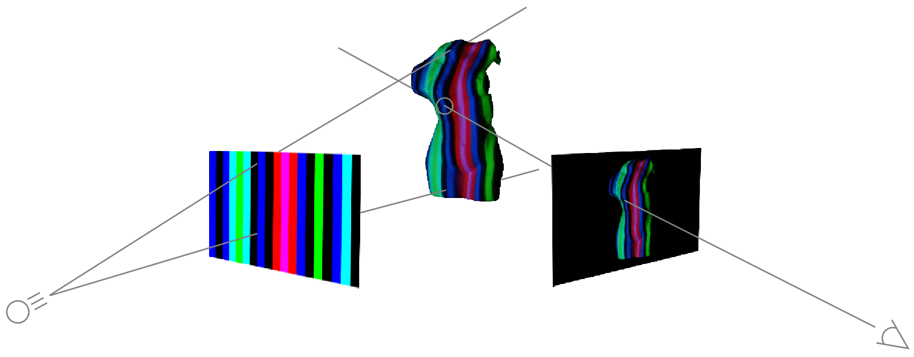


Figure 2.19: An illustration of the optical triangulation. The De Bruijn pattern on the projector's LCD is projected onto the object and acquired by the camera's CCD at a shifted position.

Once we successfully label the detected edges on the captured image with the corresponding stripe transitions on the projected pattern, we are able to compute the depth information of these edges. The vertical stripe transition pattern projected on the object represents a set of vertical planes passing through the central point of the projector. On the other hand, an edge on the sensor image corresponds to, w. r. t. the central point of the camera a line of sight that intersects the projected planes in one single point. This point corresponds to our detected edge in 3-space, yielding to a depth value. It is therefore necessary to know the camera and projector parameter in advance. To be flexible, we choose to keep these parameter values as variables in our calculations, so that a calibration can be independently made without any assumption of the chosen pattern and the configuration of the devices.

The optical triangulation with a stripes pattern has been reduced to a ray plane intersection problem. A classical method, as described in [KSK+98], is to assume perfectly vertical stripes and use the ray-ray-theorem and right triangle trigonometry formulas. Due to the limitations of this solution, we choose a solution using projective geometry that has proven to be an attractive framework for computer vision. Using projective geometry allows us to construct a model that is able to relate world coordinates with pixel coordinates in a homogeneous way. Thus, we do not need to convert the pixel coordinates into world coordinates and can obtain very compact equations to work with. Moreover camera and projector intrinsic and extrinsic parameters can be easily embedded into these equations and boundary conditions such as rays parallel to the image plane, can be excluded.

2.7.1 Projective Extension Of Affine Spaces

The projective space $\mathcal{P}(\mathbf{V})$ of a vector space $\mathbf{V}$ is defined as the set of 1 dimensional spans of $\mathbf{v}$, with $\mathbf{v} \in \mathbf{V}$. If $\mathbf{V}$ is of dimension $n+1$, $\mathcal{P}(\mathbf{V})$ is defined to be of dimension n. We note that $\rho \cdot \mathbf{v} \in \mathcal{P}(\mathbf{V})$, with $\rho \neq 0$. This element is called a point in projective space.

An affine space $\mathcal{A} = (\mathcal{A}, \mathbf{V}, +)$ has two components represented by vector spaces, the point space $\mathcal{A}$ and the vector space $\mathbf{V}$. Both are defined to have same dimensions. Moreover an operator $+ : \mathcal{A} \times \mathbf{V} \to \mathcal{A}$ is defined such that for all $\mathbf{p} \in \mathcal{A}$ and $\mathbf{q} \in \mathbf{V}$, there is exactly one $\mathbf{r} \in \mathcal{A}$ such that $\mathbf{p} + \mathbf{q} = \mathbf{r}$. Moreover the transitivity property holds. To easily handle affine spaces, we embed $\mathcal{A}$ of dimension n into an extended vector space with an additional coordinate, namely the homogeneous coordinate. This extra coordinate has values 0 if it lies in $\mathbf{V}$ and 1 if it lies in $\mathcal{A}$. A vector $\mathbf{v}$ in homogeneous coordinates is denoted $\mathbb{v}$ and a point $\mathbf{p}$ is denoted $\mathbb{p}$.

To construct a projective space $\mathcal{P}$ out of an affine space $\mathcal{A}$ of same dimension n, we exploit this additional coordinate to represent the overlying vector space of dimension $n+1$. We simply multiply the homogeneous component of the affine space element by ρ, a non zero scalar. In this manner we are able to extract all point elements from $\mathcal{P}$. The points with homogeneous coordinate of value 0 now represent all the points that cannot be projected onto a plane that is orthogonal to the axis spanned by the homogeneous coordinate. This plane refers to the ideal hyperplane. We define these points as ideal points and for every other points of non zero homogeneous coordinate, it is defined as a proper point in projective space. All point elements of $\mathcal{A}$ and the ideal points (projective closure of $\mathcal{A}$), are thus uniquely embedded into the projective space $\mathcal{P}$, which is also called the projective extension of an affine space.

The camera model, described more in detail later, is characterized by its extrinsic and intrinsic parameters, which can be described using our previously defined spaces. Our camera is located in world coordinates, namely a 3 dimensional affine

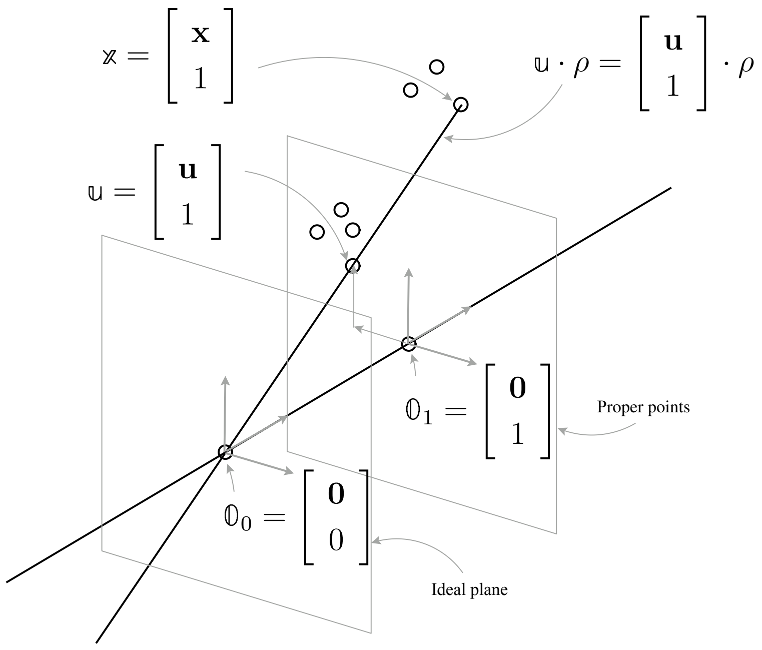

$\mathbb{x} = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}$
$\mathbb{u} \cdot \rho = \begin{bmatrix} \mathbf{u} \\ 1 \end{bmatrix} \cdot \rho$
$\mathbb{u} = \begin{bmatrix} \mathbf{u} \\ 1 \end{bmatrix}$
$\mathbb{O}_1 = \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix}$
Proper points
$\mathbb{O}_0 = \begin{bmatrix} \mathbf{0} \\ 0 \end{bmatrix}$
Ideal plane

Figure 2.20: A scheme of the projective extension of an affine space. The image plane of the camera model can be interpreted by the proper points of $\mathcal{A}^2$. A point $\mathbb{u}$ in this $\mathcal{A}^2$ represents a line of sight in $\mathcal{P}(V^3)$, which also lies in $\mathcal{P}(V^4)$. $\mathcal{P}(V^4)$ is the projective extension of $\mathcal{A}^3$ in which the triangulated point $\mathbb{x}$ lies

space (for extrinsic parameter). The unknown point in the scene that we are scanning also lies in this space. In order to triangulate, this point is extended to a ray of sight by considering the projective extension of this affine space. This ray is mapped onto the projection plane of our camera in a 2 dimensional projection extension of the pixel coordinates, namely a 2 dimensional affine space. Subsequently we use this 2 dimensional affine coordinate system to describe the projection plane properties and the focal length (intrinsic parameters).

The following scheme illustrates the relation between the spaces.

$$\underbrace{\mathcal{A}^3 = (\mathcal{A}^3, \mathbf{V}^3)}_{\text{for extrinsic parameters}} \quad \sim \mathcal{P}(\mathbf{V}^4) \to \mathcal{P}(\mathbf{V}^3) \sim \quad \underbrace{(\mathcal{A}^2, \mathbf{V}^2) = \mathcal{A}^2}_{\text{for intrinsic parameters}}$$

2.7.2 Camera And Projector Model

The camera and projector model are commonly described by an ideal pinhole model. Both models describe how a point in pixel coordinates is transformed into a ray in world coordinates w. r. t. their parameters as mentioned before. To simplify matters, we ignore the effects of lens distortion in our system. We define the points $\mathbb{p} = [u', v', 1]$ and $\mathbb{p} = [u, v, 1]$ in pixel coordinates and $\mathbb{x} = [x, y, z, 1]$ a point in world coordinates.

2.7.2.1 Mapping Function

Let us first examine the mapping of the camera model, for which the camera center is placed at the origin of the world coordinate system. The aim is to map a point $\mathbb{x} = [x, y, z, 1]$ of a 3 dimensional affine space onto a point $\rho \cdot \mathbb{u} = [u, v, 1]$ in a projective extension of a 2 dimensional affine space, as they represent the ray of sight by means of affine geometry. Before considering the intrinsic parameters, we first map onto a point $\rho \cdot \mathbb{u}' = [u', v', 1]$. We therefor use the mapping matrix $\mathbb{A}$:

$$\rho \cdot \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\mathbb{A}} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad . \tag{2.7}$$

We note that if the projection(or image) plane is the plane $z = 1$, we obtain $u' = \frac{x}{z}$, $v' = \frac{y}{z}$, which is a particular case of a simple central projection, that can also be deduced from the ray theorem.

2.7.2.2 Intrinsic Parameters

The internal parameters of the camera are a set of transformations applied in the pixel coordinates which is the 2 dimensional affine space of the overlying projective extension. For example by using a focal length f other than 1, which is the distance from the camera center to the projection plane, we must scale down the vector $\mathbb{u}'$ in the equation 2.7. In addition, the CCD properties of the camera are unknown before calibration. We must determine the image input coordinates $\mathbb{u} = [u, v, 1]$

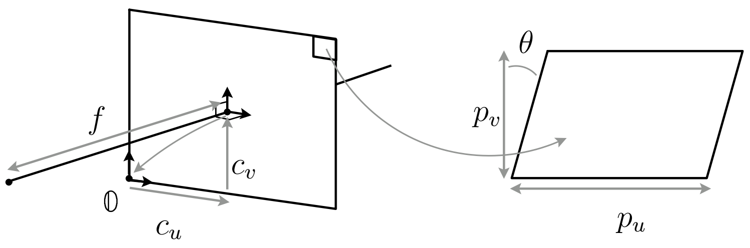


Figure 2.21: On the left: a representation of the focal length f and principal point coordinates $[c_u, c_v]$, which is responsible for centralising the acquired image's origin. In practice, the acquired image has its origin on the top left of the image. On the right: a pixel of width p_u, height p_v and skew angle θ.

w. r. t. the physical position $\mathbb{u} = [u', v', 1]$ of the CCD in the device, the size and the shape of the pixels, which is given by:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{f}{p_u} & (\tan\theta)\frac{f}{p_v} & c_u \\ 0 & \frac{f}{p_v} & c_v \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbb{K}} \cdot \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} \quad .$$

with p_u and p_v the pixel's width and height of the CCD array. $\mathbf{c} = [c_u, c_v, 1]$ is called the principal point of the camera, which is the intersection of the optical axis with the image plane. θ is called the skewing angle of a pixel, which is the difference between the angle of the retinal axes and an orthogonal angle. In practice, we can assume it to be equal to $\frac{\pi}{2}$. The Matrix $\mathbb{K}$ depicts the calibration matrix of the camera. We note $\mathbb{K}$ to be a combination of a translation of vector $\mathbf{c}$, a shearing with a coefficient $\tan\theta$, a scaling w. r. t. the pixel sizes p_u and p_v, and scaling w. r. t. the focal length f of the camera, which is conversely linear to the input coordinate. All these are basic 2D affine transformations, which are described with 3×3 matrix transformations with homogeneous coordinates:

$$\mathbb{K} = \mathbb{M}_{\text{principal point translation}} \cdot \mathbb{M}_{\text{focus scaling}} \cdot \mathbb{M}_{\text{pixelsize scaling}} \cdot \mathbb{M}_{\text{shearing}} \quad .$$

2.7.2.3 Extrinsic Parameters

We have previously assumed that the center of the camera is at the origin of the world coordinates. Therefore, the extrinsic parameters simply consist of a combination of movements in the corresponding 3 dimensional affine space to position and orientate our camera. The motion of objects in the scene are seen conversely, if we describe with the camera's motion. For instance, if an object moves to the left, it would be the same as if the camera moves to the right. Moreover, these movements are represented by 2 affine transformations, namely rotation (for the camera orientation) followed by translation (for the camera position). These transformation are also described with extended matrices with homogeneous coordinates. In this case, we have 4×4 matrices as we are in a 3 dimensional world coordinates and they are given by a concatenation of inverse rotation matrices (we use the camera motion) $\mathbb{R}^t$ around each axis and the inverse translation matrix. The camera is oriented with $\mathbb{R}$ and positioned with $\mathbb{T}$, which yields to the following transformations of the points in the scene:

$$\mathbb{R}^t = \mathbb{R}_x^t \cdot \mathbb{R}_y^t \cdot \mathbb{R}_z^t$$

with

$$\mathbb{R}_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x & 0 \\ 0 & -\sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbb{R}_y = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbb{R}_z = \begin{bmatrix} \cos\theta_z & \sin\theta_z & 0 & 0 \\ -\sin\theta_z & \cos\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} .$$

The inverse translation matrix $\mathbb{T}^{-1}$ of translation vector $-\mathbb{t} = [-t_x, -t_y, -t_z, 0]$ is given by:

$$\mathbb{T}^{-1} = \begin{bmatrix} 1 & 0 & 0 & -t_x \\ 0 & 1 & 0 & -t_y \\ 0 & 0 & 1 & -t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} .$$

Hence, the camera orientation matrix $\mathbb{M} = \mathbb{T}^{-1} \cdot \mathbb{R}^t$ is given by the following transformation:

$$\mathbb{M} = \begin{bmatrix} R^t & -t \\ \mathbf{0}^t & 1 \end{bmatrix} .$$

2.7.2.4 The Projection Matrix

The camera model can be summarized as

$$\rho \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbb{K}^{-1} \cdot \mathbb{A} \cdot \mathbb{M} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad ,$$

or simply

$$\rho \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbb{P} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

with $\mathbb{P} = \mathbb{K}^{-1} \cdot \mathbb{A} \cdot \mathbb{M}$ the 3×4 projection matrix, which comprises all parameters required for the calibration of the camera.

2.7.3 Ray-Plane Intersection Using Projective Geometry

We recall that by projecting a stripe transition on the target object, with the camera at a shifted position, we can capture a set of points in the sensor image plane corresponding to the distorted stripe. Each point on this image corresponds to a point reflected from the object. We know that these points are in the line of sight of the camera as well as in the plane represented by the stripe transition projection as shown in figure 2.19. We use the advantages of our previously described camera model which uses projective geometry to formulate a compact linear equation that has to be solved in order to extract the depth information from our acquired image. All necessary parameter information of the camera as well as of the projector are naturally embedded in this equation.

We recall that the camera's model is given by:

$$\rho_c \cdot \mathbb{p}_c = \mathbb{M}_c \cdot \mathbb{x}_c$$

with $\mathbb{x}_c$ a point on the ray of sight in $\mathcal{A}$. Respectively, the projector's model is given by:

$$\rho_p \cdot \mathbb{p}_p = \mathbb{M}_p \cdot \mathbb{x}_p \quad .$$

In addition, we have assumed, that the projector projects perfectly vertical stripes. Thus, an explicit representation of a stripe at position p in the horizontal axis in pixel coordinates is given by:

$$\mathbb{p}_p(\lambda) = \begin{bmatrix} p \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ \lambda \\ 0 \end{bmatrix} \quad .$$

Thus, let $\mathbb{e}_1 = [1, 0, 0]^t$ be a unit vector in the horizontal axis in pixel coordinates. We obtain an implicit representation:

$$\begin{aligned} & \mathbb{e}_1^t \cdot (\mathbb{p}_p - \begin{bmatrix} p \\ 0 \\ 1 \end{bmatrix}) = 0 \\ \Leftrightarrow \quad & \mathbb{e}_1^t \cdot \mathbb{p}_p = p \\ \Leftrightarrow \quad & u_p + 0 \cdot v_p - p = 0 \qquad . \\ \Leftrightarrow \quad & [1, 0, -p] \cdot \begin{bmatrix} u_p \\ v_p \\ 1 \end{bmatrix} \\ \Leftrightarrow \quad & \mathbb{n}^t \cdot \mathbb{p}_p = 0 \end{aligned}$$

with $\mathbb{n}^t = [1, 0, -p]$.

The projector model describes a line of sight between the pixel coordinates and the world coordinates. In order to describe a plane of sight, as we are projecting stripes, we embed the implicit definition of the stripe in pixel coordinates in the projector model as follows:

$$\begin{aligned} & \mathbb{p}_p = \tfrac{1}{\rho_p} \cdot \mathbb{M}_p \mathbb{x}_p \\ \Leftrightarrow \quad & \mathbb{n}^t \cdot \mathbb{M}_p \cdot \mathbb{x}_p \tfrac{1}{\rho_p} = 0 \qquad . \\ \Leftrightarrow \quad & \mathbb{n}^t \mathbb{M}_p \mathbb{x}_p = 0 \end{aligned}$$

We now set $\mathbb{x}_s = \mathbb{x}_p = \mathbb{x}$ to compute the ray-plane intersection:

$$\left.\begin{array}{l} \mathbb{n}^t \mathbb{M}_p \mathbb{x} = 0 \\ \mathbb{M}_c \mathbb{x} = \mathbb{u}_c \end{array}\right\}$$

$$\Leftrightarrow \begin{bmatrix} \mathbb{n}^t \mathbb{M}_p \\ \mathbb{M}_c \end{bmatrix} \mathbb{x} = \begin{bmatrix} 0 \\ \rho_c \mathbb{u}_c \end{bmatrix} .$$

$$\Leftrightarrow \begin{bmatrix} \mathbb{n}^t \mathbb{M}_p \\ \mathbb{M}_c \end{bmatrix} \mathbb{x} \frac{1}{\rho_c} = \begin{bmatrix} 0 \\ \mathbb{u}_c \end{bmatrix}$$

The position of $\mathbb{x}\frac{1}{\rho_c} = \mathbb{w}$ is computed by solving this linear equation and $\mathbb{x}$ can be deduced by dividing $\mathbb{w}$'s coordinates with its coordinate extension such that its coordinate extension becomes 1.

2.8 Artifact Elimination Through Meshing

The previous sections allow us to reconstruct a range map from the captured image of an object, that is illuminated by a structured light. Usually, the output is unsatisfactory. In our experiments, we observed that most areas of the range map are seriously affected by noise. Wrong triangulated points are scattered all over the object. This is due to the difficulty in detecting the projected stripe edges in high frequency shading areas of the image, that yields to bad labeling and subsequently wrong triangulation. We therefor designed two post-processing algorithms to annihilate the triangulated points that are most probably not part of the object. Although this might cause more untriangulated points, i. e. more holes in the end result, it is more important to acquire less but correct data than wrong ones.

2.8.1 Meshing Using Context Knowledge Of The Mesh Topology

Our relatively high resolution images (1500 × 1000 pixels) produce a dense set of scattered discretizations all over the visible shape of the object. Many of the triangulated ranges are still wrongly labeled, due to estimation failure during the dynamic labeling and also due to other sources of noise. We must therefor perform another step of noise reduction. Assuming that the points belong to the surface of the object, we further assume that a point is valid if and only if it has two neighboring points to produce a triangle surface. Horizontal neighboring points are those which belong to two consecutive stripe transitions and vertical points are those triangulated from the next pixel row. Our experiments have demonstrated that using the connectivity of the points (that is the mesh topology), has improved the results, as shown in figure 4.15 and 4.16. This yields to algorithm 2.4.

```
procedure artifactEliminationThroughMeshing(match_data)
for  rows=0 to number of rows-1
  for  column=0 to rowLength
    if (match at row happens before match at row+1)
      if (match upper left has less than 2 neighbors)
        remove match and its neighbors.
    else
      if (match lower left has less than 2 neighbors)
        remove match and its neighbors.
  end for
end for
```

Algorithm 2.4: Artifact elimination through meshing procedure.

with `match_data` a data structure implemented as a 1D array of lists. The 1D array of `match_data` represents the rows of the input image. Each element of this array holds a chained list of matches as shown in the following illustration:

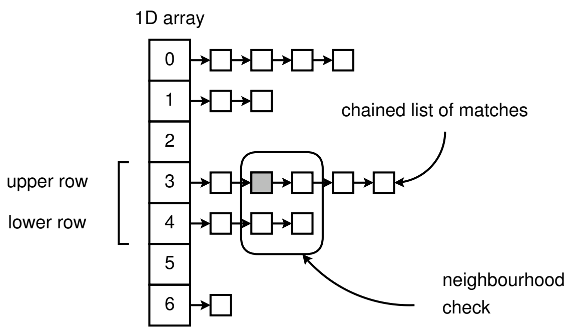


Figure 2.22: In this example, the algorithm checks if the 2nd match at row 3 has any neighbors. If the 3rd match has a stripe transition corresponding to the next stripe transition of the 2nd match, it would be a neighbor.

All possible neighboring matches are illustrated in figure 2.23, depending on which match happens first (upper row or lower row). If the scanline through the chained list, finds a match that first happens in the upper row, i. e. the stripe transition of the upper row's match happens before the lower row one, a neighborhood check is performed on the upper right, lower left and lower right matches, if they exist.

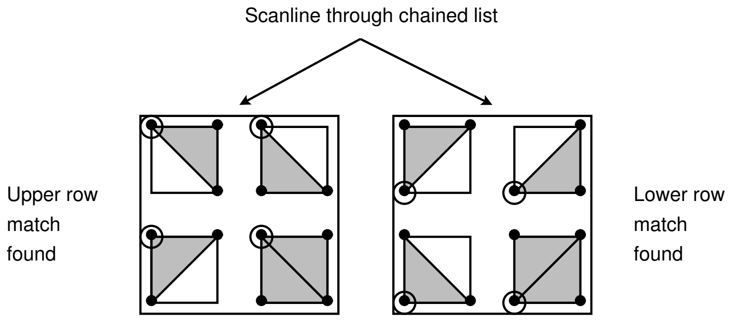


Figure 2.23: All possible neighboring matches, depending on whether a match occurs first in the upper or lower row.

2.8.2 Face orientation culling

The previous procedure actually generates a triangle mesh. Using this mesh, the task now is to remove those triangular surfaces that are not visible to the camera and the projector. This is simply a backface culling algorithm that computes the scalar product of the triangle surface normal with the normalized direction of the line of sight to the center of this triangle. Thereafter, negative values are omitted. It actually suffices to consider only one of the triangle corners as the center. Furthermore, we leave out surfaces with normals that are almost orthogonal to the direction of the line of sight. This can be parametrized with the corresponding angle or scalar product value. In practice, depending on the shape of the object, an angle of up to 60 degrees between the surface normal and the normalized line of sight vector can be reasonable.

Chapter 3

The Implementation of LIThIum

Within the framework of building a low cost 3D scanner, we have developed an experimental software that is responsible for converting 2D input images into 3D range maps using a calibrated triangulation system. Li ti, which means 3D in Chinese, yielded to the name of LIThIum as it is supposed to be a simple software depicting the light weight of the element. It is only one of several key stages of the whole system such as the calibration stage, the registration, meshing, surface fitting and texture reconstruction. The aim is to design an experimental environment that ensures its functional purpose of generating a customizable De Bruijn Pattern, reading in large input images and computing a point cloud in an appropriate file format suitable for the registration stage. Moreover, the ability to control thresholds was necessary, because we were processing images, behaving in a statistical model, during the color correction, the noise sensitive edge detection and the correspondence stage. In order to be completely independent from the automatic calibration process, a manual calibration was temporally adapted. Finally, for ease of performing experiments, a GUI has been implemented, including a 3D visualization.

3.1 Our Experimental Environment

The necessary hardware devices are a video projector, a professional digital camera and a standard PC. All these components are consumer level technology, allowing an affordable solution. We experimented with two video projectors, the Epson EMP-7250 LCD-projector with analog video input and the ASK M2 with DLP technology and digital video input. Both have a 24 bit color resolution and display patterns of 1024x768 pixels (XGA). Although the EMP-7250 provides better colors, we realized that the ASK M2 is be more appropriate for our acquisitions due to the sharper and denser pixels on the reflected object. Anyway, the images are color corrected subsequently. The selected camera is the FujiFilm FinePix S2 Pro, that

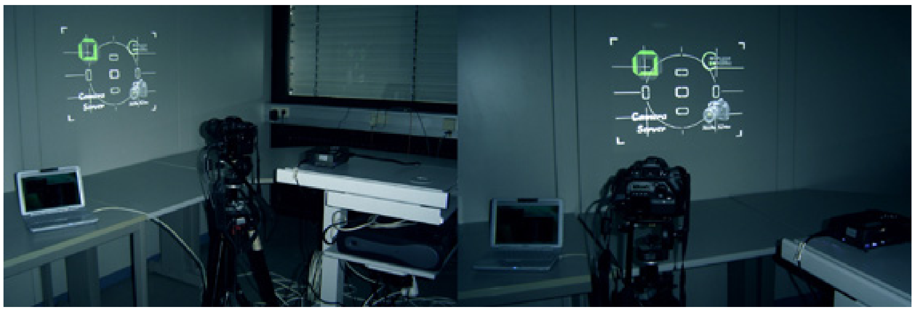


Figure 3.1: The optical triangulation devices.

can provide images with sufficient resolution (up to 6 megapixels) and quality, as uncompressed raw images in TIFF format could be saved. Moreover, the aperture of the camera can be controlled and this important to acquire sharp edges from the reflected object, especially when the whole scanning process needs to be done in total darkness. To control these two devices, we use a camera-projector server which is implemented separately on another PC. Thus, a client program on any platform could remotely send the structured light pattern in PNG file format [liba] to this server and request for the capture from the camera as an uncompressed TIFF [Libb] file. Both PCs have Pentium 4 processors with 2.4 GHz, 512 MB RAM and ATI Fire GL graphics cards.

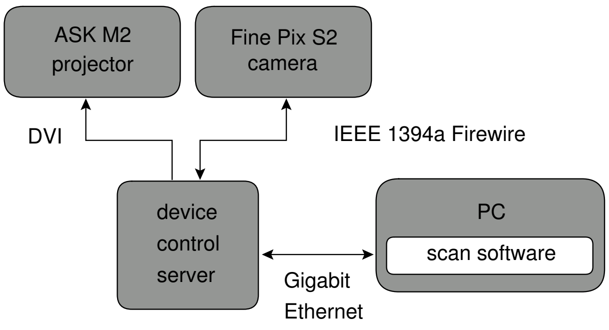
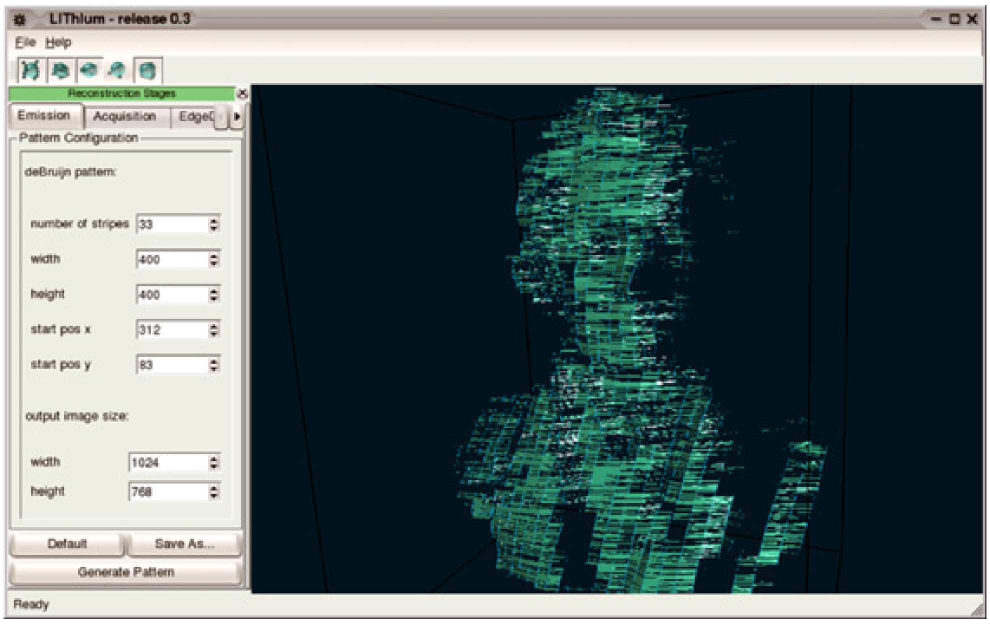


Figure 3.2: An overview showing how the devices are interconnected to form the optical triangulation system.

The software is developed on a Linux platform and the code is entirely written in C/C++. For efficient coding, we use generic data structures from the STL (Standard Template Library) [STL]. The pattern generator requires libpng and the input images are read into memory using LibTIFF [Libb]. For typical linear algebra operations, such as matrix multiplications and solving linear systems, the GSL (gnu scientific library) [GSL] proved to be sufficiently efficient for the optical triangulation computations. To store the 3D range map output in memory, we use OpenMesh's data structure [Opeb] and reuse its OFF file format writer. Trolltech's QT satisfies our implementation of the GUI because of its ease of usage and OpenGL [Opea] support that was necessary for the visualization of the final result.

3.2 Functional Requirements



Pattern Configuration

deBruijn pattern:

number of stripes 33
width 400
height 400
start pos x 312
start pos y 83

output image size:

width 1024
height 768

Figure 3.3: A screenshot of our implemented experimental software environment ”LIThIum”. Top: the visualization toolbar. Left: The side bar with tab browsing for each stage in the pipeline. The main window is the OpenGL viewer for 3D visualization of the triangulated point cloud.

Typical for a software engineering process, it is useful to make a detailed user requirements checklist to meet the satisfaction of its purpose as an experimental 3D scanning environment. The end product is a X11 application running under Linux but porting for other platforms should be possible without much effort. All stages of the acquisition process are independent programs (modules) that can be run in the command line. The graphical interface is also separately implemented to integrate these stages together. To ensure a convenient workflow, each module represents a tab in the sidebar, containing spin boxes, check boxes and other widgets for the configuration of each stage. For instance, the De Bruijn generator can be configured to generate structured light patterns based on variable numbers of stripes, variable size and position of the pattern in the image and variable resolution of the output image. The acquisition process must read the captured image under stripe pattern, white and black light illumination in TIFF format. To keep it simple, the image pre-processing stage and the edge detection stage are integrated into one tab. A threshold value can be set to purge the image from pixels under shadow and the background. An optional color cross-talk and smoothing pass can be performed by marking the checkboxes. Our special multi-spectral edge detection requires, in addition to its threshold, another parameter to restrict the number of consecutive pixels. The labeling technique does require, as mentioned, the alpha and beta values for the soft thresholding. Finally, before our computed ranges are visualized in the OpenGL display window, a triangulation tab has to be configured. More specifically the parameters of the camera and projector have to be entered. The additional post-processing, namely the artifact reduction through meshing, also requires a threshold. This stage is also included into the triangulation tab. The 3D visualization of the point cloud in the OpenGL display can be rotated, translated and zoomed in and out. An additional toolbar is added to enable or disable visualization features such as the point cloud itself, the triangulated mesh, an illumination from the projector, the camera and projector's position as well as a bounding box of our scanned object. The edge detection and labeling stage have the capability to write an image in TIFF format for analytical purposes. In order to implement these requirements, it is necessary to design a solid architecture as presented in the next section.

3.3 Software Architecture

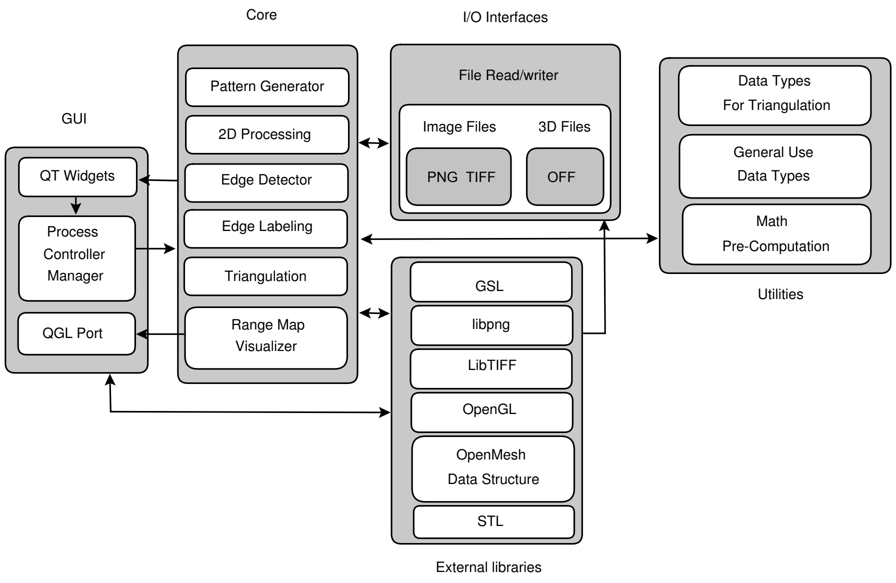


Figure 3.4: LIThIum's software architecture.

The software architecture plays an important role in the implementation by means of modularity, reliability, portability, maintainability and extendability. We therefore developed our implementation in a so-called water-fall model, which is a top down approach. The main task of our program is refined into several subtasks in a modular fashion. In this way, most stages in our previously described shape reconstruction architecture can be easily implemented as small independent programs. This modularity enables us to test multiple techniques for specific problems. For example, we experimented with several edge detectors without worrying about its dependency with other stages. The conceptual software structure is illustrated in figure 3.4. We note that a layer-based scheme is employed for the top level architecture, which is typical for an GUI based application. The GUI communicates through an interface with the Core of the program. The Core itself is represented as a pipeline. The input images are processed in this pipeline and the output is the triangulated range map. Refining this pipeline yields to a set of substages, each assigned with a unit test for effective data analysis and reliability tests during the implementation. Most stages are implemented as static libraries for reuse. For instance, all file read/writing procedures use the same library. The same applies to many reusable classes of data structures. For example, the subpixel accurate edge class, that is designed primarily for edge detection, is reused in the labeling stage. Note that the workflow management procedures are also being integrated into the GUI module as it strongly depends on the graphical interface.

Chapter 4

Results

Our experiments focus on the acquisition of the Beethoven bust as shown in figure 4.1. The complete acquisition process takes less than a minute of computation time. Figure 4.3 shows an edge detection without the consecutive edge pixel restriction procedure, which is applied on the next 3 figures. We observe in figure 4.6 that restricting 10 consecutive edge pixels, is too much for a threshold equal to 0.002. But as shown in figure 4.7, setting the threshold to 0.001 yields again to good results. By setting the threshold to 0, all possible local maximum in the square local contrast of the image signal would appear as shown in the next figure and the detected edges are useless.

Comparing figure 4.9 and 4.10 with figure 4.11 and 4.12 shows that, using the plane background masking, can avoid wrongly triangulated points at the boundary of the object. For instance the left part of the head in figure 4.11, which has additional stripes that were wrongly labeled, hence wrongly triangulated. Moreover, we observe cleaner results when the threshold is set higher for the orientational backface culling, which can also be observed in figure 4.15 and 4.16

Finally, the improvement provided by the sub-pixel accurate repositioning are demonstrated in Figures 4.17 to 4.22.

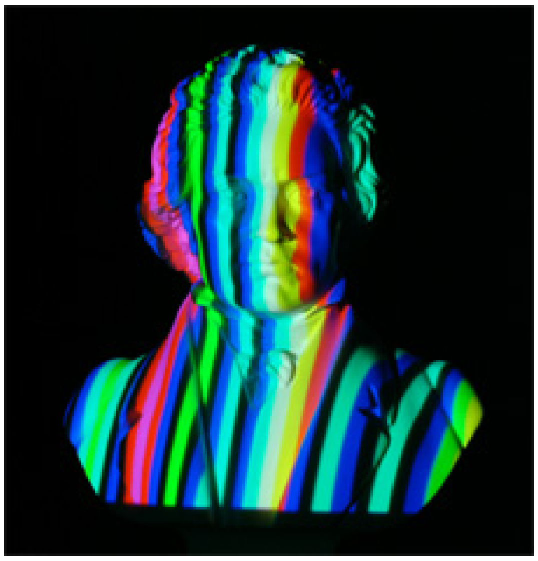


Figure 4.1: Shot of the De Bruijn Pattern illuminated Beethoven Bust. All figures below are results from this photo, except those using the plane background.

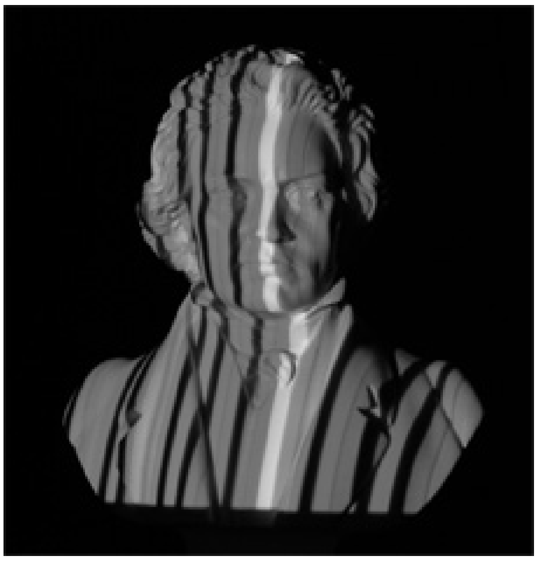


Figure 4.2: If desaturated, many color transitions are no longer visible to edge detection. Cumani's definition of multi-spectral edges is required to solve it.

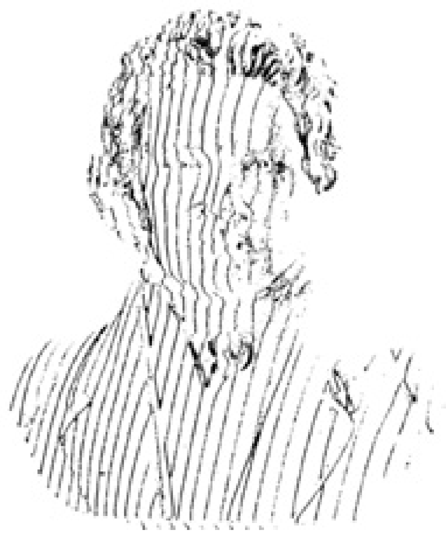


Figure 4.3: Edge detection without consecutive edge pixel restriction and threshold=0.002.

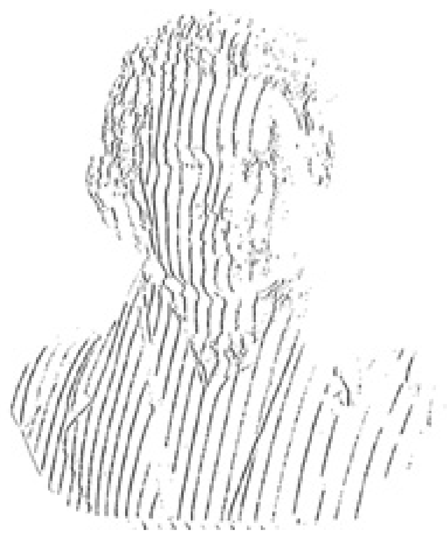


Figure 4.4: Edge detection with restricting 5 consecutive edge pixels and threshold=0.002.

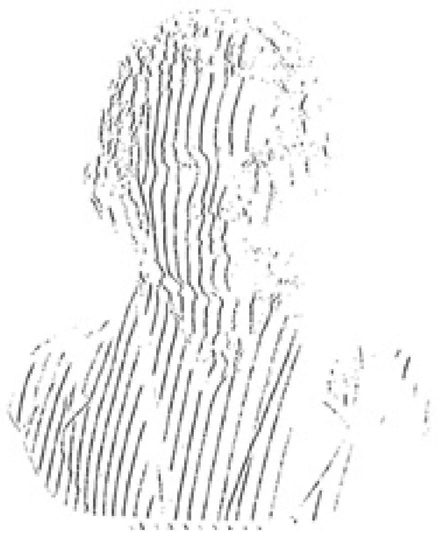


Figure 4.5: Edge detection with restricting 7 consecutive edge pixels and threshold=0.002.

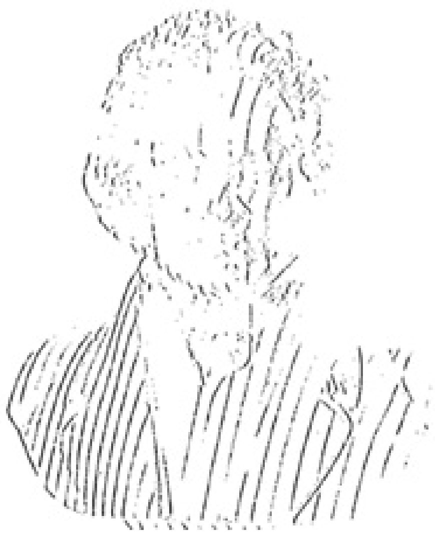


Figure 4.6: Edge detection with restricting 10 consecutive edge pixels and threshold=0.002.

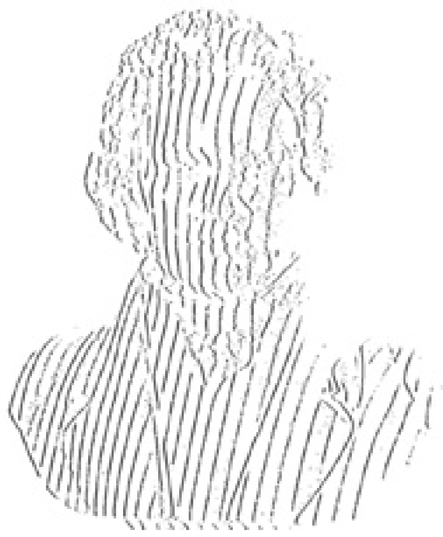


Figure 4.7: Edge detection with restricting 10 consecutive edge pixels and threshold=0.001.

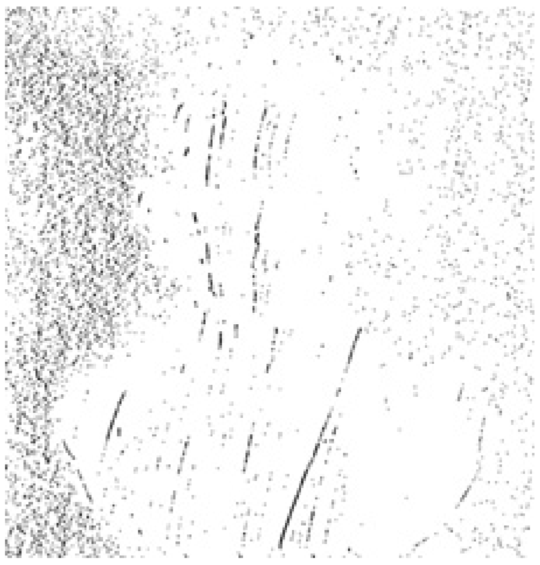


Figure 4.8: Edge detection with restricting 10 consecutive edge pixels and threshold=0.000.

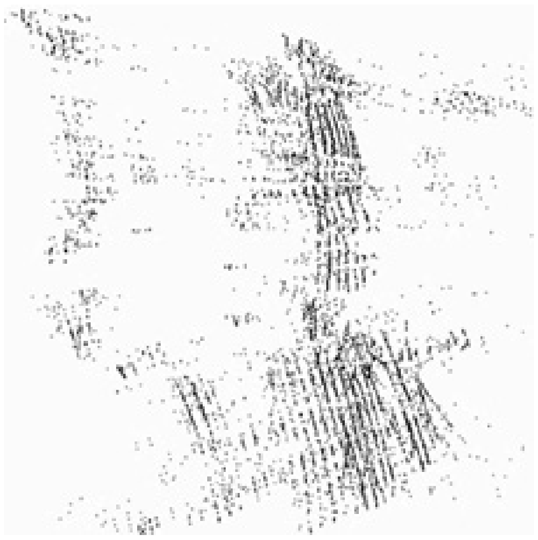


Figure 4.9: Triangulated point cloud without plane background masking and artifact elimination through meshing threshold=0.0.

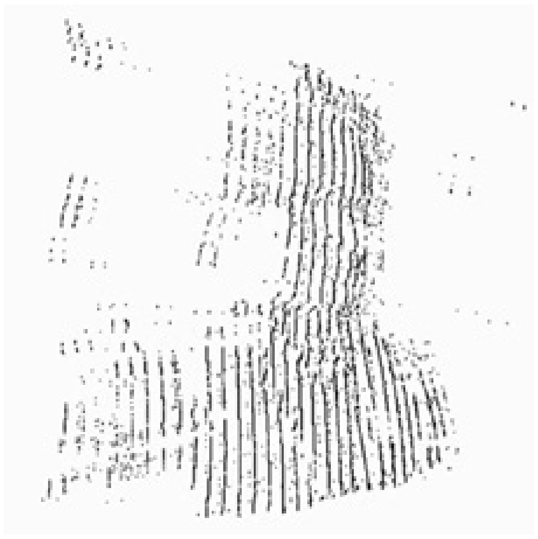


Figure 4.10: Triangulated point cloud without plane background masking and artifact elimination through meshing threshold=0.4.

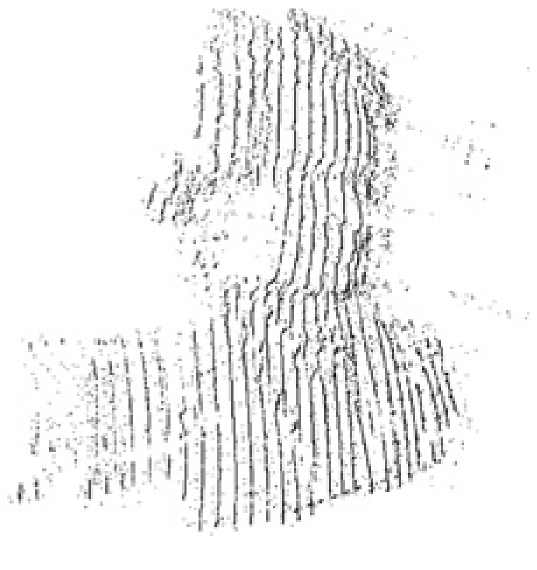


Figure 4.11: Triangulated point cloud with plane background masking and artifact elimination through meshing threshold=0.0.

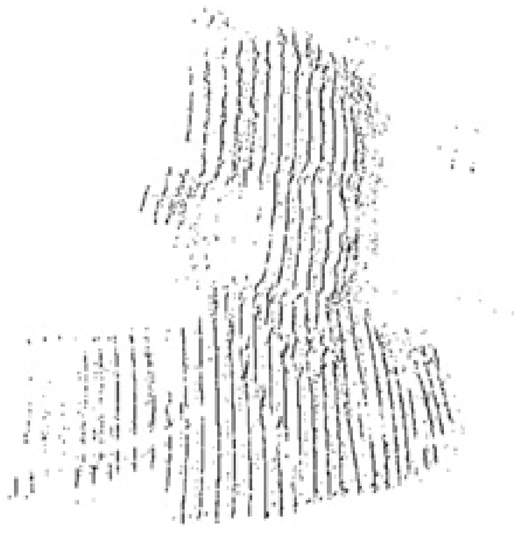


Figure 4.12: Triangulated point cloud with plane background masking and artifact elimination through meshing threshold=0.4.

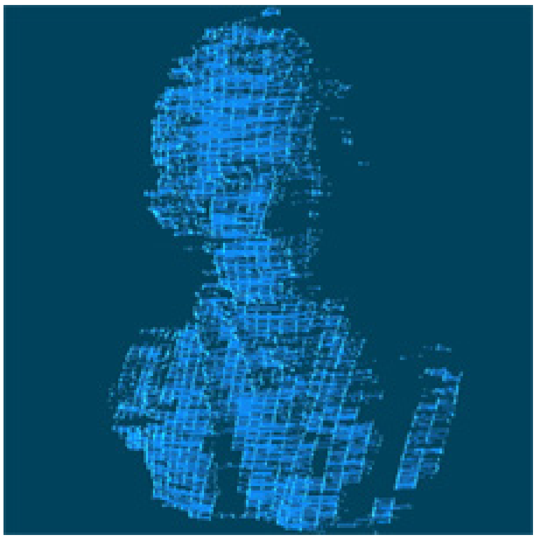


Figure 4.13: Triangulated point cloud with plane background masking and artifact elimination through meshing threshold=0.4 with meshed triangle surfaces.

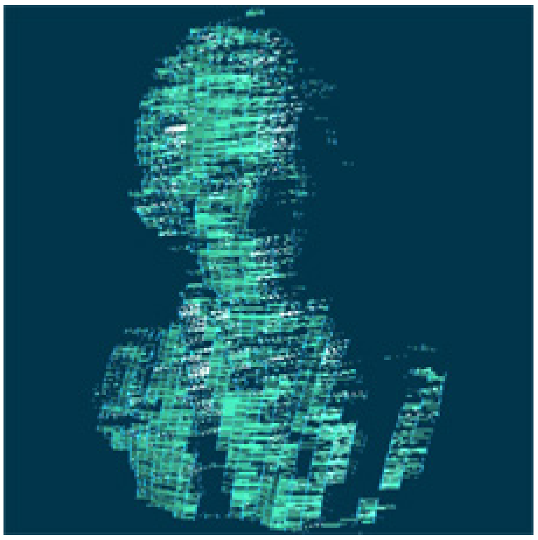


Figure 4.14: Triangulated point cloud with plane background masking and artifact elimination through meshing threshold=0.4 with illuminated triangles.

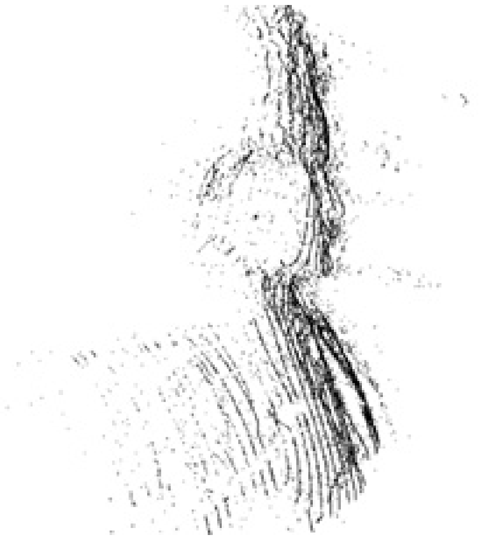


Figure 4.15: Triangulation without orientational backface culling. Point cloud visualization.

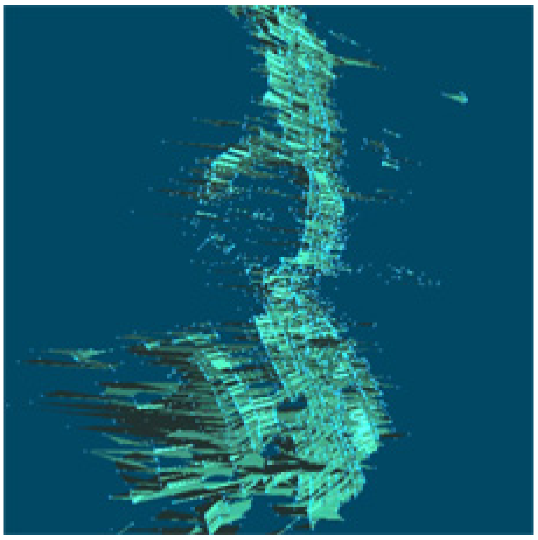


Figure 4.16: Triangulation without orientational backface culling. Triangles visualization.

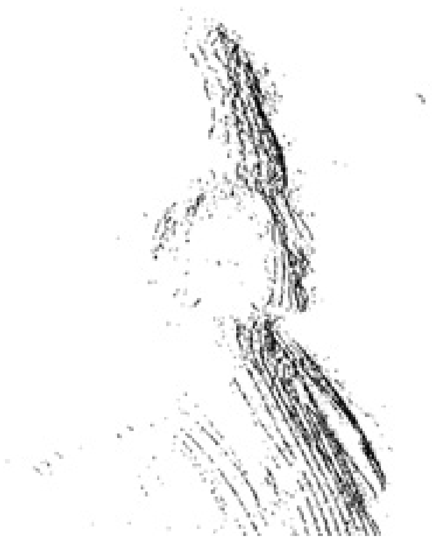


Figure 4.17: Pixel accurate triangulation.

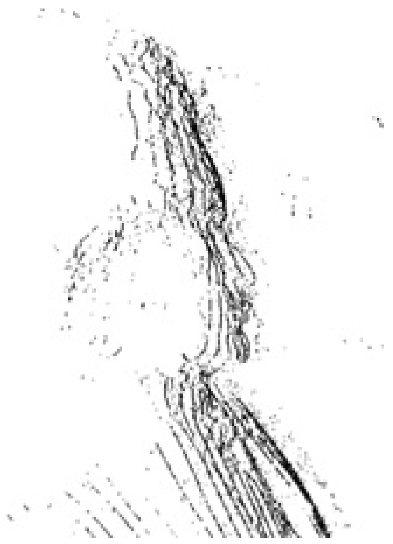


Figure 4.18: Subpixel accurate repositioning.

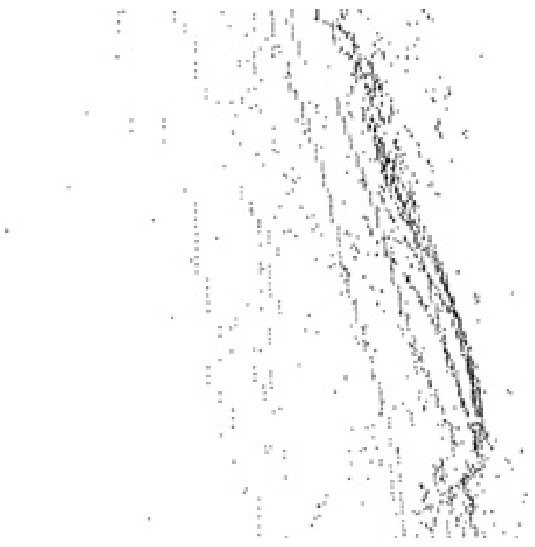


Figure 4.19: Pixel accurate triangulation. Fore head close-up.

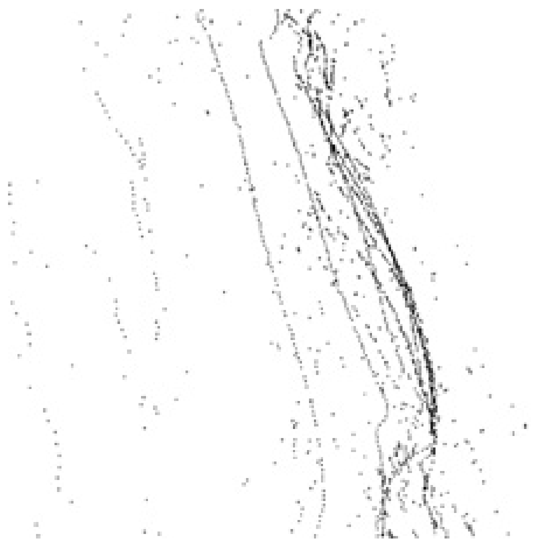


Figure 4.20: Subpixel accurate repositioning. Fore head close-up.

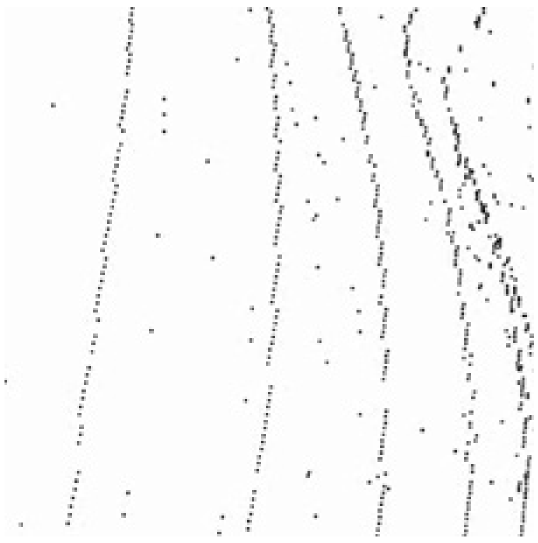


Figure 4.21: Pixel accurate triangulation. Fore head close-up, zoomed in.

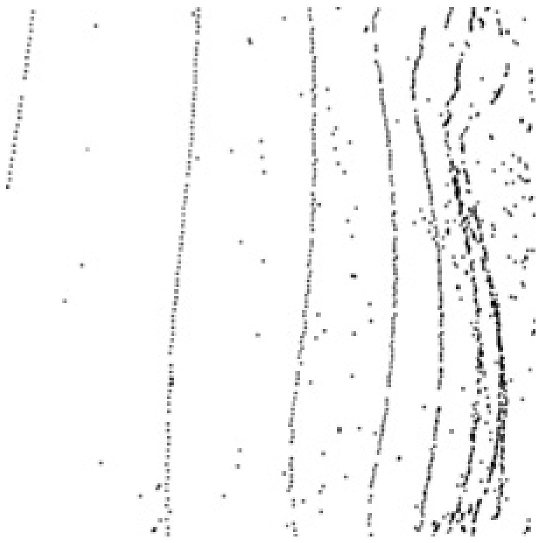


Figure 4.22: Subpixel accurate repositioning. Fore head close-up, zoomed in.

Chapter 5

Conclusion and Future Work

In this work, we have examined the structured light system proposed by Zhang et al. [ZCS02], which uses multi-pass dynamic programming to solve the correspondence problem, and we have performed some modifications to it to achieve subpixel accuracy, less erroneous matching during the labeling and better robustness against noise. Subpixel accuracy from one single frame is attained by approximating the gradient of the square local contrast with a piecewise linear curve. The labeling process is stabilized at the object's bounding by considering a plane background during the pattern projection acquisition. The noise is reduced with an extended edge detector and a post-processing procedure based on meshing and face orientation culling. In addition, an experimental software has been developed in order to independently test different solutions according to subproblems.

Our experimental results have shown a remarkable improvement in terms of subpixel accuracy. While pixel accurate triangulated point clouds show a jagged aliasing effect when zoomed in, an approximated gradient of the square local contrast produces smooth transitions within neighboring ranges.

We also observed that the edge detection, based on a restricted consecutive edge pixel occurrence, is quite effective when it comes to annihilating artifacts in high frequency shading areas. The only drawbacks are the additional parameter that has to be set manually and the few less detected edges. This step is indispensable for a usable stripe edge extraction from complex shape objects.

The post-processing through meshing and face orientation culling also has a radical impact in terms of noise reduction. A large amount of wrongly triangulated ranges can be removed, but also at the cost of an additional parameter and less extracted ranges.

Multi-pass dynamic programming, which solves the matching problem, has achieved satisfactory results except at the boundary of the objects. Due to the greedy property of the backtracking phase in dynamic programming, undesired matches are added for a global optimal solution. These artifacts are either removed by post-processing meshing as mentioned previously or by setting a plane wall behind the object to recover the unreflected stripes on the object. This simple solution enables the dynamic programming to start and end with the correct matches, which also affect the correctness of the matches within the objects surface. However, this solution has the drawback of requiring an additional frame without the background to extract its silhouette. This additional frame is taken during the white light illumination. To circumvent this additional frame, we can think of a depth threshold w. r. t. the camera's position. Every point above a certain threshold is omitted.

While some improvements have been achieved, with several extensions, it is still difficult to detect correct edges when it comes to stripe extraction from the capture. This is due to the high curvature areas in the object and unsuitable material reflection properties. Moreover the projector is only able to project sharp images from a specific distance to the object, in other words, only edges on a plane at a specific distance are sharp.

With the aim of less human inputs and more discretized points, we hope to find a way to automatically determine statistically suitable thresholds by analyzing the image's color and light intensity properties. In addition, we intend to integrate the automatic calibration for the projector and for the camera into the system. We also plan to implement an offline colorimetric calibration for real one-shot pattern acquisition, so that this technique could be applied on moving objects as well as the possibility of registration from video frames by means of a real-time model acquisition. To fill up the empty areas, caused by the relative broad stripe width of the pattern, we would like to implement a pattern shifting acquisition for single view registration. It would also be interesting to include lens distortion to the camera and projector models. Finally, we hope to complete the structured light scanning system with multi-view registration and texture reconstruction stages, both equally important, and explore the possibility of capturing surface reflectance properties.

Acknowledgment

Special thanks go to my supervisor, Raphael Straub, for providing the best conceivable assistance and for believing that there will always be a better solution to a problem, as well as to my lovely girlfriend Yuanshan Lee for reviewing and correcting my report and especially for supporting me during the endless days of implementation. My thanks also go out to my family and all my good friends for their encouragement and for believing in what I do. I would also like to thank the whole Graphics and Geometry Group at the IBDS for the excellent working environment and the great time I had. Last but not least, I want to thank Prof. Dr. Hartmut Prautzsch for giving me the opportunity to be part of the computer graphics community, which is to me a dream come true…

References

[CKS98] Dalit Caspi, Nahum Kiryati, and Joseph Shamir. Range imaging with adaptive color structured light. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):470–480, 1998.

[CL95] Brian Curless and Marc Levoy. Better optical triangulation through spacetime analysis. In *ICCV*, pages 987–994, 1995.

[CS00] Brian Curless and Steven Seitz. 3d photography, acm siggraph '00 course notes, course no. 19, 2000.

[Cum91] A. Cumani. Edge detection in multispectral images. *Computer Vision, Graphics, and Image Processing: Graphical Models and Image Processing*, 53(1):40–51, January 1991.

[GSL] GSL. Gnu scientific library. http://sources.redhat.com/gsl/.

[Kos95] A. Koschan. A comparative study on color edge detection, 1995.

[KSK$^+$98] Reinhard Klette, K. Schluns, A. Koschan, Andreas Koschan, and Karsten Schluns. *Computer Vision: Three-Dimensional Data from Images.* Springer-Verlag Singapore Pte. Limited, 1998.

[liba] libpng. libpng. http://www.libpng.org.

[Libb] LibTIFF. Libtiff. http://www.libtiff.org.

[LPC$^+$00] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The digital michelangelo project: 3D scanning of large statues. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 131–144. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[MV97] Alan M. McIvor and Robert J. Valkenburg. Substripe localisation for improved structured light system performance, 1997.

[Opea] OpenGL. Opengl. http://www.opengl.org.

[Opeb] OpenMesh. Openmesh. http://www.openmesh.org.

[RBMT] Holly Rushmeier, Fausto Bernardini, Joshua Mittleman, and Gabriel Taubin. Acquiring input for rendering at appropriate levels of detail: Digitizing a Pietià. pages 81–92.

[RCM$^+$01] C. Rocchini, Paulo Cignoni, C. Montani, P. Pingi, and Roberto Scopigno. A low cost 3D scanner based on structured light. In A. Chalmers and T.-M. Rhyne, editors, *EG 2001 Proceedings*, volume 20(3), pages 299–308. Blackwell Publishing, 2001.

[Rus00] F. Ruskey. The combinatorial object server. http://www.theory.csc.uvic.ca/cos/, 2000.

[STL] STL. Standard template library. http://www.sgi.com/tech/stl/.

[ZCS02] Li Zhang, Brian Curless, and Steven M. Seitz. Rapid shape acquisition using color structured light and multi-pass dynamic programming. In *The 1st IEEE International Symposium on 3D Data Processing, Visualization, and Transmission*, pages 24–36, June 2002.

[Zha00] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.