FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master Thesis in Informatics

Depth Map Based Superresolution Method in 3D Reconstruction

Gabriel Bernardino

FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master Thesis in Informatics

Depth Map Based Superresolution Method in 3D Reconstruction

Tiefenkartenbasierte Superresolution für 3D-Rekonstruktion

Author:	Gabriel Bernardino
Supervisor:	Prof. Dr. Daniel Cremers
Advisor:	Evgeny Strekalovskiy
Date:	September 17, 2013

I assure the single handed composition of this Master Thesis only supported by declared resources.

Munich, September 17, 2013 Gabriel Bernardino

Ich versichere, dass ich diese Master Thesis selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 17. September 2013 Gabriel Bernardino

Acknowledgments

I wish to thank my advisor Evgeny Strekalovskiy for answering my numerous doubts during the elaboration of the thesis. I wish also to thank Prof. Dr. Cremers and the chair of Computer Vision to allow me to do a Master Thesis with them at the Tecnische Universität München .

I would also like to express my special thanks of gratitude to my brother without him I would not have discovered the pleasures of informatics, as well as to Raúl who served me as inspiration of how to manage a Master Thesis.

Abstract

In this thesis a superresolution method for geometry reconstruction is presented. Superresolution methods have already been used in order to improve texture resolution and RGB images. In a lesser degree they have also been used to improve the quality of depth maps. The objective of this thesis is to validate the hypothesis that superresolution methods can also be used in the geometry reconstruction. In order to do that an algorithm has been implemented and a superresolution operator has been derived and plugged in the algorithm.

Results show that the algorithm produces better quality reconstructions than the original approach without superresolution. Not only upsampling but also deblurring help to obtain better reconstructions. With our method smoother results are obtained without losing geometric details.

Contents

Outline of the Thesis

Part I: Introduction and Theory

CHAPTER 1: INTRODUCTION

This chapter presents an overview of the thesis and its purpose.

CHAPTER 2: THEORY

This chapter presents the basic theory needed to understand this Master Thesis.

CHAPTER 3: RELATED WORK

This chapter reviews published work in thopics related to this Master Thesis.

Part II: Problem statement

CHAPTER 4: PROBLEM DEFINITION

This chapter presents the formal description of the problem and the algorithm used to solve it.

CHAPTER 5: IMPLEMENTATION

This chapter presents the code optimizations and heuristics used during the implementation of the algorithm.

Part III: Results and Discussion

CHAPTER 6: RESULTS

This chapter presents the results of analysing the experimental data.

CHAPTER 7: DISCUSSION AND FUTURE WORK

In this chapter the results of this thesis are presented.

Appendix

CHAPTER A: SOFTWARE COMPONENTS

This chapter presents an overview of the implemented software

CHAPTER B: TESTING AND EVALUATION

This chapter presents an overview of the used data sets and the tools used for testing.

Part I.

Introduction and Theory

1. Introduction

1.1. Motivation

3D reconstruction consists in computing the geometry of a certain object or scene. The input data is generally a set of normal 2D images or 2.5D depth maps.

The applications of 3D reconstruction are quite diverse: geodesy uses the depth information at certain sample points of the surface of the earth in order to generate a 3D model, automatic parking needs to know the geometry of the world in order to park a car, a robot can scan the world and store a 3D representation of it in order to compute a route between the start and the objective position. Medical applications also need 3D reconstruction to create the bone structure of a patient with data coming from a radiography and virtual reality needs real data from the real world to build a virtual one.

Even though some applications as navigation require a quick response time, for others the computing time is not so important but require that results have higher accuracy. If the resolution of the sampling device cannot be increased, a super resolution method can be employed. This method uses redundancy in data in order to generate results with better resolution than the input data given by the sampling device, at expenses of increasing considerably the computing time.

1.2. Aim and Methodology of the Thesis

The main objective of this thesis is to find out whether the effective resolution of the geometry reconstruction of a scene can be increased using superresolution methods in the volume domain or not. To do that, a superresolution operator has to be derived, implemented and coupled to the solver.

The chosen method for solving a 3D reconstruction problem is variational approach. A simple model for 3D reconstruction using depth-maps is derived without using superresolution operator and after that a superresolution operator is added.

The input data are depth maps from a static scene sampled from a Kinect camera with known position and orientation but the method can be used as a component in a pipeline with other components that compute the motion and/or depth from several colour images. This however introduces error made by the others components, making the evaluation of the accuracy of the 3D reconstruction itself difficult,

As volume reconstruction requires heavy memory and computing usage, maintaining memory requirements as low as possible is primordial in order to be able to compute the reconstruction with a normal computer. Computing time optimization is also very important, but subordinated to memory optimization. Several code optimizations, mainly narrow bands and data precomputing, have been implemented in order to obtain a faster algorithm.

2. Theory

2.1. Domains and Camera Model

Here the camera and domain models are presented.

2.1.1. Coordinates and domains

The world is modelled as a compact $\Omega_v \subset \mathbb{R}^3$ that contains the object or scene to reconstruct. The points of the world can be expressed as 3-D vectors. The object to reconstruct is noted as $S \subset \Omega_v$. It should be a closed set of bounded perimeter.

An image can be seen as a mapping between each point of the image domain and a color intensity. For example, a color image can be viewed as a map between the image domain Ω_v and $[0..1]^3 \subset \mathbb{R}^3$, where the 3 reals color expressed encode with the RGB format . The domain of the image is modelled as a subset of the projective plane. For simplicity it is supposed that all the images have the same geometry, so they can be modelled with the same domain. The domain is noted as $\Omega_p \subset \mathbb{P}^2_{\mathbb{R}}$. The image points are expressed as projective points with coordinates $[u : v : t]$. This points do not have a unique representation but $[u : v : t] = [u' : v' : t']$ if $u = ku', v = kv', t = kt']$. The points with the last coordinate different to zero admit a unique representation of the form $[x : y : 1]$.

2.1.2. Camera model

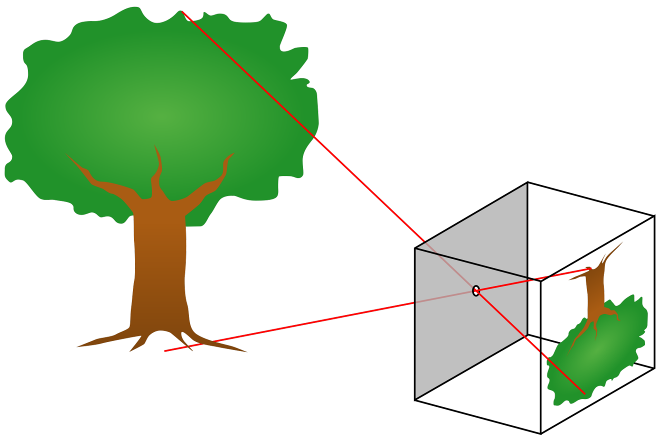
The chosen model for modelling the mapping of the world in a image is the Pinhole Model. This model is based on projective geometry. With this model the camera has a center c, a principal axis and an imaging plane P. The center is a point in the space, the principal axis an oriented line that goes through the center and the imaging plane a plane perpendicular to the principal axis at a distance $f > 0$ of the center.

All the world but the plane parallel to the image plane that goes through the center can be mapped in the image plane, and this making will result in the image. The procedure to project a point x in the image plane is simple. Just take the line $l = x + c$ and the projection is $l \cap I$. This is well defined if x is not in the plane that contains the origin and is perpendicular to the principal axis. In figure 2.1 a graphical example can be seen.

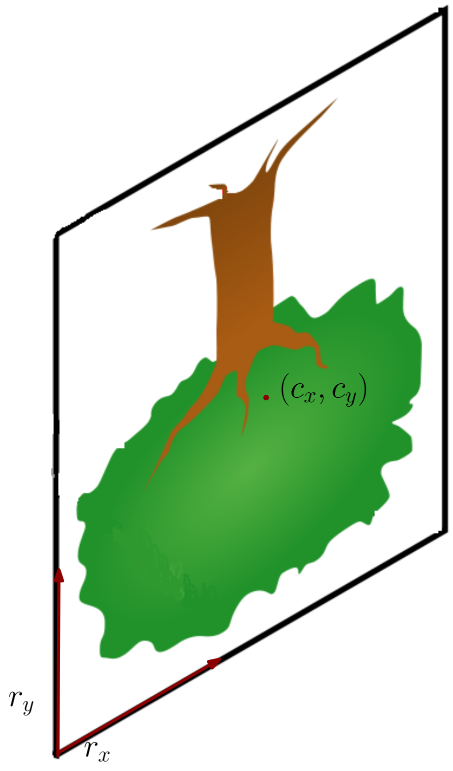
When the camera center is at the origin, and the principal axis is the z axis, this projectivity can be expressed as a matrix with the following form:

$$M = \begin{pmatrix} fr_x & 0 & r_x c_x \\ 0 & fr_y & r_y c_y \\ 0 & 0 & 1 \end{pmatrix}$$

This leaves the result in image coordinates, which has the center in $[-c_x : -c_Y : f]$ and vectors $e_u = (1/r_x, 0), e_v = (0, 1/r_y)$. As the result is an element of a projective space, it



(a) Pinhole camera model. Image taken from [26]



(b) Image coordinates. Image taken from [26] and modified

Figure 2.1.: The upper figure represents the pinhole camera model. The little hole in the box is the projection center, and the back face of that box is the image plane. The lower image is the image coordinates. The point is the intersection with the central axis, and r_x and r_y the vectors.

is not affected by non-zero scalar multiplication, so the matrix can also be multiplied by a non-zero scalar and it will produce the same projectivity. In figure 2.2 there is an example of the effect of applying a projectivity.

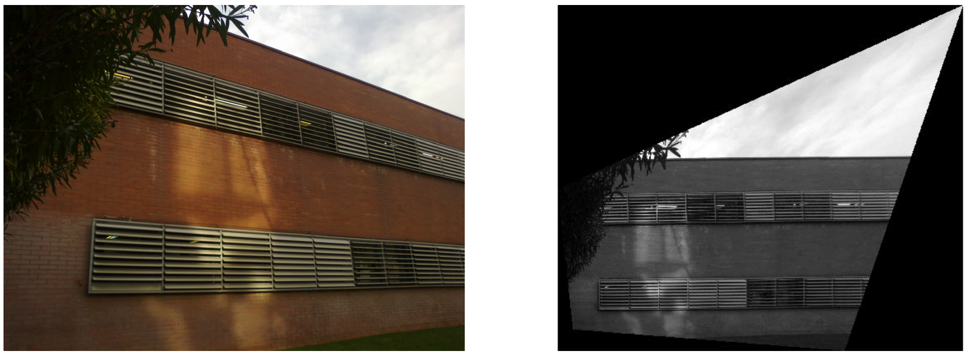
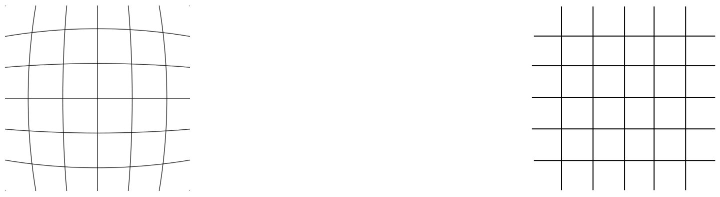


Figure 2.2.: The color image shows the original data recovered by a camera. Note that the lines of the windows, that are parallel in the real world, are not parallel in the image. The white and black image shows the real world, and is the result of applying the inverse projective transformation.

Distortion Not always the projection can be well modelled as a projectivity. Sometimes there is distortion. Lines that are straight in the real world are mapped as curves into the image. This phenomenon can be modelled by a polynomial transform from the image domain to the image domain. The model is completely described in [4]. Basically the image is magnified by a factor that increases or decreases with the distance to the camera center.



(a) Distortioned pixel grid (b) Undistortioned pixel grid

Figure 2.3.: Figure a) shows distortioned pixel grid and b) shows the undistortioned pixel grid. Lines have been mapped into curves by the non linear effect of the distortion.

The described polinomial transform depends on 5 coefficients: k_1, k_2, k_3, k_4 and k_5 and is:

$$\begin{pmatrix} u' \\ v' \end{pmatrix} = (1 + k_1 r^2 + k_2 r^4 + k_5 r^5) \begin{pmatrix} u \\ v \end{pmatrix} + \begin{pmatrix} 2k_3 uv + k_4(r^2 + 2u^2) \\ k_3(r^2 + 2v^2) + 2k_4 uv \end{pmatrix} \tag{2.1}$$

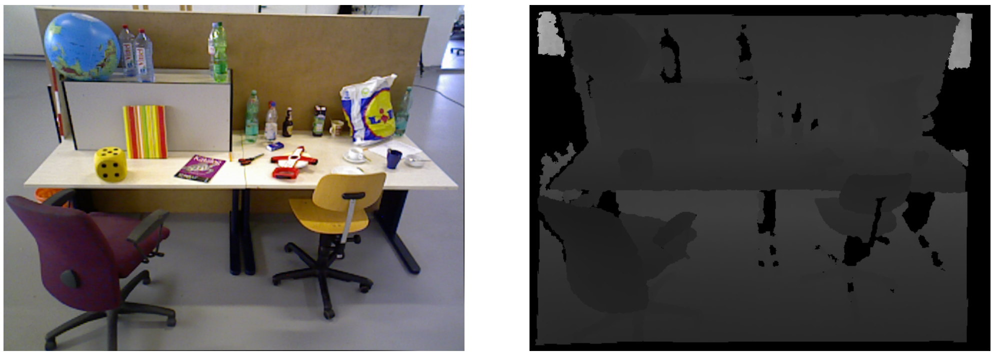
Where $r = \sqrt{u^2 + v^2}$. This function has to be applied after the projective transform has been done.

2.1.3. Depthmap

A depth map is a tuple $\langle R, P, D\rangle$ where R is an euclidean change of coordinates $\Omega_v \to \Omega_v$, P is a perspective transformation (or perspective with distortion) and D is a black and white image. p gives for each space point to which image point corresponds and d is a black and white image.

R depends of the extrinsic properties of the camera and is the only euclidean system whose origin is the position of the camera at the moment the depth map was taken, the z axis has the same direction as the principal axis of the camera and the y axis has the same direction as the up vector of the camera. P are the intrinsic properties of the camera, depend only on the properties camera which has been used to take the image and consists in a pseudo-projectivity and a distortion polynomial as explained before.

The image encodes the distance with the object. The darker the image is at a point, the closer the object is to the camera. An example depth map is figure 2.4. The measured distance is not the euclidean but $R(x) \cdot e_z$, the z coordinate of the point x in the system of reference defined by the camera. This allows to see planes parallel to the image with constant colour, making visual identification easier.



(a) RGB image (b) Depthmap

Figure 2.4.: Figure a) shows the rgb image while figure b) shows the depth map associated with the scene. The points where there is no data are coloured in black. Images taken from the data set [22].

The depth map can also define a function $s : \Omega_v \to \mathbb{R}$.This function gives the signed distance from each point to the sampled surface and is defined as $s : x \mapsto R(x) \cdot e_z - (D \circ P \circ R)(x)$. If $s(x) > 0$, x is said to be in the interior of the object with respect that camera, and if $s(x) < 0$ it is said to be in the exterior.

2.2. Variational Approach

The variational approach reduces the 3D reconstruction problem to a functional optimization problem so functional analysis knowledge can be used to solve the problem. The solution is the characteristic function of the volume and the objective functional is usually the sum of two terms, a data term and a regulariser term. The data term penalises the solutions that do not match with the sampled data and the regulariser is used to improve

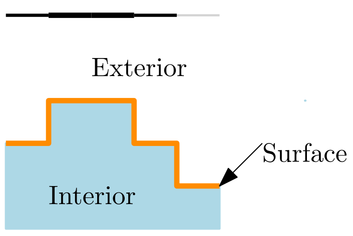


Figure 2.5.: The upper line represents the depth map of the 2-D cyan object.

the smoothness of the solution.

There are several advantages of using this method in spite of discrete methods, one of them is that it admits several modifications of the data term. The world objects can be better approximated by continuous manifolds rather than by discrete polygonal volume. It is also generally faster and requires less memory than other approaches. The other principal approach to solve computer vision problems is computing the graph cut of a graph. It has been studied in [2] or [15] This approach has the principal drawback of a huge memory consumption, and that it does not always converge to the real solution, as the resolution of the data goes to 0.

2.2.1. Variational Problem

The problem to solve is to find u such that:

$$\inf_{u:\Omega_v \to \{0,1\}} \text{Data}(u) + \text{Regularizer}(u) \tag{2.2}$$

Where $u : \Omega_v \to \{0, 1\}$ is the characteristic function of the object to reconstruct.

The feasible solutions are all the special functions of bounded variation. Usually the data term is of the form of $\sum^N |u - data_i|$ where $data_i : \Omega_v \to [0, 1]$ depends on the data given by image number i and the regularizer of the form $\lambda TV(u)$, being TV the total variation of u and λ is a constant.

The problem is not solved directly but usually a relaxation of it. There exists efficient methods for solving convex optimization problems, so a good strategy is to solve a convex relaxation of the problem, and then project the solution into the original solution space. Information about this can be found in several articles, such as [10, 7, 18]

2.2.2. Superresolution

Obviously the data obtained from the depth maps is not a perfect representation of the continuous-modelled real world. The image domain of a camera is not a continuous space but a finite n dimensional space, where n is the number of pixels . For this reason, a discretazion is done by the camera and part of the information is lost. There is also an error caused by imperfections of the camera, small movements while taking the image or strange light effects.

The superresolution method objective consists in artificially improving the resolution of the image. Basically consists in combining redundant information to solve a superdetermined system of equations, that should minimize the error. Two methods will be employed: upsampling and deblurring. In the variatonal method, a superresolution is represented as an operator that is applied to the solution in the data term.

Upsampling is simply augmenting the input images pixel resolution, choosing the closest neighbour strategy if there is no data available. When several images from slightly displaced positions are combined, the result shows a higher level of details than each of the original images. In figure 2.6 there is some data sampled with a coarser grid and in figure 2.7 there is the result of mixing all the coarse data in a finer grid.

Deblurring is trying to reverse the blurring process that the camera does. A camera does not project the data perfectly from the world to the image. A part of it is random but some of the error is caused because the camera mixes the data from the nearby pixels. This phenomenon is known as blurring and there are several methods to recover the original data.

The blurring operator in the image domain is usually a linear operator that does a weighted mean in the nearby pixels. This operator is for nearly all choices of weights invertible, but the direct approach of solving the linear system is not a good idea, as the matrix is ill-conditioned. A more complete explanation of deblurring can be found in [14].

2.3. Convex functionals

In this section a short informal introduction to some convex analysis results that will be needed in the rest of the thesis is done. The proofs are not stated. In [20] there are the complete proofs of all the results.

Definition 2.1. *A set X subset of a vector space is convex if $\forall x, y \in X, \forall t \in [0..1](tx+(1-t)y \in X$. A function (or functional) $f : X \to \mathbb{R}$ is convex if X is convex and $\forall x, y \in X, f(tx + (1 - t)y) \leq tf(x) + (1-t)f(y)$.*

One of the most important characteristics of the convex functions is that each local minimum is the global minimum, so the minimum can be found with simple algorithms such as gradient descend.

Definition 2.2. *The Legendre transform of a function f is noted with f^* and is defined as:*

$$f^*(u^*) = \sup_u \langle u^*, u\rangle - f(u)$$

It has the property that applying two times the Legendre transform yields a convexification of the original function. In [1] more information about the Legendre transform and its applications to computer vision is found.

2.3.1. Subdifferential

Although the convex functions are not always differenciable, they have always a subderivative. The subdifferential is defined as:

$$\partial f(x) = \{v \mid \forall x' \in X f(x') \leq \langle x' - x, v\rangle\} \tag{2.3}$$

in other words, the derivative of the function at a point may not be unique. It is easy to see that a convex function has a global minimum at a point x if and only if $0 \in \partial f(x)$. A

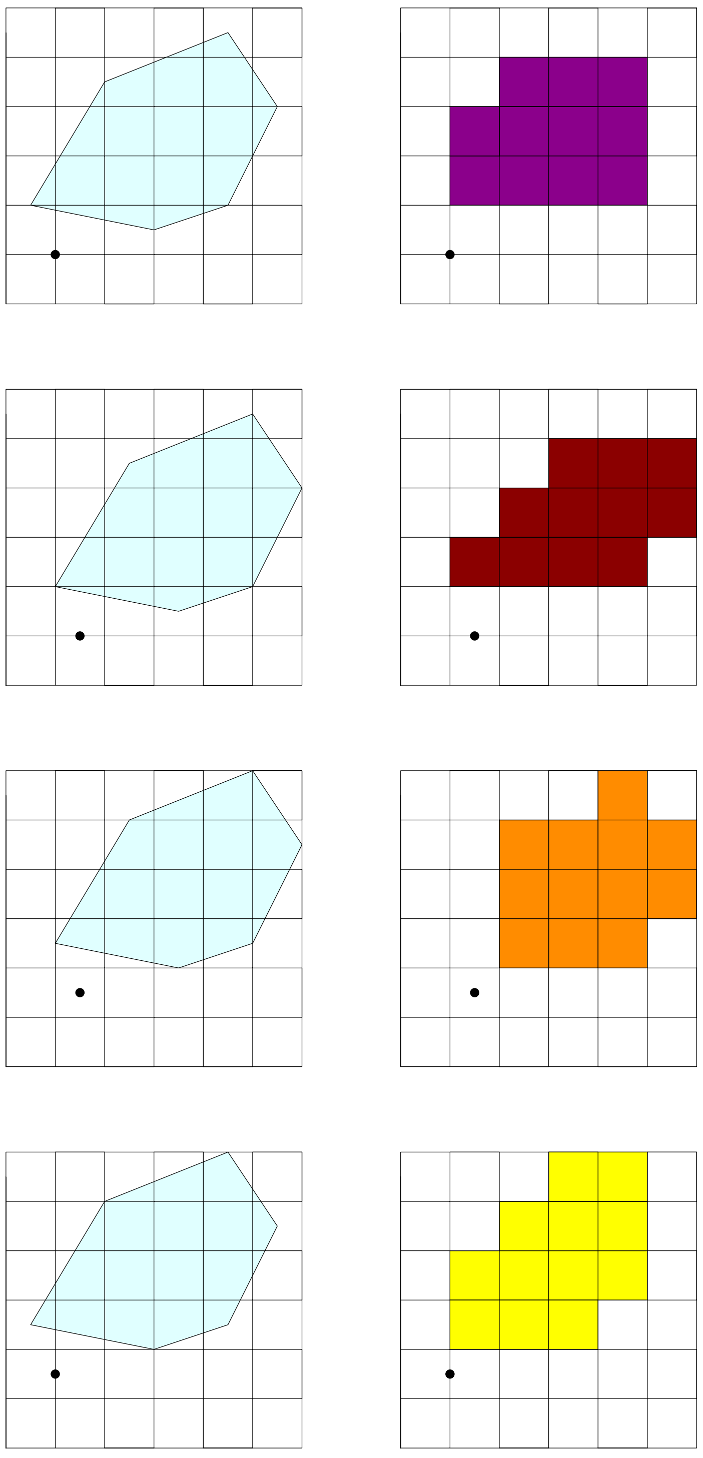


Figure 2.6.: Each of the figures show the original projection in the image plane of the object to reconstruct in cyan and on the right the object discretization in pixels made by the camera. At each pixel the camera will detect the object if the object projection covers at least half of the camera. The point has the same position in all images and is only purpose is to make visual identification of the parts easier.

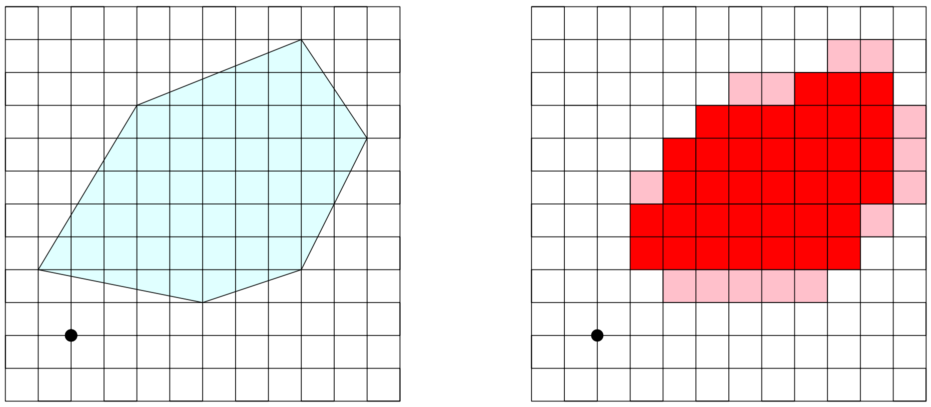


Figure 2.7.: This is the result of combining the data in 2.6 in a finer grid. The pixels where 3 or more images coincide is coloured in red, while the pixels where exactly 2 images coincide is coloured in pink. Note that this data has a higher level of detail than each other of the original data.

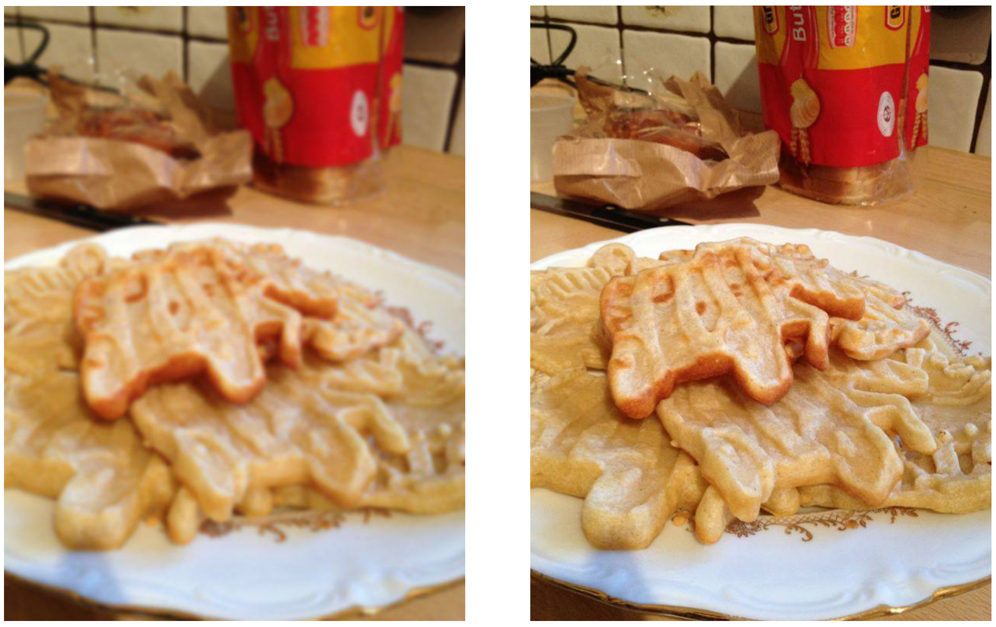


Figure 2.8.: Figure a) shows a blurred image and Figure b) shows the result of applying a gaussian deblurring.

geometrically meaning of the subderivative at a point x is the set of hiperplanes which are tangent to the graph of the function at that point.

A point x is a global minimum of a convex function f if and only if $0 \in \partial f(x)$

2.3.2. Concave functions

A function f is concave if $-f$ is convex. Anagolally to convex functions, concave functions are good for solving maximizing problems, as each local maximum is the absolute maximum.

The subderivative of a concave function is also defined and is the set:

$$\partial f(x) = \{v \mid \forall x' \in X f(x') \geq \langle x' - x, v\rangle\}$$

2.3.3. Saddle points

A saddle point of a function $f : X \subset \mathbb{R}^n \times Y \subset \mathbb{R}^m \to \mathbb{R}$ is a point (x, y) such that $\forall x' \in X f(x', y) \geq f(x, y)$ and $\forall y' \in Y f(x, y') \leq f(x, y)$.

This points are important for solving problems of the form:

$$\min_x \max_y f(x, y) \tag{2.4}$$

In general, $\min_x \max_y f(x, y) \neq \max_y \min_x f(x, y)$, but if there is a saddle point, then the equality is true. Therefore, it is important to know if a set contains a saddle point or not. Also the primal dual algorithm used to solve our problem converges not to a solution but to a saddle point (it is later introduced). Assuring the existence of a saddle point is for that reasons important.

If $f(x, y)$ is a convex-concave function, then under certain hypothesis the existance of a saddle point can be stated. This is theorem 37.6 of [20].

Theorem 2.3. *Let $k(x, y) : X \times Y \to \mathbb{R}$ be a convex-concave function, ie for all $\forall x \in X k(x, \cdot)$ is concave and $\forall y \in Y k(\cdot, y)$ is convex. If X and Y are both compact in the ordinary topology, then k has a saddle point cointained in $X \times Y$.*

2.3.4. Gamma convergence

Γ-convergence is a convergence for a succession of functionals $X \to \mathbb{R}$ that has the property that is conmutative with minimization problems, so if $F_n \to F$, and $x_n = \arg\min F_n$, $(x_n) \to x = \arg\min F$. Here only some results will be shown in a rather informal way. The complete proofs are available in [3]. In this thesis, $X = BV(\Omega_v, [0..1])$ with the topology induced by the norm BV, defined as $||\cdot||_{BV} = ||\cdot||_{L_1} + TV(\cdot)$.

First, the definition of γ-convergence. There are several more equivalent conditions, but only this is introduced here. The results also have the additional hypothesis that the functions are equi-coercive.

Definition 2.4. *Let $F_n : X \to \mathbb{R}$ a sequence of functionals, then $\Gamma - \lim F_h \to F$ if:*

i. For all convergence sequences $(x_n) \to x$, $\liminf F_n(x_n) \leq F(x)$

ii. For each x exists a convergence sequence $(x_n) \to x$ so that $\limsup F_n(x_n) \geq F(x)$

The gamma convergence is good for optimization problems, as the limit of minimizers converges to the minimizer of the limit functional.

Lemma 2.5. *If $\Gamma - \lim F_h = F$, then $\liminf_u F_h(u) = \inf_u F(u)$ and every limit of a subsequence is a minimizer of F.*

2.4. Special Functions of Bounded Variation

The total variation of a one variable function $f : D \subset \mathbb{R} \to \mathbb{R}$ is defined as:

$$TV(f) = \sup_{P \subset \mathcal{P}} \sum_{i=0}^{N} |f(x_{i+1} - f(x_i)| \tag{2.5}$$

where $\mathcal{P}$ is the set of all partitions of the domain of f. When the function f is differentiable, then the total variable is equal to:

$$TV(f) = \int_A |\frac{df}{dx}(x)| dx \tag{2.6}$$

And its dual form is:

$$TV^{**}(f) = TV(f) = \sup_{w \in \mathcal{C}^{\inf}, |w| \leq 1} \langle f, \nabla \cdot w \rangle \tag{2.7}$$

When the function is of several variables the total variation is defined as in 2.7. The special functions of bounded variation are the ones that can be expressed as a difference of two non-decreasing functions. A proof of the previous results can be found in [9]. Notice that when the image of the function is discrete, the total variation of the function is the sum of perimeters between the different level sets scaled by the difference.

Total variation is a good regularizer because it does not penalize the roughness of the change, but only the "size" the change itself. It is important not to penalize the roughness, as the characteristic function of the real object presents rough transitions.

3. Related Work

Our work uses the well known approach of multi-labeling with total variation regularizer to solve the 3D reconstruction using the variational approach. Chambolle et al. have explained in [7] and [6] how to get the variational form of a multi-label problem as well as the benefits of using it. Our work also uses the work of Handa et al. in [1] to obtain a convexification of our problem. This problem is finally solved with the primal-dual algorithm of Chambolle in [8].

Our work uses depth maps taken with a kinekt as input. They are part of the benchmark of Sturm et al. [22] that also provides the trajectory of the camera. The original purpose of the framework is to test SLAM (Simultanious location and mapping) algorithms. This algorithms compute simultaniously a 3D reconstruction of the scene and the position of the camera. Our algorithm just computes the reconstruction of the scene. This decision was made in order to evaluate only the quality of the reconstruction discarting the errors made by erroneous location of the camera. Our approach is similar to the reconstruction part of some SLAM algorithms like the one used by Graber et al. in [13]. Their approach is more complex as they use a multivalued signed distance while we use a simpler binary characteristic function.

Superresolution methods have mainly been studied and applied in other fields of computer vision such as texture reconstruction to get better results. A good example of this is [12], where Cremers et al. use superresolution to obtain a huge augmentation of texture resolution.The aim of our work is to reproduce the same obtained results they obtained in the field of geometry reco nstruction. In [24] Pock et al. explain how a generic superresolution problem can be dualized in order to get a form that can be computed with a primal-dual algorithm. Kosarev studies the maximum quality gain produced by superresolution in [16].

There are not many examples of superresolution usage in geometry reconstruction. Schuon and Thrun have already used superresolution in [21, 23]to generate depth maps of higher resolution joining several low resolution depth-maps. The deblurring operator has not been used in this methods, as the authors consider that no gain in the quality has been obtained. The differences from our work is that we study the effect of deblurring in the reconstruction and we also work in a volume 3D domain instead of in a 2 dimensional image. Our approach is supposed to be more accurate but also more computationally expensive.

Part II.

Problem statement

4. Problem definition

In this chapter the specific problem and the exact algorithm that is used to solve it are introduced. The problem data consists in a set of N depth-maps, for each of them both the position and orientation of the camera as well as its internal properties are known. In this chapter and latter on this thesis the integration domains are not always specified. If it is not specified the domain is the maximum where the function is defined. To improve the readability the functions parameters are not specified in the integrals.

4.1. Optimization Problem

In this section the actual modelization of the problem is presented. The functional to optimize consists in two terms: a regularizer and a data term. The feasible solutions u are characteristic function so their image is the set $\{0, 1\}$.

4.1.1. Data term

The data term is the sum of the data term for each one of the N depth map. It is defined as the following:

$$\sum_{i=1}^{N} |(Au - d_i)\chi_i| \tag{4.1}$$

A is the superresolution operator, d_i the data function and χ_i the visibility function. All of them are introduced in the following paragraphs.

Data function $d_i : \Omega_v \to \{0, 1\}$ is the interior-exterior function, defined as 1 if the z-distance from the point is more than the depth of the point and 0 else. Figure 4.1 shows an example of the data. It is formally defined as:

$$d_i(x) = \begin{cases} 1 & \text{if } s(x) \geq 0 \\ 0 & \text{if } s(x) < 0 \end{cases} \tag{4.2}$$

Visibility χ_i is the visibility function of the depth map number i which the fact that no data can be seen after the object. The intention to this function is to represent the fact that a depth map can not give any information of the parts of the object that are far behind the observed depth. For instance, the depth map from position x of the both 2D objects represented in figure 4.2 would be the same.

Figure 4.3 shows the visibility function that is used for. In this case, the function value is defined as following:

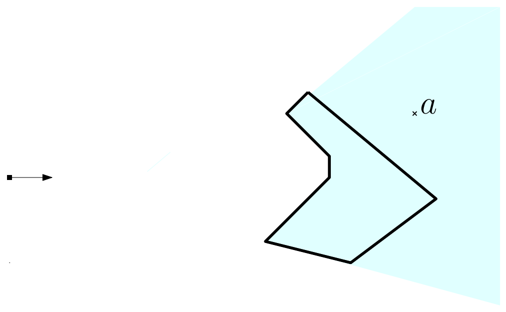


Figure 4.1.: This figure shows the values of the data function in a certain 2D object. The point represents the position of the camera and the vector its principal axis. The black line is the object. Note that the value of the function at point a is 1, even though the point does not belong to the volume. This is due because no information is known far away behinf the visible boundary.

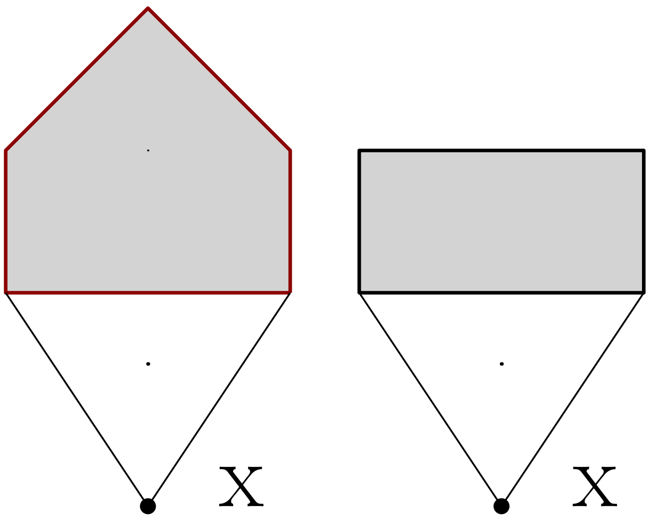


Figure 4.2.: If the camera is at the point x, the resulting depth map from both objects will be the same. This means that the information given by a depth map can not be extrapolated to all the object but should be only taken into account until the nearby of the visible boundary.

$$\chi_i(x) = \begin{cases} 1 & \text{if } s(x) \leq \delta_v \\ 0 & \text{if } s(x) > \delta_v \end{cases} \tag{4.3}$$

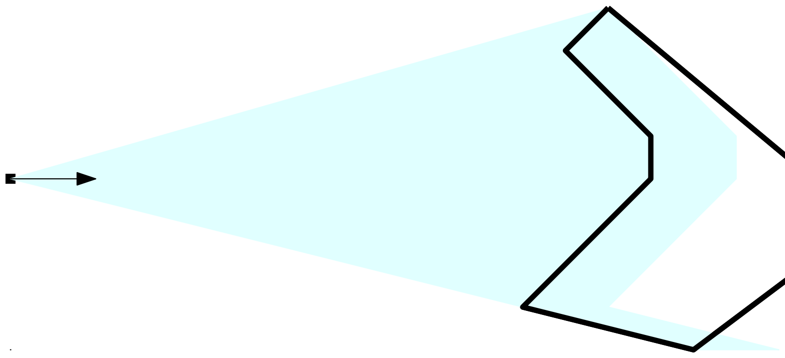


Figure 4.3.: If the camera is in the square with the given orientation, then the visible area is coloured in light cyan and the non-visible is left white.

This is probably the simplest visibility function, but others more complex can be used. For example, the visibility function with linear transition has also been implemented, but the observed changes in the result are minor in respect with the rough transition and has not used in the rest of the thesis:

$$\overline{\chi_i(x)} = \begin{cases} 1 & \text{if } s(x) < \delta_v/2 \\ \frac{\delta_v - s(x))}{\delta_v/2} & \text{if } \delta_v/2 \leq s(x) \leq \delta_v \\ 0 & \text{if } s(x) > \delta_v \end{cases} \tag{4.4}$$

Superresolution $A : BV(\Omega_v, \{0,1\}) \rightarrow BV(\Omega_v, [0..1])$ is the superresolution operator that consists in a bounded Gaussian convolution. It is different for each camera and for each voxel. It is defined as following:

$$A_i u(x) = \frac{\int_{B(0,\delta)} u(x+h) exp\left(\frac{h^2}{2\sigma_i(x)}\right) dh}{weight(\sigma_i)} \tag{4.5}$$

$\sigma_i(x)$ is the variance and $weight(\sigma_i) = \int_{B(0,\delta)} \exp\left(\frac{h^2}{2\sigma_i(x)}\right) dh$ the total weight so the weighted mean is well defined. The variance is $\sigma_i(x) = d(x, p_i)k$ the distance from the camera times a constant k. Points far from the camera have more uncertainty and more influenced from their neighbours than points near the camera.

This approach has a problematic situation that makes holes appear. When the super-resolution operator involves voxels without visibility they do not affect the total count of interior/exterior voxels. This can cause to uncorrectly mistake interior voxels as exterior voxels, as it can be seen in figure 4.4. To correct this flaw, the voxels without visibility are not used in the convolution and the value of the central voxel is used instead. This has given better results than increasing the range of the visibility function.

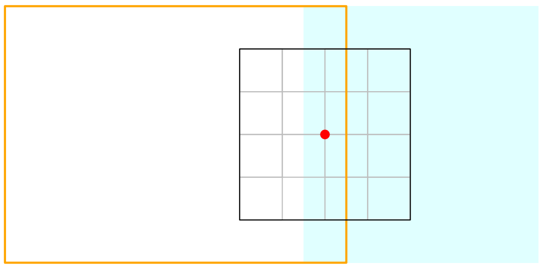


Figure 4.4.: This is a $2D$ representation of the situation where the error arises. The red point is the center of the convolution. The area of the convolution is delimited by the black lines. For simplicity, the convolution is the mean. The orange lines marks out the visible area, while the cyan area represents the object. If the original approach is used then the result will be that the point is outside, as in the convolution there are 4 square inside the voxel, 6 outside and 6 without visibility. The data term solution would say that incorrectly this point is exterior.

$$A_i u(x) = \frac{\int_{B(0,\delta)} \left(\chi_i(x+h)u(x+h) + (1-\chi_i)u(x))exp(-\frac{h^2}{2\sigma_i(x)})\right) dh}{weight(\sigma_i)} \tag{4.6}$$

Notice that no operator is needed for the upsampling as it is done implicetely while recovering the data function by projecting the points to the image.

4.1.2. Regularizer

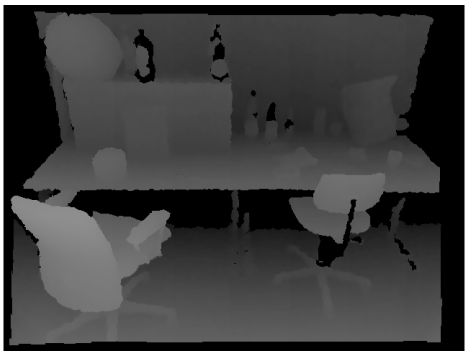
The regularizer is the total variation of the characteristic function weighted by a factor λ. The correct value is determined experimentally and depends on how many depth maps are used and their spacial distribution. With big regularization details are lost and objects are melted together. If the regularizer is too big then the optimum will be the void solution which says that there is no object at all. In figure 4.5 it can be seen the effect of too much regularization.

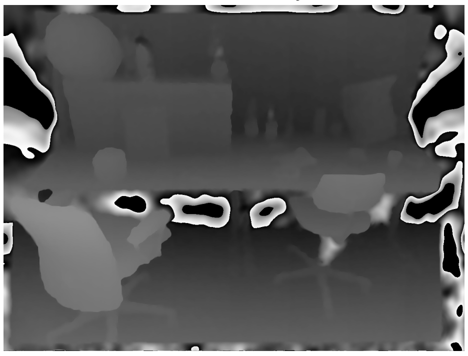
4.1.3. Feasible set

As explained in chapter 2, the feasible set appropriate to the problem is the functions of bounded variation. All the considered functions are 0 at the boundary of the volume domain.

4.2. Convex relaxation

An arbitrary optimization problem is nearly impossible to exactly solve. On the other hand, convex functions have the property that every local minimum is a global minimum. Our optimization problem needs to be relaxed to a convex optimization problem in order to be solvable. The first thing to do is to convexify the feasible set, which is changed to


(a) Data only


(b) High regularization

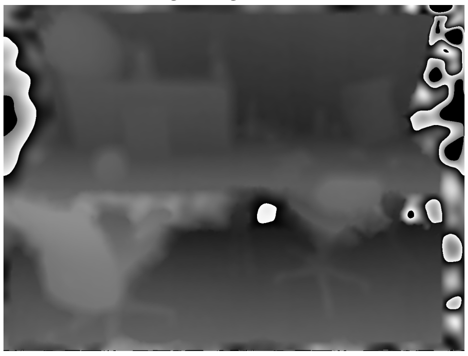

(c) Regularization near ∞

Figure 4.5.: This figures shows the depth maps of the reconstruction of a scene with different values of λ. It can be seen that the objects tend to be melted in a plane as the regularizer augments.

$\{u : \Omega_v \to [0,1]\}$. It is easy to see that both the data term and the regularizer are both well defined and convex on $\{u : \Omega_v \to [0,1]\}$.

$$TV(tu_1 + (1-t)u_2) = \int_{\Omega_v} |\nabla(tu_1 + (1-t)u_2)|dx = \int_{\Omega_v} |t\nabla u_1 + (1-t)\nabla u_2|dx$$

And using the triangular inequality, it is less than:

$$\int_{\Omega_v} |t\nabla u_1 + (1-t)\nabla u_2|dx \leq \int_{\Omega_v} t|\nabla u_1| + (1-t)|\nabla u_2|dx = tTV(u_1) + (1-t)TV(u_2)$$

The data is also convex:

$$\text{Data}(tu_1 + (1-t)u_2) = \sum_i^N \int |A_i(tu_1 + (1-t)u_2) - (t + (1-t))d_i|\chi_i dx$$

$$= \sum_i^N \int |A_i(tu_1) - td_i + (1-t)Au_2 - (1-t))d_i|\chi_i dx \leq t\text{Data}(u_1) + (1-t)\text{Data}(u_2)$$

After the optimal solution of the convex relaxation has been computed, a projection of the resulting function to the original feasible set $\{\Omega_v \to \{0,1\}\}$ must be done. The chosen projection is defined as:

$$\overline{u}(x) = \begin{cases} 1 & \text{if } u(x) \geq 0.5 \\ 0 & \text{if } u(x) < 0.5 \end{cases} \tag{4.7}$$

This projection is the one that minimizes the L_1 norm of the difference $u - \overline{u}$. Under the assumptions that the relaxed problem has been exactly solved, the difference of energy between the computed solution and the real solution can be bounded. Let $E(u) = Data(u) + \lambda TV(u)$, u^* the solution of the convex relaxation problem, $\overline{u}$ the projection of this solution and $\hat{u}$ the solution of the original problem,then the following inequality holds:

$$E(u^*) \leq E(\hat{u}) \leq E(\overline{u})$$

4.3. Discretization

The problem has to be discretized in order to be treated by a computer. To get a discrete problem the domains have to be discrete, as well as the functional have to be redefined to have the discrete functions as domain.

Domain discretization The volume domain is discretised in a finite number of cubic voxels equally distributed along it. The image domain has already been discretized in pixels by the camera. Ω_v^h denotes the set of all the centres of the voxels. To make things simpler, it is supposed that the domain can be decomposed in a natural number of voxels.

Note that the set of functions $\Omega_v^h \to \mathbb{R}$ is a finite dimensional vector space, in contrast with the set of functions $\Omega_v \to \mathbb{R}$. This makes that the cost functional has also to be redefined to have the discrete functions as domain.

Regularizer The regularizer must be discretized. This means that the continuous gradient operator has to be replaced by a discrete one. Forward differences has been chosen to approximate it, considering that the image of the function is 0 when the point lies outside the reconstruction domain. Once the gradient is defined, the divergence is automatically defined as its dual:

$$\begin{cases} \nabla u(i,j,k) &= (u(i+1,j,k)-u(i,j,k), u(i,j+1,k)-u(i,j,k), u(i,j,k+1)-u(i,j,k)t) \\ \nabla \cdot p(i,j,k) &= (p(i,j,k)_x - p(i-1,j,k)_x) + (p(i,j,k)_y - p(i,j-1,k)_y) \\ & \quad +(p(i,j,k)_z - p(i,j,k-1)_z) \end{cases} \tag{4.8}$$

Notice that as the value of the function u is 0 outside the volume domain Ω_v, this definition is also valid on the boundary voxels.

Data term The data term $\sum_i^N \int |Au - d_i|\chi_i dx$ is discretized to $\sum_i^N \sum_{x\in\Omega_v^h} |Au - d_i|\chi_i$.

Convergence Each discrete problem is of the form:

$$\min_{u\in\Omega_V^h\to\mathbb{R}} TV^h(u) + \text{Data}^h(u) \tag{4.9}$$

This Γ-converges to the continuous functional implying that the solutions energy also converges to the solution of the continuous functional. The discrete solutions are also near some solution of the continuous functional. In the article [7] a proof of the convergence of the total variation functional for 2 dimension can be found but it can be easily adapted to the 3 dimensional problem.

4.4. Primal Dual algorithm

The primal dual algorithm is a numerical algorithm that finds saddle points of the objective functional. If there exists a saddle point, it will be a solution of the problem (the solution may not be unique). Thought there always exists a solution for a optimization problem with continuous objective function in a compact domain, it is not true that there always exists a saddle point. With some extra hypothesis the existence of a saddle point can be guaranteed. A proof of this can be found in [11].

The concrete algorithm used is algorithm 1 described by Chambolle and Pock in [8] solves the following problem:

$$\inf_u \sup_d \langle u, Kd\rangle + F(u) + G^*(d) \tag{4.10}$$

where $K : Y \to X$ is lineal, $F : X \to \mathbb{R}^+$ convex and $G^* : Y \to \mathbb{R}$ concave. The variable u is called primal, and d is called dual. Primal steps where the primal variables are actualised and dual steps where the dual ones are. The algorithm is found in algorithm 1.

One remark useful for doing computations is that:

$$(\mathcal{I} + \gamma\partial G^*)^{-1}(q) = \arg\min_p (\frac{|p-q|}{2\gamma} + G^*(p))$$

Algorithm 1 Generic primal-dual algorithm

```
$u_0 \leftarrow$ initial primal
$p_0 \leftarrow$ initial dual
while Not convergence do
    $p_{n+1} \leftarrow (\mathcal{I} + \gamma \partial G^*)^{-1}(p_n + \gamma K \overline{u}_n)$
    $u_{n+1} \leftarrow (\mathcal{I} + \tau \partial F)^{-1}(p_n - \tau K^* \overline{u}_n)$
    $\overline{u}_{n+1} \leftarrow (\theta + 1)u_{n+1} - \theta u_n$
end while
```

A proof of this can be found in [8].

The primal and dual steps are chosen following the method explained by Pock and Chambolle in [19]. This choice guarantees convergence.

4.5. Problem to optimize

In order to apply the primal dual algorithm, the problem has to be transformed in an equivalent one which is easier to solve with the algorithm. The original problem is:

$$\inf_u \int |\nabla u| dx + \sum_{i=1}^{N} \int |\mathcal{A}_i u - d_i| \chi_i dx \tag{4.11}$$

4.5.1. Data term

If the image of each d_i is a discrete set $\{val_j\}$ the data term can be rewritten as:

$$\inf_u \int |\nabla u| dx = \sum_{i=1}^{N} \sum_j |\mathcal{A}_i - val_j| h_i^j dx \tag{4.12}$$

where h_i^j is a function whose value is the same as the visibility function χ_i where $d_i = val_j$ and 0 if $d_i \neq j$.

$$h_i^j(x) = \begin{cases} \chi_i(x) & \text{if } d_i(x) = j \\ 0 & \text{if } d_i(x) \neq j \end{cases} \tag{4.13}$$

In the case that the primal variable is constrained to the interval $[0..1]$, and the image of each d_i is $\{0,1\}$, the data term can be written as:

$$\int \sum_{i=1}^{N} h_i^0 |\mathcal{A}_i u| + h_i^1 |\mathcal{A}_i u - 1| = \int \sum_{i=1}^{N} (h_i^0 - h_i^1)(\mathcal{A}_i u) + h_i^1 \tag{4.14}$$

This function is linear on u. This is equal to:

$$\int \sum_{i=1}^{N} \langle (d_i^0 - d_i^1), \mathcal{A}_i u \rangle + \langle d_i^1, \overline{1} \rangle$$

Using the Hermitian conjugate of $\mathcal{A}_i u$, using the bilinearity of the inner product and grouping the constant terms with respect u, the data term is equal to:

$$\left\langle \sum_{i=1}^{N} \mathcal{A}_i^*(d_i^0 - d_i^1), u \right\rangle + cnst \tag{4.15}$$

As adding constant terms does not change the solution of an optimization problem, the final data term is:

$$\langle v, u \rangle, where\ v = \sum_{i=1}^{N} \mathcal{A}_i^*(d_i^0 - d_i^1) \tag{4.16}$$

Should the data function image different from $\{0, 1\}$, the absolute values must be transformed into its dual formulation as explained in [24]. The following scalar equality is used:

$$\forall y \in \mathbb{R}, |y| = \sup_{|w|<1} w \cdot y$$

As the variable u is a function, then the following conversion is done:

$$\sum_{i=1}^{N} \int |\mathcal{A}_i u - d_i| \chi_i dx = \sup_{w} \sum_{i=1}^{N} \int (\mathcal{A}_i u - d_i) w_i \chi_i dx = \sup_{w} \langle w_i, \chi_i((\mathcal{A}_i)u - d_i) \rangle \tag{4.17}$$

The problem of this approach is that a dual variable is needed for each of the cameras, resulting in a huge memory and computation cost. If the operator $\mathcal{A}_i = \mathcal{A}_j$ for each i, j, then an histogram approach can be done, needing only one dual variable for each element of the discretezed image of the data function.

4.5.2. Regularizer term

The primal form of the total variation $TV(u) = \int |\nabla u|$ can not be easily used in the algorithm, so it must be changed to an equivalent form that can be easily fitted in the algorithm. The total variation is used instead in its dual form:

$$TV(u) = \sup_{p \in \mathcal{P}} \langle u, \nabla \cdot p \rangle \tag{4.18}$$

4.5.3. Stopping criterion

As neither the theoretical decreasing rate nor a bound of the distance to the solution are known, the Cauchy convergence criterion must be used. The execution will stop when $||u_{n+1} - u_n||_1 < \epsilon$, for a fixed ϵ.

The principal drawback of using this approach is that computing $||u_{n+1} - u_n||_1 < \epsilon$ is costly and bad parallelizable and that even though $||u_{n+1} - u_n||_1 \to 0$ implies the convergence of the sequence, it is not possible to estimate the error to the exact solution from this. In order to avoid as many computations as possible the convergence is not checked after each iteratio but after several iterations.

4.5.4. Saddle point existence

Theorem 2.3 guarantees the existence of a saddle point, as the functional is concave- convex and the domain is the cartesian product of two compacts.

4.5.5. Final form

The final form of the problem is:

$$\inf_u \sup_p \langle u, \nabla \cdot p \rangle + \langle \sum_{i=1}^{N} \mathcal{A}_i^*(d_i^0 - d_i^1), u \rangle \tag{4.19}$$

So $K = \nabla \cdot$, $F(u) = \langle \hat{d}), u \rangle$ and $G(p)$ is the convex characteristic function of the set $\{p \mid p \in \mathcal{C}^1(\Omega_v, \mathcal{R}^3 \cap ||p||_\infty < 1\}$. Joining with algorithm 1 the final form is algorithm 2 where $\hat{d}) = sum_{i=1}^{N} \mathcal{A}_i^*(d_i^0 - d_i^1)$

Algorithm 2 Primal dual algorithm

```
$u_1 \leftarrow 0$
$u_0 \leftarrow 0$
$p \leftarrow 0$
**while** $|u_n - u_{n-1}| < \epsilon$ **do**
    $u_{n+1} \leftarrow \text{proj}_{u:\Omega_v \to [0..1]}(u_n + \tau \nabla \cdot p - \tau \hat{d})$
    $u_{n+1} \leftarrow \text{proj}_{||_{\inf} \leq 1}(p_n + \gamma \nabla (2u_{n+1} - u_n)$
**end while**
```

5. Implementation

This chapter explains the decisions done while programming the solver. The code has been programmed in CUDA to use take advantage of the highly parallelizability of the primal-dual algorithm. More information about CUDA can be found in [17]. The most important code optimizations that do not modify the problem are presented. The efficiency is also discussed.

5.1. Narrow Bands

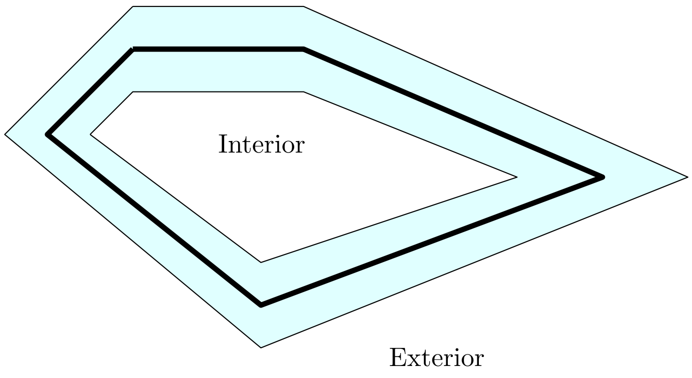


Figure 5.1.: Narrow band of an object. The light cyan represents the subset of the domain that belongs to the narrow band. The narrow band decomposes the rest of the domain in two connected sets, the interior of the object, and the exterior.

In order to reduce the memory and computing time, narrow bands have been implemented. Narrow bands are sustained on the fact that nearly all cameras agree for most points of the volume domain it is known a priori whether they belong to the volume or not. This can be used to compute the optimization only in the areas where there is uncertainty imposing the previous knowledge in the other voxels. This allows to reduce significantly the memory usage and computational cost. In figure 5.1 there is an example of the points beloning to a narrow band. The narrow band domain is noted as Ω_n.

The narrow band structure is stored as a list of nodes. The nodes are doubly linked with each of its neighbours in the axis of coordinates. There are two special nodes that represent the inside and outside of the narrow band. The neighbours of that nodes are themselves. In figure 5.2 the node structure can be seen. Each of the nodes has the following atributes:

- The indexs of the neighbours in the 6 allineated directions.
- The voxel position, 3 coordinates.

- The u and u_{old} values.
- The dual variable for the total variation (a vector of dimension 3).
- The coefficient of the data term (scalar).

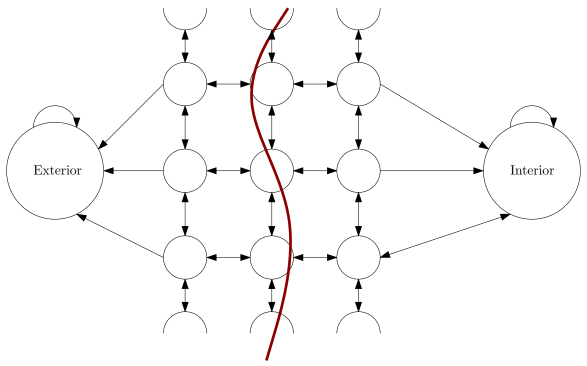


Figure 5.2.: Narrow band as a list of nodes in a 2D model. The brown line represents the observed surface. The big nodes the special nodes of interior and exterior. The narrow band continues above and down.

There are two special nodes that do not take part in the optimization: the interior and the exterior. They represent respectively the voxels that are deep inside the object and the ones that are far away from the boundary of the object. To make the structure closed the special nodes are linked with themselves . This voxels have the primal variables fixed respectively to the values 1 and 0 and both have the dual variable to $(0, 0, 0)$.

A problem may arise to the nodes that have back link to a special node. If their primal value is different to the primal value of the special node, then the total variation is not correctly computed. This is caused because the backward differences are taken for computing the divergence of the dual variable and the dual variable is not correctly defined for the special nodes. The situation can be seen in figure 5.3. The solution is to create a line of fake nodes, whose primal variable is fixed but the dual can change. Each time a node would be linked backwards to a special node it will be linked to a fake node instead. Figure 5.4 shows how the fake nodes are linked to the narrow band.

To compute the narrow band, first of all an histogram is done for every voxel, counting the number of cameras that see the voxel behind the boundary and the ones that see it before. Also it is determined whether the voxel belongs to the narrow band, if its distance to the boundary is less than δ_{narrow}. After that, the links to the other narrow band nodes are stablished. This process can be done with a coarser grid than the processing one to lower the memory and computing requirements. If done so, for each coarse voxel belonging to the narrow band several smaller voxels will be added.

The difficulty in paralleling this process is the part in which the narrow band is actually created. To allocate the memory for the narrow band the size must be known and also it is important the order in which the voxels are introduced. This makes difficult the implementation in CUDA.

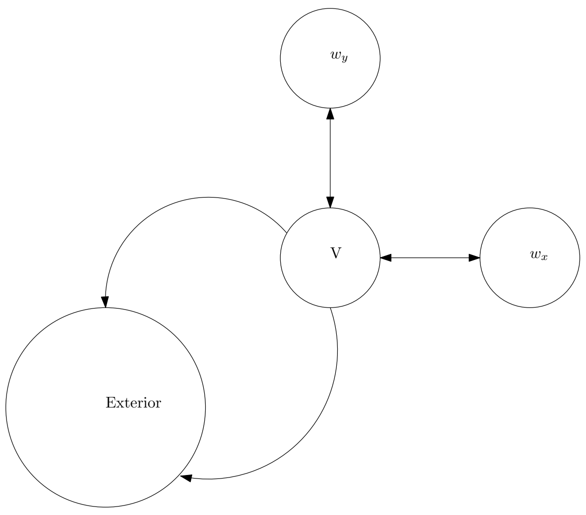


Figure 5.3.: This figure summarizes the situation in which the total variation is not correctly computed. The narrow band is a list of nodes in a 2D model. If all the nodes primal value is 1, then the total variation will be 0. The variable of the dual variable of the exterior should be $\sqrt{2}(1, 1)$, but is $(0, 0)$ instead. As the exterior is referenced by more than one node, no value can be given of p.

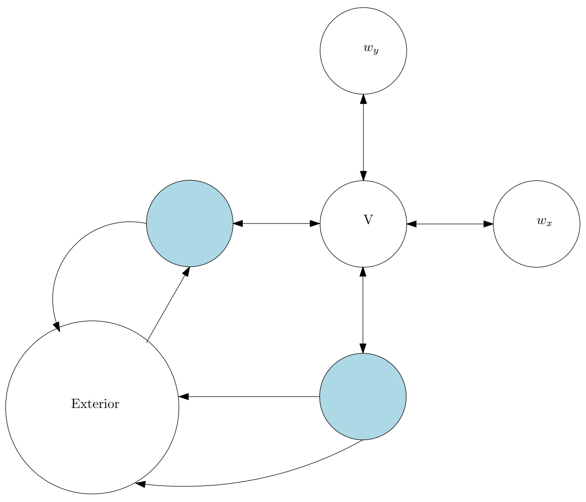


Figure 5.4.: Solution to the problem described in figure 5.3. The blue nodes are added to the narrow band and they store the correct value of the dual variable. Their primal variable is fixed to 0.

Algorithm 3 Algorithm for computing the narrow band

```
$NarrowBand \leftarrow \emptyset$
for all $x \in \Omega_v^h$ do                    ▷ This computing can be done in the GPU
    if $\exists n \mid \text{dist}(x, S_n) < \delta_n$ then
        $NarrowBand \leftarrow NarrowBand \cup \{x\}$
    end if
end for
for all $x \in NarrowBand$ do
    Search the neighbours of $x$ in the narrow band and link $x$ with them.
    If $x$ is linked to a special node by a back link then add the fake node $y$ to the narrow band instead and link it to $x$ with the correct coordinates.
end for
```

5.2. Taylor polynomial

To precompute the data it is necessary to compute the dual of the superresolution operator. This computation is costlier than the computation of the primal operator because for computing the value at each voxel, the weights of every neighbour in convolution range has to be known. If narrows bands are not implemented it can be precomputed, but the implementation of the narrow band makes the things more difficult when the neighbour whose weight is needed lay outside the narrow band. Computing the exact weight is a costly operation with cost $O(v^3)$, where v is the range of the convolution.

In order to improve the efficiency, the taylor polynomial of the weight considered as a function of the distance is computed and used. The chosen degree of the polynomial is 9 and the center is the mean distance of the data only solution to the first camera. It has produced no visible difference between the original exact solution.

Dynamic programming has been used to compute the derivatives of the function $exp\left(-\frac{x^2}{km}\right)$ with respect m. In the rest of the thesis the variable m is called d, but here is called m in order to avoid confusion with the differential. By induction can be easily proofed that for $n \geq 1$:

$$\frac{d^n}{dm^n} \exp\left(-\frac{x^2}{km}\right) = \frac{1}{m^n} \sum_{i=1}^{n} a_i^n (-x^2/m)^i \exp\left(-\frac{x^2}{km}\right) \tag{5.1}$$

where a_i^n are defined with the following recursive equation:

$$\begin{cases} a_1^1 & = & -1 \\ a_0^i & = & 0 \\ a_i^n & = & -a_{i-1}^{n-1} - (n+i-1)a_i^{n-1} \end{cases}$$

5.3. Computational and memory complexity

In this section the computational and memory complexity of the process are calculated. The biggest computational cost is at computing the data term and the memory cost at computing the narrow band. This would change if the data function image has some value different than 0 and 1, making the primal and dual steps really costly (making them as costly as precomputing the data term).

The following notation is assumed: $|\Omega_v^h|$ is the size of the voxel grid with voxel length h, $|\Omega_n|$ is the size of the narrow band, v is the range of the narrow band and N the number of images.

5.3.1. Initialization

Previously to compute the narrow band, an histogram has to be done counting for every voxel how many cameras see it before or outside the depth surface. Also it has to be computed if the point lays near the surface for some camera. This has a cost of $O(N\Omega_v)$ and can be executed on the graphic card. The next step is to create the narrow band structure, deciding which voxels pertains to the narrow band and linking each node with its neighbours. This can not be easily parallelized and has a complexity of $O(\Omega_v)$.

The data function coefficients $\sum_{i=1}^{N} \mathcal{A}_i^*(h_i^0 - h_i^1)$ has to be computed for each voxel belonging to the narrow band. For each node, the v^3 neighbouring pixels have to be visited and a constant amount of work has to be done. The work is constant due to the weight being approximated by the Taylor polynomial. Should it not be the situation, then an extra work of $O(v^3)$ for each neighbour should be done. Precomputing is not possible, as there are voxels that lay outside the narrow band. The total amount of work for precomputing the data term is $O(N|\Omega_n|v^3)$. The computing of the taylor polynomial can be done in $O(v^3T^2)$ where T is the degree of the Taylor polynomial. This term can be thrown away of the cost analysis because it does not depend neither on the number of images nor the size of the voxel grid.

To sum up, the total computational cost for the initialization is $O(N(|\Omega_v|+|\Omega_n|v^3))$ and requires a minimum memory of $O(|\Omega_n|)$ to store the narrow band. No auxiliary memory in the GPU is needed.

5.3.2. Steps

As the data term has already been precomputed, each iteration is quite simple. For each voxel only the divergence and the gradinet have to be computed, and some of scalar sums and multiplications, so the temporal cost is constant. This gives a cost of $O(|\Omega_{Narrow}|)$, that can bee done. No extra memory has to be used but the one already used to store the narrow band. If the data term can not be precomputed then for each iteration the dual of the superresolution operator has to be computed. This makes the iteration quite costly, needing $O(N|\Omega_{Narrow}|v^3)$.

Computing the difference between steps , $||u_n - u_{n-1}||_1$, in order to know if convergence has been achieved has a cost of $O(|\Omega_{Narrow}| + \log_2|\Omega_{Narrow}|)$, that can be computed in the GPU the first term can be done in parallel but the second not. This is not computing at all iterations but only on a few of them. This allows to discard the term from the asymptotic analysis.

Part	**GPU**	**CPU**
Initialize	M	$3\lvert\Omega_v\rvert$
Computation	0	0
Binarization	0	$60\lvert\Omega_n\rvert$
NarrowBand	$60\lvert\Omega_n\rvert$	0
Maximum	$60\lvert\Omega_n\rvert$	$3\lvert\Omega_v\rvert$

Table 5.1.: Memory cost. M refers the number of voxels whose histogramms are computed at the same time and can be as low as desired.

Part	**GPU**	**CPU**
Initialize	$O(N(\lvert\Omega_v\rvert + \lvert\Omega_n\rvert v^3))$	$O(\lvert\Omega_v\rvert)$
Computation	$O(\lvert\Omega_n\rvert)$	0
Binarization	$O(\lvert\Omega_n\rvert)$	$O(\lvert\Omega_n\rvert)$
NarrowBand	$60\lvert\Omega_n\rvert$	0
Maximum	$O(N(\lvert\Omega_v\rvert + \lvert\Omega_n\rvert v^3))$	$3\lvert\Omega_v\rvert$

Table 5.2.: Computational cost of the algorithm, broken down by parts. $|\Omega_n|$ refers to the size of the narrow band, $|\Omega_v^h|$ to the size of the voxel grid and v is the range of the convolution.

5.3.3. Post processing

During the postprocessing the thresholding of the solution in the space $BV(\Omega_v, \{0, 1\}$ must be don as well as computing the boundary of the solution. All of this can be trivially done in linear time . This takes a total of $O(|\Omega_{Narrow}|)$.

5.3.4. Total Cost

The total memory consumption can be seen in 5.1. The narrow bind size is not included in the parts count as it is needed during all the execution. The total computational complexity can be seen in table 5.2.

Part III.

Results and Discussion

6. Results

In this chapter the results of several executions are presented. Contrary to the previous chapters it is based in experimental data rather than in theoretical reasoning. Though in this section only a few images are show, a lot of executions have been run and only the more exemplifying of them are shown in this thesis. The lack of ground-truth for the reconstruction forces the evaluation of the resulting images to be mainly qualitative. For a better visualization of the images please read the electronic version. The evaluation consists in computing the exact depth-map of the reconstructed volume from a certain position. This allows to compare different reconstructions. Another important section of this part is the memory consumption and the execution time. Although the theoretical complexity is known it is important to know the exact computing time in a real computer. A description of the test environment, the used data sets and the used hardware is found in the appendix B.

6.1. Effect of upsampling

In this part the executions are done without deblurring in order to analyse the effect of upsampling. The chosen object for this test is the plane. The reasons to choose that dataset are that it is a small object so its computation is not as costly as a whole scene but has enough level of details in order to appreciate the effects of superresolution. When it is not stated otherwise regularization is used. In B.2.1 a complete description as well as color images of the data set are found.

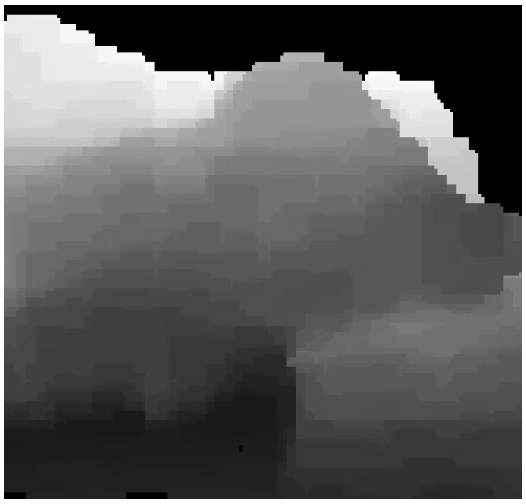
Results confirm that redundancy in the data allows to produce results with better resolution that the input depth maps. Figure 6.1 shows that there is a clear relationship between the density of the data and the quality of the results. The more density of images the better the result is.

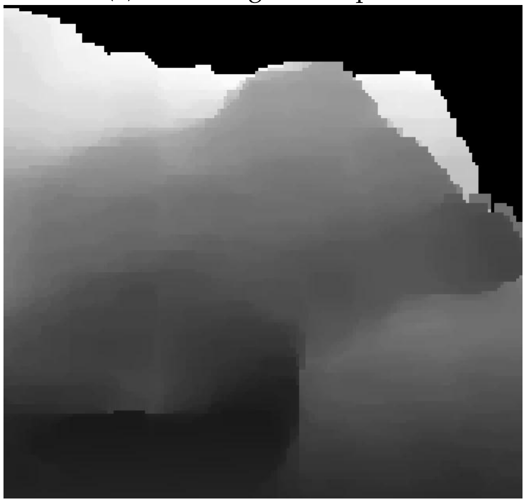
3D reconstruction is highly sensitive to noise in the data. Details are quickly lost due to the noise. Joining distanced images can produce bad results. In figure 6.2 can be seen that joining several images produces worse results than joining a few of images even though they are equally distanced. This is due that the relative error between depth maps increases with their distance and more noise is introduced when many images are joined.

6.2. Effect of deblurring

In this part the effect of deblurring is analysed. Previous authors have discarted the effect of deblurring, but we have found that there is a positive effect of deblurring with kinekt data. If the data is dense, deblurring smooths the boundaries like a regularizer. If the data is sparse, then deblurring manages to recover some details. This can be seen in figure 6.3. With the correct choice of the range and variance of the gaussian convolution, deblurring

(a) RGB image of the plane


(b) Example of a input depthmap image


(c) Solution obtained joining 5 images distanced 4 cm

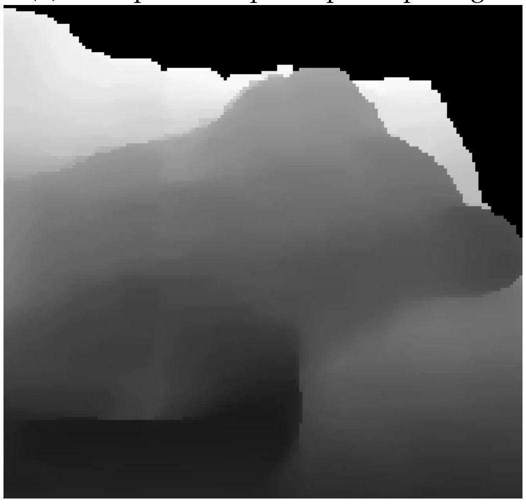
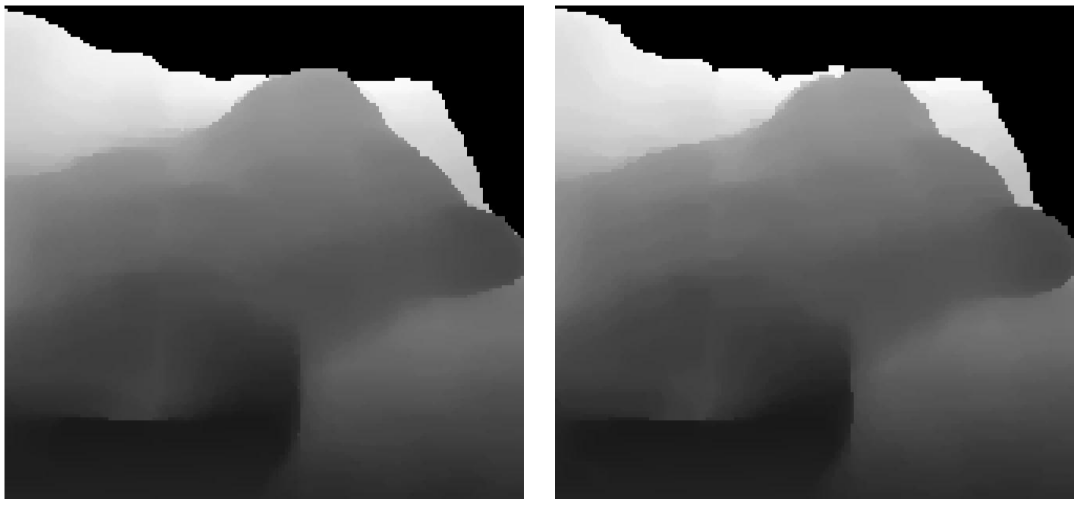

(d) Solution obtained joining 20 images distanced 1 cm

Figure 6.1.: This figure shows the effect of upsampling. On the upper row an input depth map and a color image of the plane can be seen in order to get an idea of how the result should be. The black pixels mean that no data is available (it lays outside the reconstruction volume).The lower row shows the result with 5 and 20 images using regularization with $\lambda = 2$. Results show that the result is better when the density of input depth maps is higher. The round detail in the left wing is recovered.


(a) Solution obtained joining 40 images distanced 1 cm

(b) Solution obtained joining 20 images distanced 1 cm

Figure 6.2.: This figure shows the results obtained joining 20 images and 40 images. The distance between the cameras is the same in both cases. The results show that there is a loss of quality in the images where more depth maps were used. The right wing curve is lost when using more depthmaps.

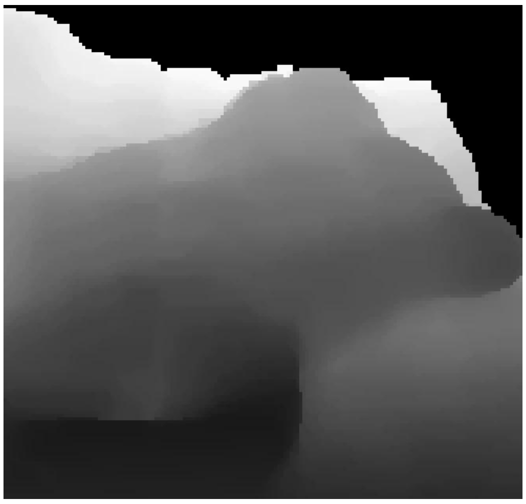
can be used to get smooth boundaries without losing details. In our experiments a range of 4 and $k = 10$ have proven to be the best choice.

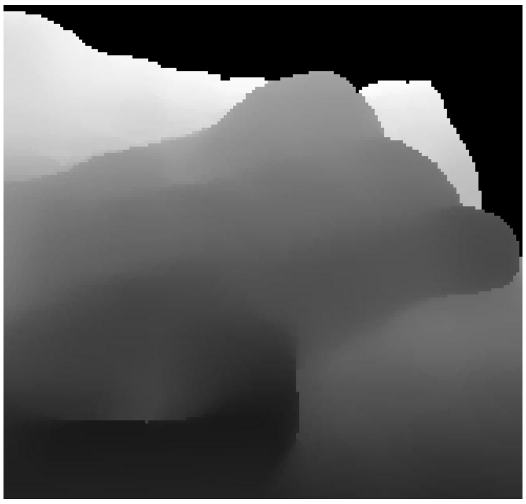
6.3. Half-Synthetic data

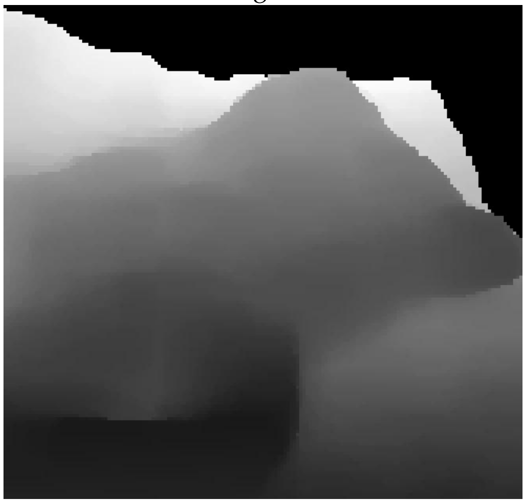
The effect of superresolution is difficult to appreciate due to the relatively high quality of the data coming from the kinekt. Details are difficult to distinguish with the naked eye. In order to study the effects of superresolution, the data has been manipulated to introduce some error. The results using the original data are used as ground truth.

Downsampled images have been used to see how much level of detail can be recovered. Their lower quality makes the difference between images easier to distinguish. In this case there is little difference between using the regularization and both regularization and deblurring. This results reliability is lower because the data has been artificially downsampled, in contrast with the real data used in the previous tests. The resulting image can be seen in 6.4.

The effect of errors in the trajectory is also analysed. To do that artificial error has been introduced to the position of the cameras. They have randomly moved to another point at most 1cm away following an uniform distribution. Results show that using deblurring makes the quality loss smaller. This can be seen in figure 6.5.


(a) Solution obtained joining 20 images without deblurring


(b) Solution obtained joining 20 images with deblurring


(c) Solution obtained joining 40 images without deblurring

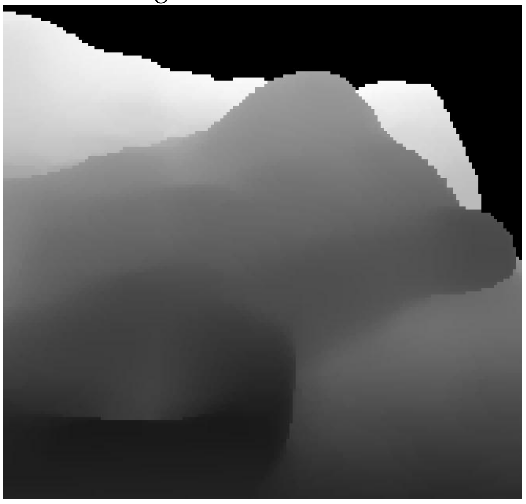
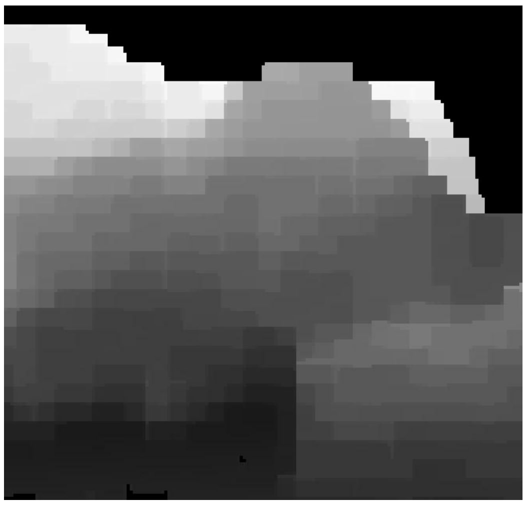
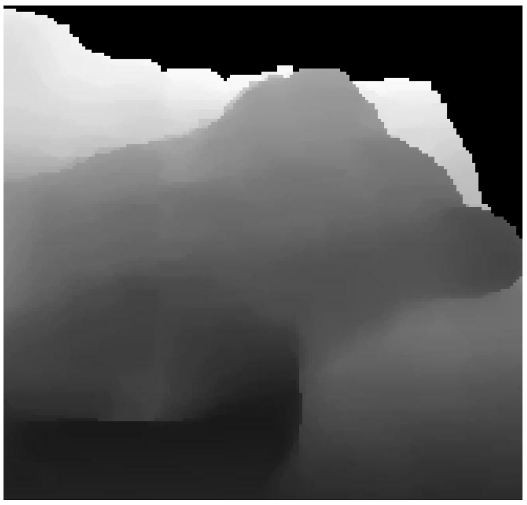
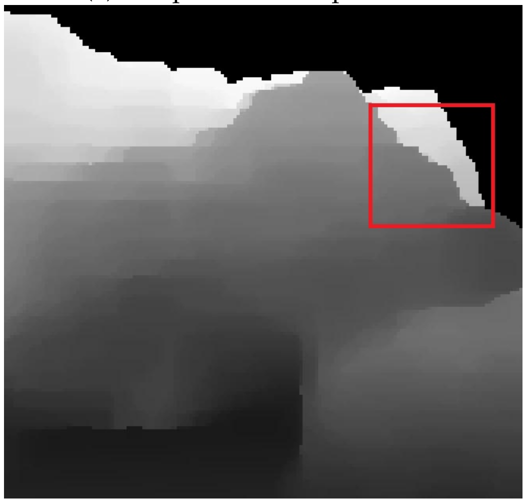

(d) Solution obtained joining 40 images with deblurring

Figure 6.3.: This figure shows the difference between using and not using deblurring. It can be seen that the boundary is more curved. Using a regularizer has the problem that details are lost, in contrary to using deblurring that preserves the real geometry of the object.


(a) Sample downsampled data


(b) Result obtained using the original data


(c) Result obtained without using deblurring and downsampled data

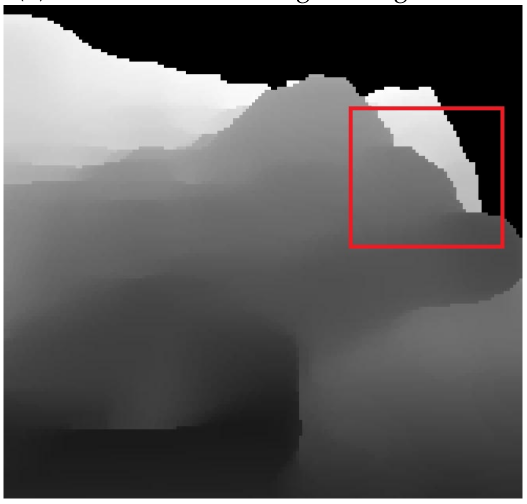
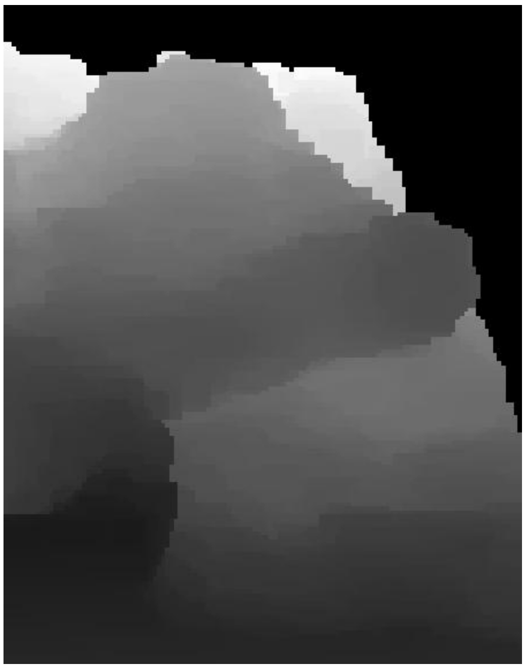
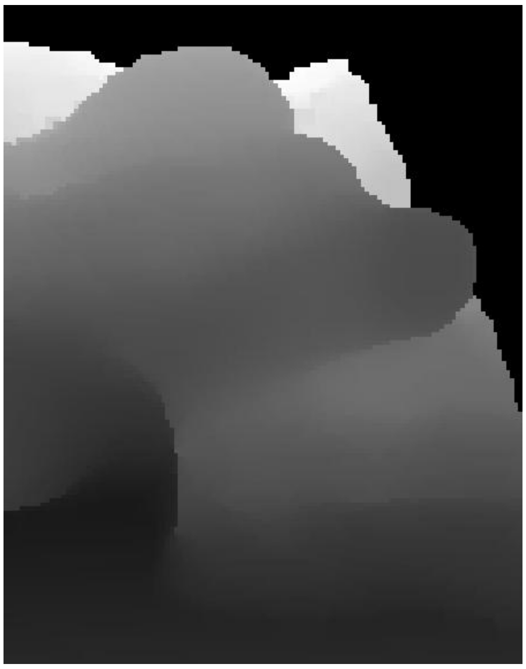

(d) Result obtained using deblurring and downsampled data

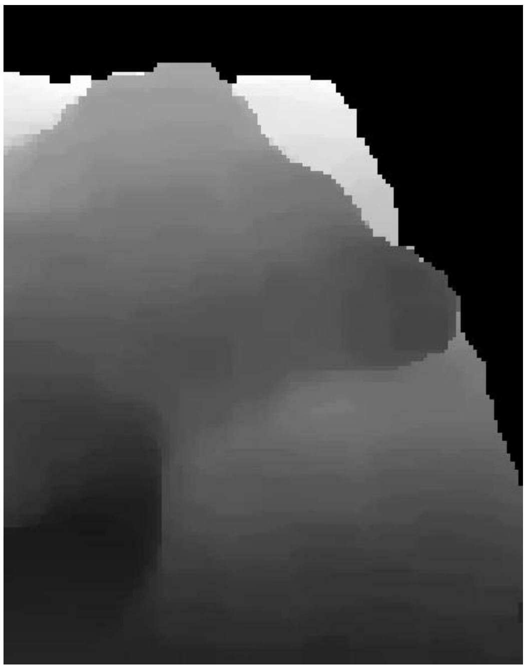
Figure 6.4.: This figure shows the results obtained using the downsampled data. The result obtained using deblurring better quality rather than the regularizer only. The round detail has nearly lost in the regularized result. The right wing curve (red) is better reconstructed when deblurring is used.



(a) Result obtained without deblurring and the noised data



(b) Result obtained with deblurring and the noised data



(c) Result obtained without deblurring and the original data

Figure 6.5.: This figure shows the results with the data with noised camera positions. It can be seen that the detail of the curved wing is lost.

6.4. Other data sets

The previous tests have been also repeated with other data sets in order to check whether the results also hold. The data sets are the dice and the teddy bear. More information about them can be found in B.2. The executions confirm that using deblurring helps to produce smoother boundaries of the objects and a higher level of details. In figure 6.6 this effect can be appreciated.

Regularization can smooth the boundary but has the drawback that geometric details are lost. On the other hand deblurring increases the quality of the details. This can be seen in figure 6.7. There it can be seen that deblurring manages to get a more round reconstruction of the eye detail.

6.5. Comparison with other algorithm

There are other algorithms that generate depth map with higher resolution from low resolution depth maps. The algorithm described by Thrun et al in [23] has been compared with our approach. This algorithm has an objective depth map whose resolution will be improved. Each depth map of the data is back projected into the volume domain and then projected to the objective depth map. The version of their algorithm that was used for the comparison uses deblurring, even though it was discarded as they said that no effect was found. Experiments with the data shows that there is indeed a positive effect with a long range deblurring. The drawback is that the execution time increases highly to 170 seconds were needed to mix 30 images and a deblurring range of 3.

Results show that the quality of our algorithm outperforms the image-domain algorithm, as can be seen in image 6.8. Strangely it is also faster, as is stated in the next section. It must be said that their algorithm generates the whole image while our approach reconstructs only an object in the scene. Our algorithm however generates the whole 3D model.

6.6. Memory consumption and running time

Though the complexity of the algorithm has already been theoretically discussed, it is important to see its running time in an actual computer. It is also important to know the maximum amount of GPU memory that is allocated. In this section the dataset of the plane will be discussed, sometimes with variations in the size of its voxel grid.

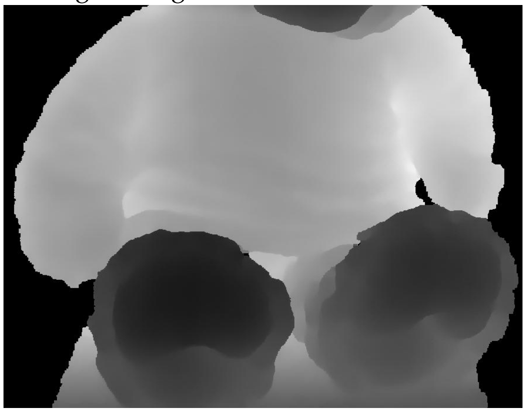
The computing has been splitted in 3 parts: computing the narrow band, computing the data term and the iterations. The binarization has not been studied due to its simplicity.

Computing the narrow band The narrow band is relatively easy to compute. In the figure 6.9 it can be seen the size and computing time of the narrow band for different grid sizes.

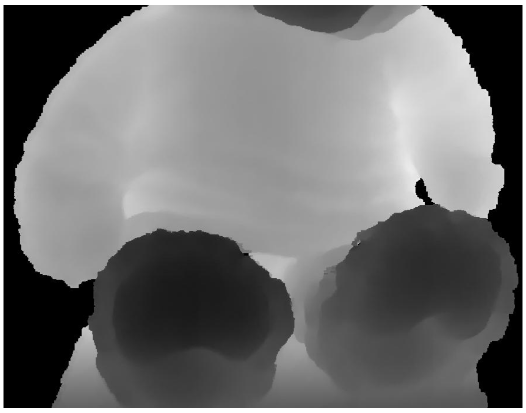
Computing the data term The precomputing of the data term is the part that consumes most of the execution time. The dual of the gaussian convolution has to be computed. The

(a) Reconstruction of the dice using deblurring and regularization.

(b) Reconstruction of the dice using regularization.

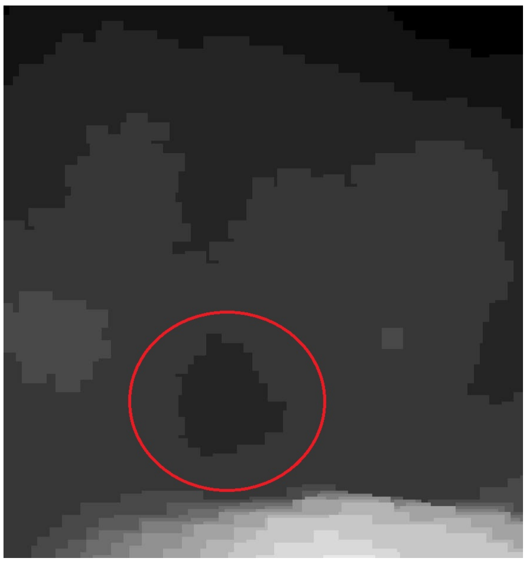


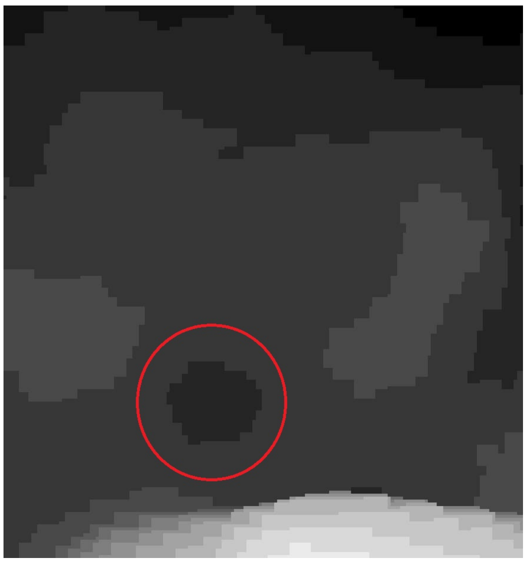
(c) Reconstruction of the teddy bear using deblurring and regularization.



(d) Reconstruction of the teddy bear using regularization.

Figure 6.6.: This figure shows the results with different objects. As with the plane, the deblurred result shows smoother boundaries. The feet of the teddy bear show a slightly smoother boundary in the deblurred result. Though not many differences can be seen in the teddy bear, a focus in reveal that there are indeed differences between using and not using deblurring (see figure 6.7).


(a) Reconstruction of the eye of the teddy bear using regularization and without deblurring


(b) Reconstruction of the eye of the teddy bear using deblurring and low regularization

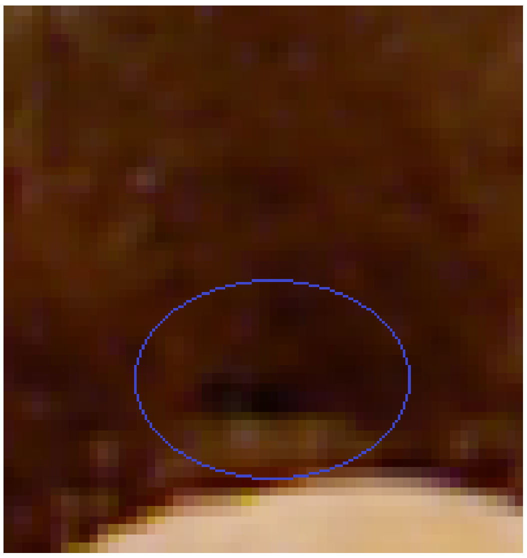
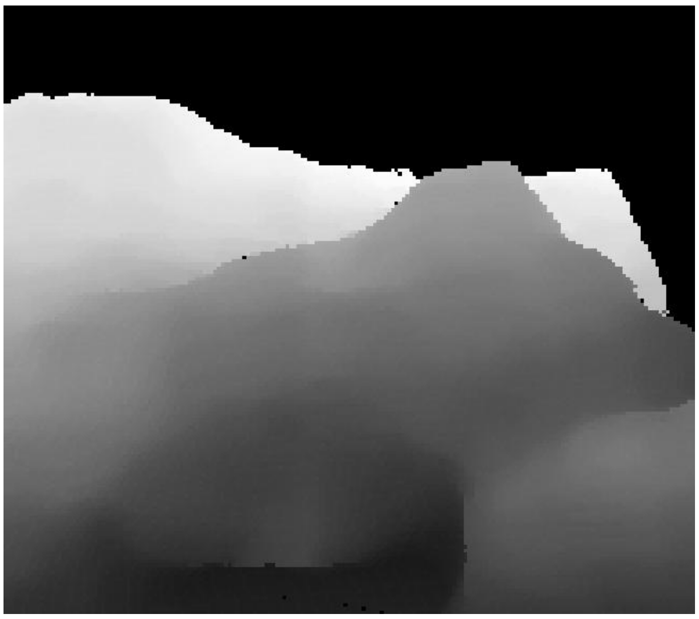

(c) RGB image containing the eye of the teddy (blue)

Figure 6.7.: This figure shows the zoom up the upper part of the face of the teddy bear. The colors have been inverted and rescaled to ease the visualization of the detail. The white part at the lower part of the image is the snout. The black part in the middle of the red circle is the right eye of the teddy bear. The lower row shows a RGB image of the same region. It is much rounder in the result that uses deblurring. The irregularities in the depth map are due to the irregular texture skin of the teddy bear.


(a) Solution using our algorithm

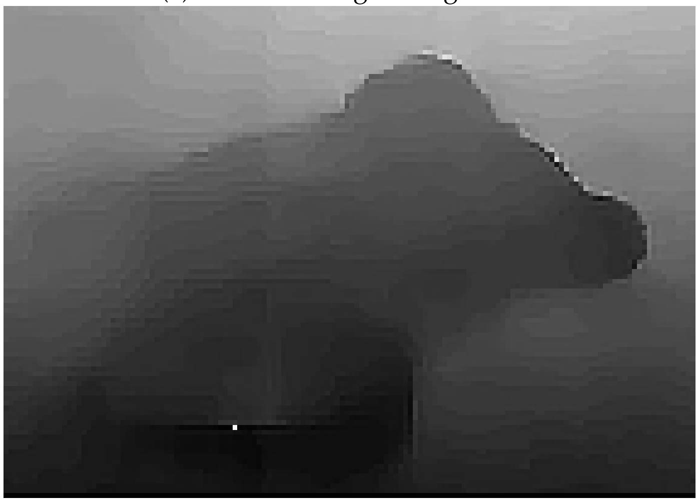

(b) Solution using the algorithm from Thrun et al. The chosen upsampling factor is 2

Figure 6.8.: This figure shows the result given by our algorithm opposed to the algorithm proposed in [23]. Note that the difference between the left wing and the ground is clearer in our algorithm. The other algorithm seems to have melted the left wing of the plane with the surface of the table. The surface boundary is also smoother.

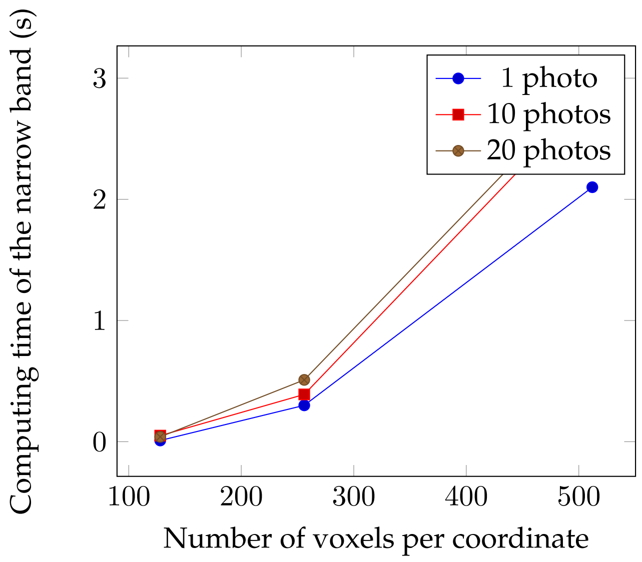


Figure 6.9.: This graphic shows the time needed to compute the band for different voxel grids. It follows that most important factor is the voxel grid and not the number of images.

process is divided in two parts: first the Taylor polynomial is computed and after that the dual of the operator is applied to the vector containing the data function.

The first part can not be computed in parallel and has to be done in the CPU. The computing time has always been less than 10 ms. The weight precomputing time is independent of the size of the narrow band and the number of photos, so can be considered as a constant.

The actual computation of the data term is done in the GPU. The main properties that affect it are the size of the narrow band, the number of images and the convolution range.

Iterations The iterations are easy to analize, as they are the same for every execution. Always less than 10000 iterations have been needed to achieve convergence. More iterations are needed when the regularizer constant is bigger.

Memory consumption The maximum memory usage is for storing the narrow band. It has a total of 15 floats per element of the band or 60 bytes. The size of the narrow band varies on the number of images used and the distance between them. In figure 6.12 the effect can be seen the effect of the grid on the size of the narrow band. The data set is also important as well as the dimensions of the voxel grid, as can be seen in figure 6.13.

At the precomputing the original volume can be cropped into several pieces so they can be computed one at a time, adapting the memory consumption to the available memory.

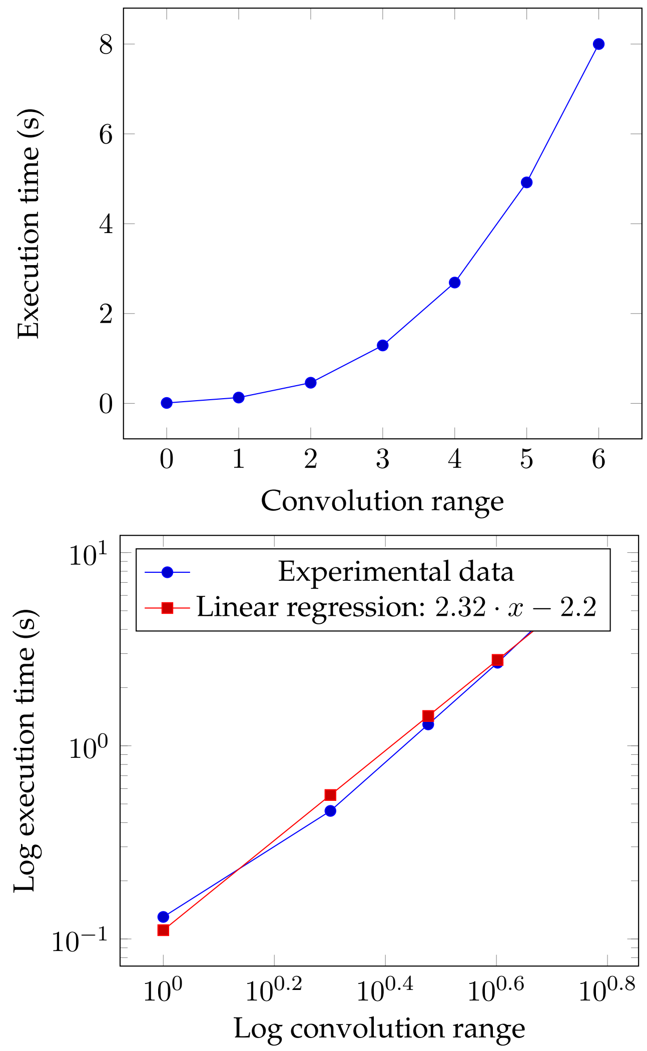


Figure 6.10.: This figure shows the execution time of the data term against the range of the convolution. Notice that the log-log graphic shows that the complexity with respect to the range of the convolution is less than cubic. This can be due to a better parallellization when the convolution range or a better success rate in the cache.

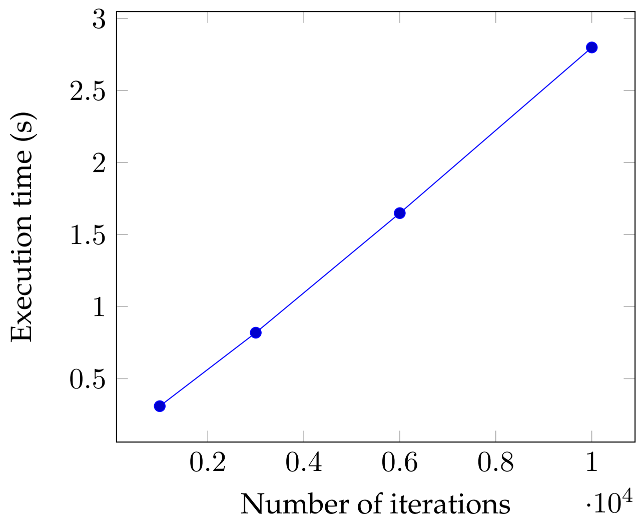


Figure 6.11.: This graphics shows the relationship between the number of iterations and the time needed to execute it. The relationship is clearly lineal.

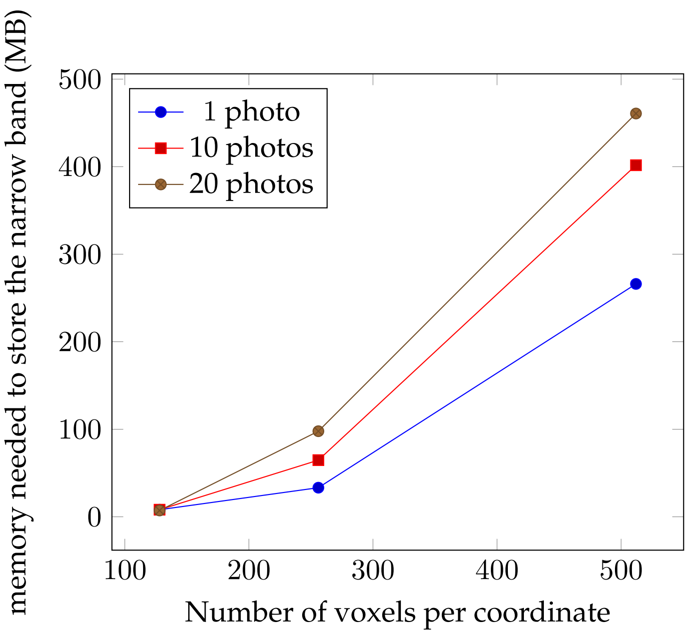


Figure 6.12.: Size of the narrow band with different number of grid sizes and photos. The chosen data set is the plane. The growth is approximately linear with resp to the size and to then number of photos. The latter may be explained due to each new image adds a part of the object that was not visible. If enough images are added, the size of the narrow band should stabilize.

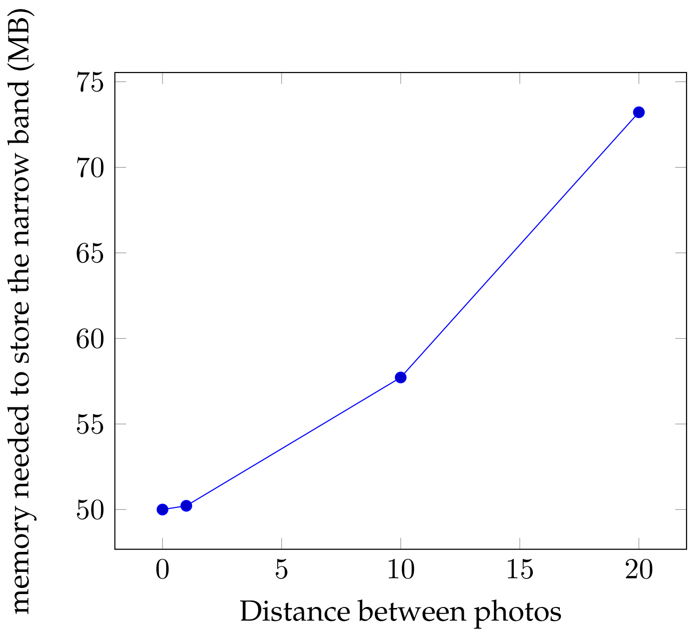


Figure 6.13.: This graphics shows the size of the narrow band against the distance between the photos. In all the cases there are 10 images in a grid of 256^3. It shows that the more space between photos, the more memory is needed.

7. Discussion and future work

7.1. Discussion

Results show that there are benefits produced by using the described superresolution operator. The boundaries of the objects are smoother and their transition is sharper. Not only upsampling but also deblurring have a positive effect in the quality of the result with the used data. As the positive effects of upsampling are already known and accepted, most of the discussion is about the effects of deblurring.

The main improvement in quality observed in real data is making the boundaries smoother. The result is similar to the one obtained by using a strong regularizer even when little or no regularizer is used. There may be small differences, but the lack of geometry ground truth makes difficult to decide which version is the most accurate. Superresolution needs redundant information to work. If there is not enough data available regularizing gives better results. On the other hand, superresolution has managed to palliate the loss of quality resulting of mixing distanced depth maps.

The approach without superresolution is very sensible to small errors in the data and details are lost with little perturbations. When noise have been added to the camera positions or the cameras are distanced deblurring has proven to be useful to recover details that have been lost due to the noise. Deblurring effect is also seen clearer when downsampled is used. With only a few images the original resolution is restored, outperforming by far the algorithm that does not use deblurring. We can conclude that deblurring makes the algorithm more tolerant to noise.

An explanation of this effects is that with deblurring the data influences not only a voxel but also the nearby voxels. Due to the noise, a point of the real world may be back projected to different voxels by two different cameras. If no deblurring is done then this data redundacy will be lost while deblurring will manage to detect that there is a high density of occupied voxels in that section.

The optimizations done to the algorithm make the problem treatable in a reasonable time. Without them the iteration time would be as costly as the precomputing time. The current state of the algorithm allows to attempt an online reconstruction for convolution with range 1 with a powerful computer. In order to do that the postprocessing part has to be improved. Bigger values of the convolution range force the superresolution to be computed offline.

7.2. Future work

Results encourage doing more work on geometry superresolution. More choices of the superresolution operator could be studied. This algorithm admits with little change any linear operator. The main objective of this thesis is to test the effect of superresolution in geometry reconstruction. In order to make the test as accurate as possible, all the other components have been kept as simple as possible in order that their effect in the result is minimal.

The actual data term is quite simple. It would be interesting to expand it to represent better the reality. Using signed distances as proposed by Bylow et al in [5] may give better results. The main problem of this approach is that it would make impossible the precomputing of the data term and the data term would need to be dualized, making the algorithm much more costly. Another possible improvement to the visibility function is to give more reliability to the voxels that are seen directly by the camera and discard the voxels that are only tangentially seen. Also other choices for the regularizer can be studied.

The current state of the algorithm works with known trajectory and static scene. It can be tried to make the algorithm a part of another algorithm that solves a more complex problem, such as tracking or optical flow.

Appendix

A. Software components

The programme has been designed using the modula design pattern. Each module coincides with a functionality of the programme. The programme has been made so it is compatible with some of the oldest versions of CUDA, in which the extern linking for variables is not supported. In figure A.1 a module diagram of the programme is found.

Basic functionalities This module contains basic functions and data structures that should be accessed from all the other modules. It contains the narrow band structure, as well as the math functions such as divergence and gradient and the needed structures to work with vectors and quaternions. It also contains the coordinate changes functions between the image domain and the volume domain.

The cuda functions for handling errors are also defined here.

Compute This module is responsible for the iterations of the primal dual iterations and the convergence checking.

Images input/output This module is used to read and write images and trajectory information. The interpolation to make the timestamps match is also done here. The library *lodepng* is used ([25]) to read and write png images from a file.

Initialization This module is the responsible of computing the narrow band and the data term.

Parameters This module contains all the variables and constants that should be accessible from all the other parts of the programme. It also contains the computing of the taylor polynomial.

Postprocessing This module is responsible of doing the threshold, compute the boundary of the narrow band and store it in disk.

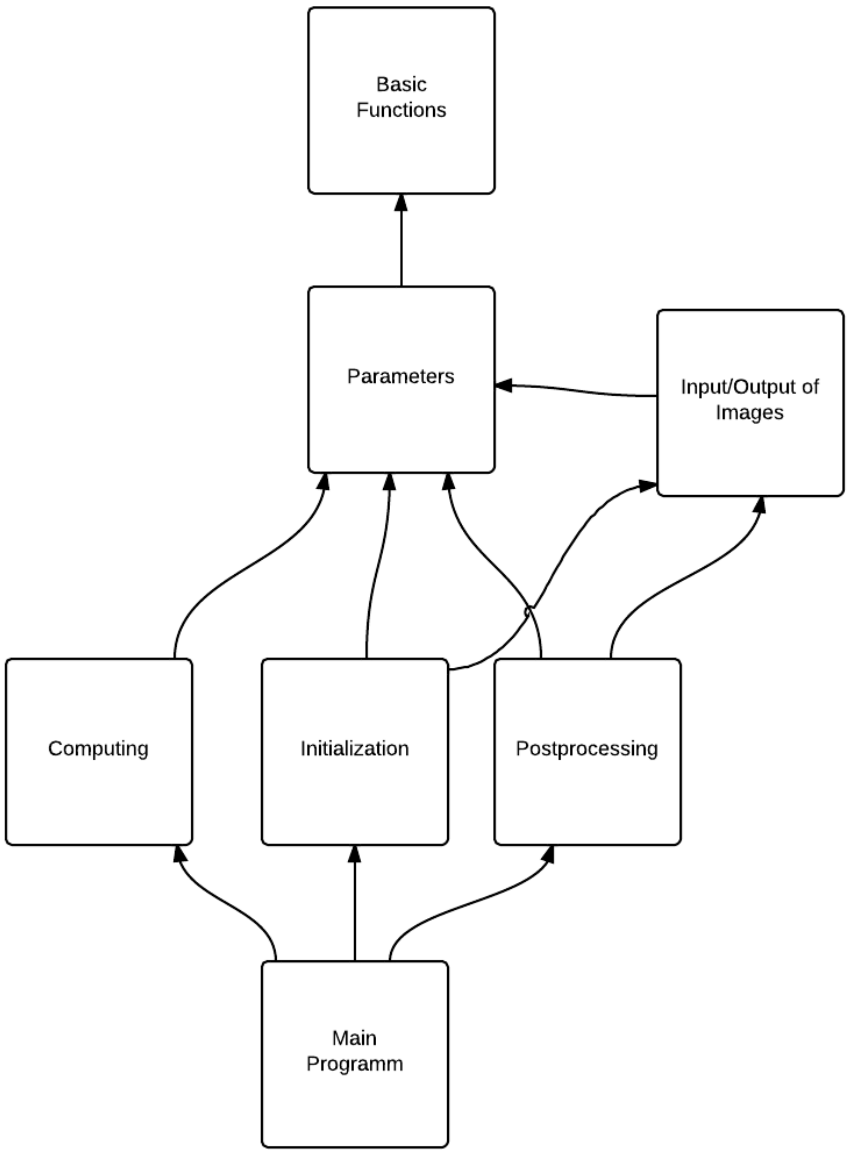


Figure A.1.: Module diagram of the programme.

B. Testing and evaluation

B.1. Evaluation tools

In this sections the tools used for evaluating the executions of the algorithm. The used hardware is a computer with 32 GB of RAM memory, a Intel®Core™i7-3770 CPU @ 3.40GHz processor and a NVIDIA®Tesla™C1060.

B.1.1. GUI interface

An interface in QT has been implemented to help the analysis of results. This tool allows to enter the problem parameters through the graphic interface and display the resulting depth maps. It also allows to make zoom in the parts of the image and rescale the depth values.

Depth Map

Once the reconstruction is done, then the depth map of the reconstructed object from a certain point can be computed. This allows to compare the output depth map with the input ones and get an idea of how the reconstructed surface looks like. Algorithm 4 shows how the depth map is computed.

Algorithm 4 Compute depth map

```
$Foto \leftarrow \text{inf}$
for all $x \in NarrowBand$ do
    $pixel \leftarrow \text{projectivity}^{-1} \circ R(x)$
    $pixelDepth \leftarrow R(x).z$
    $Foto[pixel] \leftarrow min(Foto[pixel], pixelDepth)$
end for
```

B.1.2. GL viewer

A simple OpenGL viewer has been developed for evaluating the result. The viewer displays a coloured surface and allows to rotate around it with 3 degrees of freedom, as well as doing zoom and move the viewing point. The color is obtained from several cameras, as described in algorithm 5

Algorithm 5 Load color

```
for all $x \in$ narrowBand do
    $rgb \leftarrow (0,0,0)$
    $n \leftarrow 0$
    for all $c \in Cameras$ do
        $pixel \leftarrow \pi_c(x)$
        if $|Rx_3 - distanceCamera(pixel)| < \delta$ then
            $n \leftarrow n + 1$
            $rgb \leftarrow rgb + \text{color}_c[pixel]$
        end if
    end for
    $\text{color}[x] = \frac{rgb}{n}$
end for
```

Characteristic	Value
Voxel grid	$256 \times 256 \times 256$
Depth map resolution (approx)	70×60
Distance between voxels	1.75 mm
Distance from camera (approx)	1.33 m
Camera	Freiburg3

Table B.1.: Characteristics of the plane

B.2. Testing envioronment

The depth data used in this thesis is sampled from a Kinect . The data is described in [22], as well as instructions of how to download and use the data. The sampling device has an accelerometer, a depth camera and a rgb camera. Both cameras have a resolution of 640×480 pixels.

Each dataset has depth maps, color images as well as the trajectory and information of the camera parameters. Unfortunately, the sampling moments of the depth maps, images and trajectory do not coincide. In order to make them coincide, the data is linearly interpolated between two consecutive timestamps, so one can acquire an approximation of the depth maps at the moments in which the position and orientation of the camera is known.

Several objects has been used to test the superresolution method. Here is a description of their main properties.

B.2.1. Plane

This is the object used in most of the tests. It is a relative flat and small object, fact that helps to do the computations. Another important thing is that its boundary is more complex than the simple dice, thing that allows to be used to test whether there have been or not improvements in the quality of the details. The technical characteristics can be found in table B.1 and example images in figure B.1.

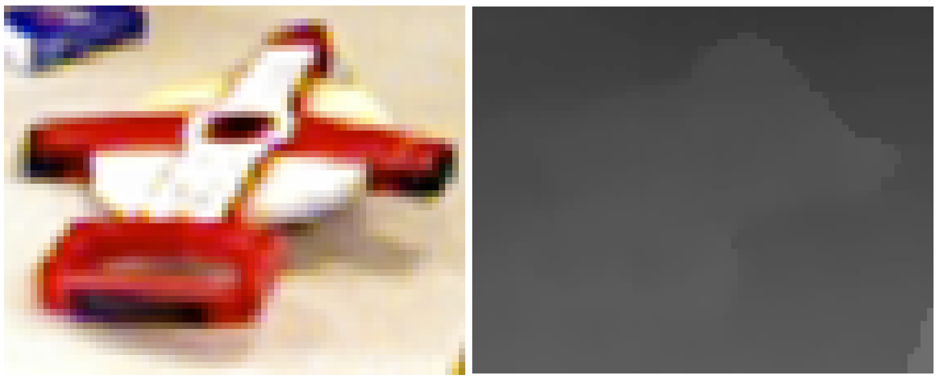

(a) RGB photo of the plane (b) Depth map of the plane

Figure B.1.: This figure contains a depth map and a RGB photo of the plane. Both images are taken from the dataset. Notice that the left wing can not be barely distinguished from the surface and the right wing is heavily pixelized

Characteristic	**Value**
Voxel grid	$256 \times 256 \times 256$
Depth map resolution (approx)	60×60
Distance between voxels	1.75 mm
Distance from camera (approx)	2.29331
Camera	Freiburg3

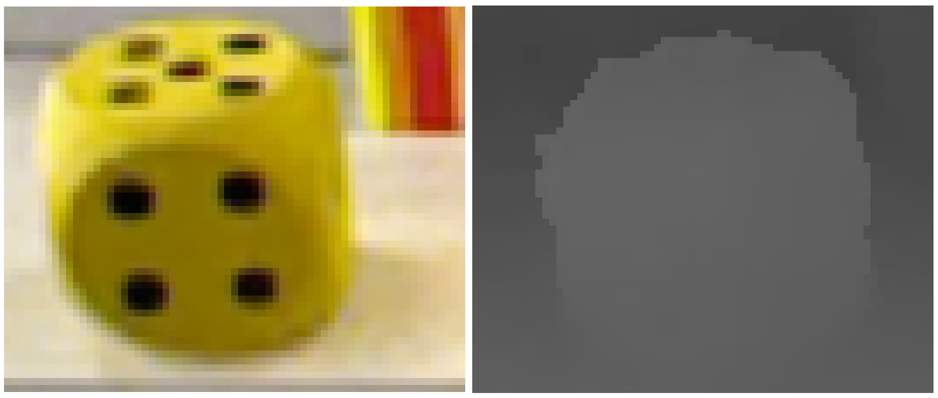
Table B.2.: Characteristics of the teddy bear

B.2.2. Dice

The dice is a simple cubic object that belongs to the desk. Its main flaw is its lack of details, so the augment of resolution can not be easily observed. The technichal characteristics can be found in table B.2 and example images in figure B.2.

B.2.3. Teddy

The teddy bear is a relatively big object in the sense that it occupies most of the depth map. It has several details in the head, but they appear with a relatively high resolution in the initial data, so it is not a good reference for testing the superresolution. The right eye of the teddy is reconstructed using superresolution. It has also details in its feet.

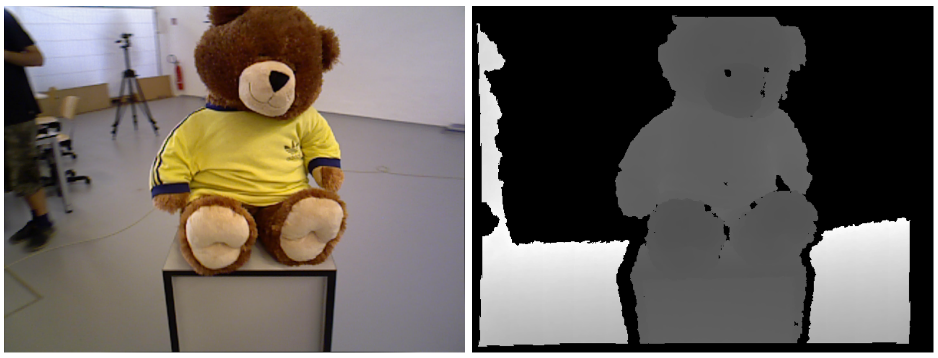


(a) RGB photo of the dice (b) Depth map of the dice

Figure B.2.: This figure contains a depth map and a RGB photo of the dice. Both images are taken from the dataset. Notice the low quality of the input depth map.

Characteristic	**Value**
Voxel grid	$1024 \times 1024 \times 1024$
Depth map resolution (approx)	300×480
Distance between voxels	0.35 mm
Distance from camera (approx)	0.88 m
Camera	Freiburg3

Table B.3.: Characteristics of the teddy bear



(a) RGB photo of the teddy bear (b) Depth map of the teddy bear

Figure B.3.: This figure contains a depth map and a RGB photo of the teddy bear. Both images are taken from the dataset. Notice the wrinkles in the shirt that can be appreciated RGB photo, but they are not visible in the depth map.

Bibliography

[1] Adrien Angeli Ankur Handa, Richard A. Newcombe and Andrew J. Davison. Applications of legendre-fenchel transformation to computer vision problems. Technical Report DTR11-7, Imperial College - Department of Computing, September 2011.

[2] Yuri Boykov and Gareth Funka-Lea. Graph cuts and efficient n-d image segmentation. *Int. J. Comput. Vision*, 70(2):109–131, November 2006.

[3] Andrea Braides. *A handbook of Γ-convergence*. Handbook of Differential Equations. Stationary Partial Differential Equations, Volume 3, 2006.

[4] D. C. Brown. Decentering distortion of lenses. *Photogrammetric Engineering*, 32(3):444–462, 1966.

[5] Erik Bylow, Juergen Sturm, Christian Kerl, Fredrik Kahl, and Daniel Cremers. Real-time camera tracking and 3d reconstruction using signed distance functions. In *Proceedings of Robotics: Science and Systems*, Berlin, Germany, June 2013.

[6] A. Chambolle, V. Caselles, D. Cremers, M. Novaga, and T. Pock. An introduction to total variation for image analysis. In *Theoretical Foundations and Numerical Methods for Sparse Recovery*. De Gruyter, 2010.

[7] A. Chambolle, D. Cremers, and T. Pock. A convex approach to minimal partitions. *SIAM Journal on Imaging Sciences*, 5(4):1113–1158, 2012.

[8] Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *J. Math. Imaging Vis.*, 40(1):120–145, May 2011.

[9] Tony F. Chan, Jianhong Shen, and Hao-Min Zhou. Total variation wavelet inpainting. *J. Math. Imaging Vision*, 25:107–125, 2006.

[10] D. Cremers, T. Pock, K. Kolev, and A. Chambolle. Convex relaxation techniques for segmentation, stereo and multiview reconstruction. In *Markov Random Fields for Vision and Image Processing*. MIT Press, 2011.

[11] Ernie Esser, Xiaoqun Zhang, and Tony F. Chan. A general framework for a class of first order primal-dual algorithms for convex optimization in imaging science. *SIAM J. Imaging Sciences*, 3(4):1015–1046, 2010.

[12] B. Goldluecke and D. Cremers. Superresolution texture maps for multiview reconstruction. Kyoto, Japan, 2009.

[13] Gottfried Graber, Thomas Pock, and Horst Bischof. Online 3d reconstruction using convex optimization. In *ICCV Workshops*, pages 708–711. IEEE, 2011.

[14] Per Christian Hansen, James G. Nagy, and Dianne P. O'Leary. *Deblurring Images: Matrices, Spectra, and Filtering (Fundamentals of Algorithms 3) (Fundamentals of Algorithms)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2006.

[15] Pushmeet Kohli and Philip H. S. Torr. Dynamic graph cuts and their applications in computer vision. In Roberto Cipolla, Sebastiano Battiato, and Giovanni Maria Farinella, editors, *Computer Vision: Detection, Recognition and Reconstruction*, volume 285 of *Studies in Computational Intelligence*, pages 51–108. Springer, 2010.

[16] E. L. Kosarev. Shannon's superresolution limit for signal recovery. *Inverse Problems*, 6:55–76, February 1990.

[17] NVIDIA Corporation. *NVIDIA CUDA Compute Unified Device Architecture Programming Guide*. NVIDIA Corporation, 2007.

[18] T. Pock, T. Schoenemann, G. Graber, H. Bischof, and D. Cremers. A convex formulation of continuous multi-label problems. In *European Conference on Computer Vision (ECCV)*, Marseille, France, October 2008.

[19] Thomas Pock and Antonin Chambolle. Diagonal preconditioning for first order primal-dual algorithms in convex optimization. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 1762–1769, 2011.

[20] R.T. Rockafellar. *Convex Analysis*. Princeton University press, 1972.

[21] Sebastian Schuon, Christian Theobalt, James Davis, and Sebastian Thrun. High-quality scanning using time-offlight depth superresolution. In *In IEEE CVPR Workshop on Time-of-Flight Computer Vision, page NN, 2008 (In*. Press.

[22] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*.

[23] Christian Theobalt, James Davis, and Sebastian Thrun. Lidarboost: Depth superresolution for tof 3d shape scanning.

[24] Markus Unger, Thomas Pock, Manuel Werlberger, and Horst Bischof. A convex approach for variational super-resolution. In *Proceedings German Association for Pattern Recognition (DAGM)*, volume 6376 of *LNCS*, pages 313–322. Springer, 2010.

[25] Lode Vandevenne. `http://lodev.org/lodepng/`.

[26] Wikipedia. Pinhole camera model — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/wiki/Pinhole_camera_model`, 2013. [Online; accessed 22-July-2013].