Master's Thesis

Multi – View 3D Reconstruction with Variational Method

by

Sergey Kosov

Thesis handed in on: 22 Jan 2008

Supervisor:
Prof. Dr. Joachim Weickert

Advisor:
Dr. Andrés Bruhn

Reviewers:
Prof. Dr. Joachim Weickert
Prof. Dr. Michael Kröning

Mathematical Image Analysis Group, Department of Computer Science,
Saarland University, 66041 Saarbrücken

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have used only the resources given in the list of references.

Also I authorize the Saarland University to publish this thesis in networks and/or data bases.

Sergey Kosov.

Abstract

This paper describes a variational approach to dense stereo reconstruction, which combines powerful tools such as regularization and multi–scale processing to estimate directly depth from a number of stereo images, while preserving depth discontinuities. Variational methods currently belong to the most accurate techniques for the computation of the displacement field between the frames of stereo images. In the past few years this problem has raised a great deal of interest due to the increasing number of applications, both in vision and in graphics, where this problem has become of crucial importance. Accuracy and time performance improvements of these methods are achieved every year. Most of the efforts are directed towards finding better data and smoothness terms for the energy functional. In this paper different data terms as soon as different smoothness terms are considered. During the work on variational approach two methods have been developed: the depth-driven method, where depth is computed directly from the grey-level images; and the disparity–driven method, where depth is computed from calculated disparity map. Also in this paper a number of novel principles and techniques are offered which allow looking at the application of variational methods in computer vision from a new point of view. All the methods can also deal with any number (greater than one) of cameras. Moreover, to solve the problem numerically, a row of PDE–based schemes have been applied. Experimental results illustrate the capabilities and shortcomings of these methods.

Acknowledgements

First of all, I would like to gratefully acknowledge my supervisor Prof. Joachim Weickert, who introduced me to the fascinating world of Image Analysis and Computer Vision and who, through his passion for the subject, convinced me go into this field.

I am indebted to Prof. Michael Kröning for giving me an opportunity to start with my graduation. I have never seen a teacher with such a thoughtful vision, lively ideas and humanity.

I owe my highest gratitude to Dr. Andrés Bruhn for giving me the chance to work in the Mathematical Image Analysis Group at the Saarland University and for his help and supervision during the development of this thesis.

I thank all of my friends for their support, and the wonderful time we had spent together.

And last but not least, I thank my parents for their ultimate support during all of these years.

Sergey Kosov.

Contents

„Even mathematics needs it, even the invention of integral and differential calculus could be impossible without imagination. Imagination is the quality of the highest value."

– Vladimir Ilyich Lenin.

Chapter 1

Introduction

1.1 Problem description

One of the key classical and long-debated correspondence problems in Computer Vision is the reconstruction of a detailed depth-map from a set of cameras. The cameras give us an ordered set of pictures – a stereo image and we would like to be able to correctly estimate the distance from an observer to the objects represented in such a picture array. These distances values in the image points constitute the depth-map. In order to compute the depth-map we need to compare certain features that stay invariant in the picture set and that can help us to identify the objects and distances to them in the scene. These features we call *invariance* or *constancy assumptions*.

So far, most of the research done in the field of application of variational methods to computer vision problems has been directed to computation of the optic flow field between the frames of an image sequences. In this thesis we apply that experience to the problem of the depth-map reconstruction from a set of cameras.

The main goal of this thesis is to investigate the application of variational methods to the problem of depth-map reconstruction; to formulate method's strong points and advantages over other map reconstruction methods as well as its shortcoming; to place the method among existing map reconstruction methods. Another primary goal of the thesis is to define how the number of cameras and their position / orientation influences the resulting depth-map. But let us first formulate the reconstruction problem in a more formal manner.

1.2 What is a depth-map?

The term "depth-map" refers to a visual phenomenon that we experience every day. Humans like all other mammals make use of the *binocular vision* – the vision in which both eyes are used together. It gives four advantages: a spear eye if one is damaged, a wider field of view, binocular summation and *stereopsis*[1]. The human vision system gives a precise depth perception due to the stereopsis in which a parallax is provided by the two eyes' different position. A machine has the advantage to have more than two eyes, and vary their position on body.

Having a stereo image (a picture set) we would like to estimate the distances to the objects, represented at the image. For this purpose, we take two pictures of the set (left and right, for example) and try to estimate the position to which every pixel from the first picture has moved in the second picture. Having this information we calculate the depth value in the pixel.

A picture is represented as a pixel matrix, where every pixel has a unique coordinate position. In order to represent the problem mathematically we consider a scalar-valued picture set $I_i(u,v)$, where $(u,v)^T$ denotes the coordinates of the pixel.

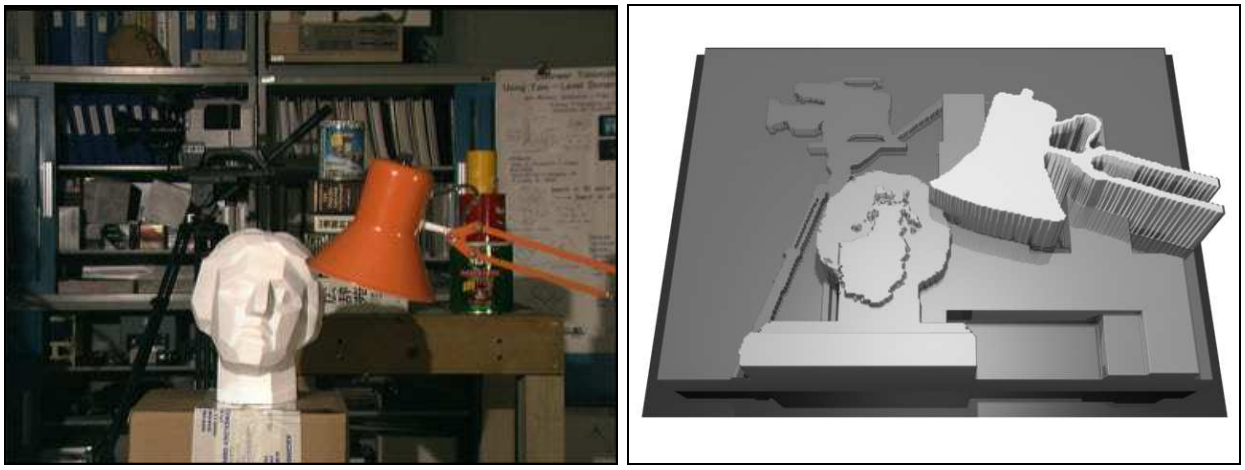

Figure 1.1: "Tsukuba" scene: **Left:** The left frame of scene; **Right:** The ground truth depth-map, estimated between the left and the right pictures of the same set.

Let the left picture be denoted by $I_l(u,v)$. Then the right picture will be denoted by $I_r\left(\xi(u,v), \zeta(u,v)\right)$ where $\xi(u,v)$ and $\zeta(u,v)$ show the new position of the pixel, which was previously at position $(u,v)^T$. And the depth-map value in this pixel is an another function: $z(u,v) = f\left(\xi(u,v), \zeta(u,v)\right)$. As we are working with real-world data, which is continuous, not discrete, like the pixel notation, the actual displacement position coordinates are not necessary integer values.

Having formulated the problem like that, we can state that the computation of the depth-map is actually the computation of the depth values for every pixel of an image (see Figure 1.1).

[1] An ability to make fine depth discriminations.

1.3 Fields of application

The depth-map has many fields of application:

The main area of depth-map application is the robotics branch. A robot gets information about the surrounding world through its visual sensors system. Having the depth-map it can determine how far the objects around him are situated and even determine its own position in the world. Of course without this information, machine can not make movement decisions or even operate with nearby objects. Moreover, the depth-map helps the robot to make a segmentation of the world's objects and recognize them (figure 1.2, left picture).

Another wide area of depth-map application lies in aeronautics and aero graphics. Calculated depth-map helps to define the altitude and the speed of a flying aircraft or a satellite. Coming to the problem from other side, a satellite or an aircraft can determine the earth surface height above sea level, or cities infra structure, what is very useful in region recognition.

Figure 1.2: Application fields of the depth-field: **Left:** Robot eyes ([WRM07]); **Centre:** Human face reconstruction ([YHR04]); **Right:** Driver assistance system ([KB07]).

A relatively similar way of using the depth-field is applied in the driver assistance systems. Novel car computers are capable to determine the motion around a car and distances to the other cars at the street. Such systems allow preserving car from hijacking or having an accident by making it stop or just by warning the driver that a dangerous situation is approaching (figure 1.2, right picture).

Also depth-maps are widely spread in computer graphics industry. They allow creating virtual models of real existing objects. For example it is enough to make 2 - 3 photos from different positions to reconstruct the human face in 3 dimensional virtual worlds (figure 1.2, centre picture).

Science also uses depth-maps, for example, in crack detection. Using ultra sound or X-rays it is possible to look inside a probe without destroying it. Depth-map helps to determine the ultrasound or X-ray sensor position on the investigating object's surface.

These were only a few examples that show the usefulness of the depth estimation as a computer vision problem. And for every specific application there are specific problems exist. With no doubts they should be taken into account and should be considered, but the more important thing is to develop a reliable depth-map computation algorithm for a general case.

1.4 Definitions and notations

We denote the set of *natural numbers* as N and use N^+ for $\mathrm{N} \cup \{0\}$. The set of *integer* numbers is given by $\mathrm{Z} = \{...,-1,0,1,...\}$. The set of *real* and *rational* numbers are denoted as R and Q respectively.

Let $f \equiv f(x,y)$ (this means "*identical with*") be a smooth function; "*smooth*" means it is as many times continuously differentiable in the variable x and y as required. Further we denote:

$$f_x(x,y) := \frac{\partial f(x,y)}{\partial x}$$
$$f_y(x,y) := \frac{\partial f(x,y)}{\partial y}$$
$$\nabla f := \begin{pmatrix} f_x \\ f_y \end{pmatrix} - \textit{nabla operator}$$
$$\nabla \cdot \nabla f = \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{pmatrix} \cdot \begin{pmatrix} f_x \\ f_y \end{pmatrix} = \frac{\partial}{\partial x} f_x + \frac{\partial}{\partial y} f_y = f_{xx} + f_{yy} = \Delta f$$

Definition 1: Let $m, n \in \mathrm{Z}$ be numbers. We define **integer intervals** as

$$[m;n] = m,...,n$$
$$[m;n) = m,...,n-1$$
$$(m;n] = m+1,...,n$$

Definition 2: Any column-vector can be transformed into the corresponding row-vector and vise versa by means of the **transpose** operation. Let v be a row-vector, then

$$v^T = (v_0, v_1, ..., v_{n-1})^T = \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{n-1} \end{pmatrix}$$

Definition 3: Let v be a vector, than its **length** is defined as

$$|v| = n$$

Definition 4: The highest occurring derivative in a partial differential equation (PDE) determines its order. For example:

- $f_x + f_y = 0$ – is a first order PDE;
- $f_x + f_{yy} = 0$ – is a second order PDE;
- $\Delta f = 0$ – is a second order PDE;
- $f_{xxx} + f_y = 0$ – is a third order PDE.

Definition 5: In a PDE relating a function f to its derivatives, we denote sometimes

- f as the dependent variable (as it is not known and depends on x and y.
- x and y as the independent variables.

Definition 6: Non linear PDEs in a dependent variable f arise by coefficients of derivatives of f depending on f itself. For example:

- $\Delta f = 0$ – is a linear PDE;
- $f \cdot \Delta f = 0$ – is a non linear PDE;
- $\nabla \cdot (f^2 \Delta f) = 0$ – is a non linear PDE.

Definition 7: In order to convert an analog signal into its digital representation we use the **sampling** procedure. Let $f : R \to R$ be an analog function. Then the corresponding discrete function $f \in R^n$ is given for all $\forall k \in [0; n-1]$ as

$$f_k = f(k\Delta x),$$

where $n \in \mathrm{N}^+$ is a number of samples and $\Delta x \in R$ is a **sampling interval**.

Definition 8: We begin by noticing, that since $f'(x) = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x} = \lim_{\Delta x \to 0} \frac{f(x) - f(x-\Delta x)}{\Delta x} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x}$ a reasonable approximation of $f_x(k\Delta x)$ can be given by

- $\frac{f_{i+1} - f_i}{\Delta x}$ – *forward difference* approximation;
- $\frac{f_i - f_{i-1}}{\Delta x}$ – *backward difference* approximation;
- $\frac{f_{i+1} - f_{i-1}}{2\Delta x}$ – *centre difference* approximation;

in a similar manner we approximate $f_{xx}(k\Delta x)$ at

- $\frac{f_{i+1} - 2f_i + f_{i-1}}{\Delta x^2}$.

1.5 Outline of the thesis

Variational methods are the most precise way, so far, to compute the optic flow. In Chapter 2 the general structure of variational methods is briefly explained, together with the meanings of the data and the smoothness terms. We adopt the famous variational methods of Horn-Schunck and Brox *et al*. which were developed for optic flow computation, to the depth-map reconstruction problem.

As the main goal of this thesis is the investigation of application of variational methods to the depth-map reconstruction problem, in Chapter 3 we create a mathematical model for correspondence problem, discuss constancy assumptions for data term and chose a suitable smoothness term relying on the diffusion processes theory. Also in this chapter we introduce the technique of automatic controlling the variational process during solution, capable to improve the final result, handle on-run solver mistakes and speed up the convergence process.

In Chapter 4 we discuss the depth-map reconstruction based on information from two cameras. We formulate principles and criteria for the reconstruction and offer new methods in addition to classical optic-flow methods, adopted for our problem. Later in the chapter, we step by step derive several numerical schemes for implementing and resolving the problem on a computer and achieve first results.

In Chapter 5 as a primary goal of the thesis, we extend all the theory foundation, gained in previous Chapter, for the multi – view case.

This thesis is closed with the Chapter 6, where we combine the theory from the above chapters in one and present and discuss our experimental results. After that we make the conclusion of the thesis.

Chapter 2

Variational Optic Flow Methods

Calculus of variations is a field of mathematics that deals with *functionals*[2], as opposed to ordinary calculus which deals with functions. Such functionals in our case are formed as integrals involving an unknown function and its derivatives. The interest is in *extreme* functions: those making the functional attain a maximum or minimum value [Els61]. Nowadays, variational methods are among the best performing techniques in image processing for depth-map reconstruction: being global methods and thus operating on entire image domain, they recover the depth-map as the minimizer of a suitable functional, which we will call *energy functional*.

In this chapter we explain how to construct such energy functionals, we illustrate the mathematical meaning of parts, of which the functional consists, and we show how to find the unknown function – the depth-map which minimizes the constructed energy functional.

2.1 General structure

Let us suppose that we are given a stereo image, represented as a number of pictures of a certain scene from different positions and angles $I_i(u,v)$, $i=1,...,N$, where (u,v) denotes the point coordinates in image. And moreover let us suppose that we would like to compute the depth-map field $z(u,v)$. According to variational methods we should construct an energy functional which has the following structure:

$$E(z(u,v)) = \iint_{\Omega} F\left(I_1(u,v),...,I_N(u,v), z(u,v), \frac{\partial z(u,v)}{\partial u}, \frac{\partial z(u,v)}{\partial v} \right) du dv \qquad (2.1.1)$$

[2] The functions that take functions as theirs arguments.

or in more general form:

$$E\big(z(u,v)\big)=\iint_{\Omega} F\left(u,v,z(u,v),\frac{\partial z(u,v)}{\partial u},\frac{\partial z(u,v)}{\partial v}\right)dudv\ . \tag{2.1.2}$$

The integration domain Ω – is the entire image domain. Here the unknown function $z(u,v)$ has the dimensions which equal to the dimensions of the input set of images. For the simplicity, let us introduce the following notations:

$$z_u := \frac{\partial z(u,v)}{\partial u},\ z_v := \frac{\partial z(u,v)}{\partial v}. \tag{2.1.3}$$

The functional $E\big(z(u,v)\big)$ consists of two terms: the *data term* and the *smoothness term*. While the data term provides us with the information about depth, the smoothness term distributes this information:

$$E(z)=\iint_{\Omega} Data\_term(u,v,z)+Smothness\_term(z_u,z_v)dudv\,. \tag{2.1.4}$$

Such an approach guaranties us that the computed depth-map will be always dense. For example, in a case of data term can not give us in some region more or less useful information, the smoothness term fills in this region with information, computed from neighboring regions.

2.1.1 Presmoothing step

Instead of using the original image set $I_i(u,v)$ as input, we will use the presmoothed versions of it. It will help us to get rid of small noise and disturbances in pictures, make discrete image data more suitable for calculating its derivatives via difference schemes and thus improve the final result. The presmoothing step could be done with the help of Gaussian low-pass filtering, e.g. via convolution with a Gaussian kernel K_ρ, where ρ is the standard deviation:

$$I_i^\rho = K_\rho * I_i\,. \tag{2.1.5}$$

The parameter ρ is a very important parameter since it greatly affects the final result of the depth-map computation. From this place and so on, we will write I_i without ρ, but mention already presmoothed image.

2.1.2 Minimization

Our main goal is to find such a function $z(u,v)$, which minimizes the energy functional $E(z(u,v))$. By another words, having constructed the energy functional, we should minimize it in order to find the best solution for the depth-map. Moreover, if the constructed functional (2.1.2) is *strictly convex*[3], it will have a unique solution that minimizes it.

The major formula of the calculus of variation was developed by *Leonard Euler* (April 15, 1707 – September 18, 1783) and *Joseph-Lois Lagrange* (January 25, 1736 – April 10, 1813) in the 1750s. In an honor of these mathematicians it is called *the Euler-Lagrange equation*.

The Euler-Lagrange equation is an equation satisfied by a function z of the parameters u and v which extremises the functional (2.1.2), and F is a given function which has continuous first order partial derivatives. The Euler-Lagrange equation then is the partial differential equation:

$$F_z - \frac{d}{du}F_{z_u} - \frac{d}{dv}F_{z_v} = 0, \qquad (2.1.6)$$

where the unknown function $z(u,v)$ must necessary satisfy this equation.

In that way, in order to minimize our energy functional, we should solve the Euler-Lagrange equation with homogeneous Neumann boundary conditions. This step is done via discrete numerical schemes. We are working with discrete images which consist of atom picture elements – *pixels*. The Euler-Lagrange equation is discretized and approximated via finite-differences schemes. At the end we have linear or non-linear system of equations. To solve it, a number of iterative numerical schemes has been developed.

2.2 Construction of the energy functional

Since the energy functional consists of two parts – the data term and the smoothness term (2.1.4) and both terms have its own meaning, we will build them separately. For simplicity in this chapter as well as in the next two chapters we consider the case of a stereo image given by only *2* pictures. All the theory derived for the case of *2* pictures is valid in the multi-picture case, and we will consider the extension to *N* cameras in the fifth chapter.

Data term. The data term is a number of combined assumptions that certain features of the image do not change, but remain constant from one picture to another.

[3] A real-valued function f defined on an interval is called convex if for any two points x and y in its domain and any t in [0; 1], we have $f(tx+(1-t)y) \le tf(x)+(1-t)f(y)$.

The data term is responsible for supplying us with information about depth-field and stand for the constancy assumptions that are used. The most general assumption used in this paper is the *brightness constancy* assumption.

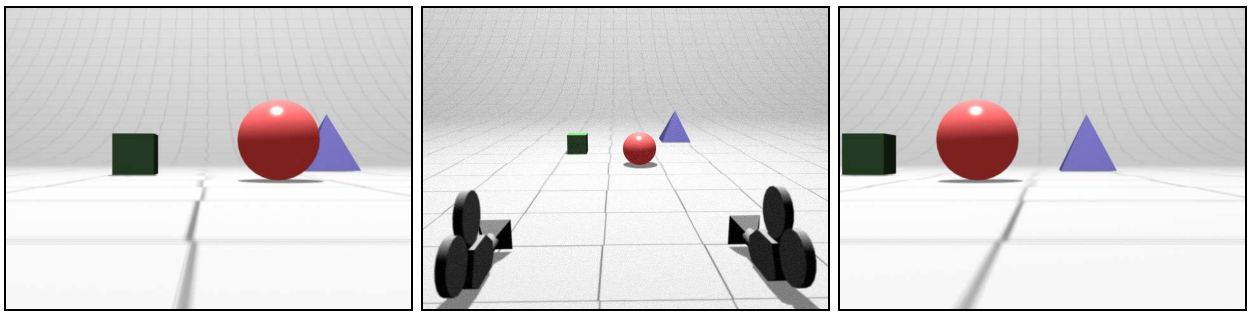

Figure 2.1: "Geometric primitives" scene: **Left:** Picture achieved by the left camera; **Centre:** The whole scene; **Right:** Picture achieved by the right camera

To explain the main point of the grey-value constancy assumption, let us consider figure 2.1. At the centre of the figure we can see "Geometric primitives" scene with two cameras, shooting it to achieve a stereo image. At the right and at the left we observe pictures gained by the right and the left cameras correspondingly. As we have noticed, all the primitives shot by the left camera, are shifted relative to the primitives, shot by the right camera. And the red ball has bigger shift than more distant from cameras object – blue pyramid. Also we can notice that green cube, which is situated between the red ball and the blue pyramid in depth, has smaller shift than the ball, but bigger shift than the pyramid. So, if we could estimate this shift, and knowing the distance between the cameras, we could calculate the distances to the objects in scene.

The grey-value constancy assumption is based on assumption that all the objects in scene have *lambertian surfaces*[4]. It means that the grey value of a pixel is not changed by the displacement:

$$I_1(u_1, v_1) = I_2(u_2, v_2)\,. \tag{2.2.1}$$

This literally means that if $I_1(u_1, v_1)$ is the grey value of pixel at point (u_1, v_1) in the first picture, this value remains equal to the grey value of pixel at point (u_2, v_2) in the second picture. It is also very important to express the coordinates (u_1, v_1) through the coordinates (u_2, v_2) and the unknown function $z(u_2, v_2)$:

$$\begin{cases} u_1 = \xi(u_2, v_2, z(u_2, v_2)); \\ v_1 = \zeta(u_2, v_2, z(u_2, v_2)). \end{cases} \tag{2.2.2}$$

Pay your attention that if $I_1(u_1, v_1)$ is another feature of the picture, which remains constant from one picture to another of a stereo image, the statement (2.2.1) is still valid.

We can rewrite the equality (2.2.1) in the following form:

[4] The perfect diffuse surfaces that scatter incident illumination equally in all directions.

$$I_1(u_1, v_1) - I_2(u_2, v_2) = 0. \tag{2.2.3}$$

But there is nothing ideal in real life, so the equality (2.2.3) as usual is not true. But we can fulfill the following demand:

$$\left|I_1(u_1, v_1) - I_2(u_2, v_2)\right| \to \min, \tag{2.2.4}$$

or

$$\left|I_1(u_1, v_1) - I_2(u_2, v_2)\right|^2 \to \min. \tag{2.2.5}$$

We put the square in the formula (2.2.5) because of two reasons: we want to penalize positive and negative deviations in the same way and the quadratic penalizer leads to the linear system of equations. In the section 3.5 we discuss other penalizing functions. Now we can write down the simplest data term based on the grey-value constancy assumption:

$$Data\_term(u, v, z) = \left|I_1(u_1, v_1) - I_2(u_2, v_2)\right|^2, \tag{2.2.6}$$

with coordinates expression (2.2.2).

Smoothness term. The smoothness term stands for the assumption that the neighboring regions belong to the same object and thus these regions have similar depth. The main role of the smoothness term is the redistribution of the computed information and smoothing of depth outliers. In case we get no reliable information from the data term, the smoothness term will realize its smoothing effect by filling in the problem region with data, calculated from neighboring regions.

In fact, we introduce here an additional assumption that the depth-map is globally smooth – a *smoothness* assumption. In such a way, we can write down the simplest smoothness term, constituted by the *homogeneous regularizer*:

$$Smoothness\_term(z_u, z_v) = \left|\nabla z\right|^2, \tag{2.2.7}$$

where $\left|\nabla z\right| = \sqrt{z_u^2 + z_v^2}$.

As usual to control how much the smoothness term will prevail during the computation, the parameter φ is introduced. If the parameter φ is larger, then the computed depth-map will be smoother:

$$Smoothness\_term(z_u, z_v) = \varphi \cdot \left|\nabla z\right|^2. \tag{2.2.8}$$

Having constructed the data term and the smoothness term, we now can write down the energy functional:

$$E(z) = \iint_{\Omega} \left|I_1(u_1, v_1) - I_2(u_2, v_2)\right|^2 + \varphi \cdot \left|\nabla z\right|^2 du dv\,, \tag{2.2.9}$$

or, taking into account (2.2.2):

$$E(z) = \iint_{\Omega} \left|I_1(\xi, \zeta) - I_2(u, v)\right|^2 + \varphi \cdot \left|\nabla z\right|^2 du dv\,. \tag{2.2.10}$$

Now we have to derive the Euler-Lagrange equation (2.1.6) from the functional (2.2.10). Let us do it step by step to understand how it works.

$$\frac{\partial F}{\partial z} = 2 \cdot \left(I_1(\xi, \zeta) - I_2(u, v)\right) \cdot \left(I_{1u}(\xi, \zeta)\xi_z + I_{1v}(\xi, \zeta)\zeta_z\right) \tag{2.2.11}$$

Pay your attention, that in the formula (2.2.11) we have the derivative only from data term, since in smoothness term no function $z(u,v)$ exists, but only its derivatives. Thus this part of the Euler-Lagrange we will call *data term of the Euler-Lagrange equation*.

$$\frac{d}{du} F_{z_u} + \frac{d}{dv} F_{z_v} = 2\varphi \cdot div(\nabla z) = 2\varphi \cdot \Delta z \tag{2.2.12}$$

We have got here Laplacian of unknown function $z(u,v)$: $\Delta z = \frac{\partial^2 z}{\partial u^2} + \frac{\partial^2 z}{\partial v^2}$. Correspondingly to the smoothness term, this part of the Euler-Lagrange we will call *smoothness term of the Euler-Lagrange equation*. The Laplacian is the core of linear diffusion process, which is identical to the Gaussian blurring process.

Combining (2.2.10) and (2.2.11) together and cancelling the common factor *2*, we can write the Euler-Lagrange equation for the functional (2.2.9):

$$\left(I_{1u}(\xi, \zeta)\xi_z + I_{1v}(\xi, \zeta)\zeta_z\right) \cdot \left(I_1(\xi, \zeta) - I_2(u, v)\right) - \varphi \cdot \Delta z = 0\,. \tag{2.2.13}$$

After discretizing this equation we have linear or non-linear system. That depends on the relations between u and z, and between v and z in functions ξ_z and ζ_z. A number of numerical schemes has been developed for efficiently solving such linear and non-linear equations [Bru06].

2.3 The Euler-Lagrange equation

In this section we will construct more sophisticated energy functional and derive its Euler-Lagrange equation. We will precede parallel with the famous Brox et al. method which was developed for optic flow problems and integrates several successful

concepts [BBPW04]. First of all let us discuss the gradient constancy assumption and then add it to our data term.

The grey value constancy assumption has one decisive drawback: It is quite susceptible to slight changes in brightness, which often appear in natural scenes. Therefore, it is useful to allow some small variations in the grey value and help to determine the displacement vector by a criterion that is less sensitive to grey value changes. Such a criterion is the gradient of the image grey value, which can also be assumed not to vary due to the displacement. This gives:

$$\nabla I_1(u_1, v_1) = \nabla I_2(u_2, v_2)\,. \tag{2.3.1}$$

This formula is very close to the one in (2.2.1). Doing the same steps like with the grey-value constancy assumption, we can formulate the corresponding constraint that must be fulfilled:

$$\left|\nabla I_1(u_1, v_1) - \nabla I_2(u_2, v_2)\right|^2 \to \min\,, \tag{2.3.2}$$

taking into account the coordinates derivation (2.2.2).

Now, we just add this constraint to our data term and use a weighted approach to control the influence of each of the assumptions during the computation:

$$Data\_term(u, v, z) = \Theta \cdot \left|I_1(u_1, v_1) - I_2(u_2, v_2)\right|^2 + (1-\Theta) \cdot \left|\nabla I_1(u_1, v_1) - \nabla I_2(u_2, v_2)\right|^2, \tag{2.3.3}$$

where $\Theta \in [0;\ \ 1]$.

For improvement we additionally introduce a Ψ function, which is in general a *penalizing* function. Now we minimize two squared constancy assumptions, but using such a quadratic penalizer gives us too much influence to outliers, that may appear during the computation in depth-map. Brox *et al*. suggested the usage of another kind of penalizing function that is given by the absolute value function:

$$\Psi(s^2) = \sqrt{s^2 + \lambda^2}\,, \tag{2.3.4}$$

where λ is a small constant.

By this definition, function (2.3.4) is still convex and that's why it is easier to minimize. And, what is more important, function Ψ penalizes the outliers in a less severe way. Moreover, we have possibility to control this penalization with the parameter λ.

The type of smoothness term that we have constructed is not always the most suitable one, since the smoothness assumption implies that the resulting depth-map has no discontinuities. In real world, which is consists of different objects these objects have different distance from an observer. And the object's boundaries should be preserved, but not smoothed

The Laplacian, that we have now, constitutes the Gaussian blurring process, which realizes smoothing in the same manner in all directions. Such an approach smoothes all the discontinuities which appear in depth-map. If we would like to preserve the discontinuities from smoothing, we should somehow decay the blurring process in discontinuities regions. It is done by imposing piecewise smoothness, by penalizing the variation from the depth-map, using the same Ψ function. It gives us more accurate and sharp results.

Now, let us add the function Ψ to the data term as well as to the smoothness term and rewrite the energy functional (2.2.8):

$$E(z) = \iint_{\Omega} \Psi\left(\Theta \cdot \left|I_1(u_1,v_1) - I_2(u_2,v_2)\right|^2 + (1-\Theta)\cdot\left|\nabla I_1(u_1,v_1) - \nabla I_2(u_2,v_2)\right|^2\right) + \varphi \cdot \Psi(\left|\nabla z\right|^2) du dv \tag{2.3.5}$$

or, reformulating the gradient:

$$E(z) = \iint_{\Omega} \Psi\begin{pmatrix} \Theta \cdot \left|I_1(u_1,v_1) - I_2(u_2,v_2)\right|^2 + \\ (1-\Theta)\cdot\left|I_{1u}(u_1,v_1) - I_{2u}(u_2,v_2)\right|^2 + \\ (1-\Theta)\cdot\left|I_{1v}(u_1,v_1) - I_{2v}(u_2,v_2)\right|^2 \end{pmatrix} + \varphi \cdot \Psi(\left|\nabla z\right|^2) du dv \tag{2.3.6}$$

and, taking into account (2.2.2):

$$E(z) = \iint_{\Omega} \Psi\begin{pmatrix} \Theta \cdot \left|I_1(\xi,\zeta) - I_2(u,v)\right|^2 + \\ (1-\Theta)\cdot\left|I_{1u}(\xi,\zeta) - I_{2u}(u,v)\right|^2 + \\ (1-\Theta)\cdot\left|I_{1v}(\xi,\zeta) - I_{2v}(u,v)\right|^2 \end{pmatrix} + \varphi \cdot \Psi(\left|\nabla z\right|^2) du dv\,. \tag{2.3.7}$$

Now let us derive the Euler-Lagrange equation for this energy functional. The data term of the Euler-Lagrange equation is given by:

$$\frac{\partial F}{\partial z} = \Psi'\begin{pmatrix} \Theta \cdot \left|I_1(\xi,\zeta) - I_2(u,v)\right|^2 + \\ (1-\Theta)\cdot\left|I_{1u}(\xi,\zeta) - I_{2u}(u,v)\right|^2 + \\ (1-\Theta)\cdot\left|I_{1v}(\xi,\zeta) - I_{2v}(u,v)\right|^2 \end{pmatrix} \cdot \begin{pmatrix} 2\Theta\cdot\left(I_1(\xi,\zeta) - I_2(u,v)\right)\cdot\left(I_{1u}(\xi,\zeta)\xi_z + I_{1v}(\xi,\zeta)\zeta_z\right) + \\ 2(1-\Theta)\cdot\left(I_{1u}(\xi,\zeta) - I_{2u}(u,v)\right)\cdot\left(I_{1uu}(\xi,\zeta)\xi_z + I_{1uv}(\xi,\zeta)\zeta_z\right) + \\ 2(1-\Theta)\cdot\left(I_{1v}(\xi,\zeta) - I_{2v}(u,v)\right)\cdot\left(I_{1vu}(\xi,\zeta)\xi_z + I_{1vv}(\xi,\zeta)\zeta_z\right) \end{pmatrix}. \tag{2.3.8}$$

And the smoothness term of the Euler-Lagrange equation reads:

$$\frac{d}{du}F_{z_u}+\frac{d}{dv}F_{z_v}=2\varphi\cdot div\left(\Psi'(\left|\nabla z\right|^2)\cdot\nabla z\right), \tag{2.3.9}$$

where $\Psi'(s^2)=\dfrac{1}{2\sqrt{s^2+\lambda^2}}$.

Now, having successfully accomplished the above calculations, canceled the common factor *2* we are ready to construct the Euler-Lagrange equation:

$$\begin{aligned}&\Psi'\begin{pmatrix}\Theta\cdot\left|I_1(\xi,\zeta)-I_2(u,v)\right|^2+\\(1-\Theta)\cdot\left|I_{1u}(\xi,\zeta)-I_{2u}(u,v)\right|^2+\\(1-\Theta)\cdot\left|I_{1v}(\xi,\zeta)-I_{2v}(u,v)\right|^2\end{pmatrix}\cdot\\&\begin{pmatrix}\Theta\cdot\left(I_1(\xi,\zeta)-I_2(u,v)\right)\cdot\left(I_{1u}(\xi,\zeta)\xi_z+I_{1v}(\xi,\zeta)\zeta_z\right)+\\(1-\Theta)\cdot\left(I_{1u}(\xi,\zeta)-I_{2u}(u,v)\right)\cdot\left(I_{1uu}(\xi,\zeta)\xi_z+I_{1uv}(\xi,\zeta)\zeta_z\right)+\\(1-\Theta)\cdot\left(I_{1v}(\xi,\zeta)-I_{2v}(u,v)\right)\cdot\left(I_{1vu}(\xi,\zeta)\xi_z+I_{1vv}(\xi,\zeta)\zeta_z\right)\end{pmatrix}-\\&\varphi\cdot div\left(\Psi'(\left|\nabla z\right|^2)\cdot\nabla z\right)=0.\end{aligned} \tag{2.3.10}$$

It looks a little bit more sophisticated than the one in (2.2.12), but gives much better results. When discretizing (2.3.10) we obtain a non-linear system of equations due to the functions $\xi_z(u,v,z(u,v))$ and $\zeta_z(u,v,z(u,v))$. In that case we can refer to the *time-marching* numerical scheme, which considers this equation as a steady–state of the following evolution:

$$\begin{aligned}&\frac{\partial z}{\partial t}=\Psi'\begin{pmatrix}\Theta\cdot\left|I_1(\xi,\zeta)-I_2(u,v)\right|^2+\\(1-\Theta)\cdot\left|I_{1u}(\xi,\zeta)-I_{2u}(u,v)\right|^2+\\(1-\Theta)\cdot\left|I_{1v}(\xi,\zeta)-I_{2v}(u,v)\right|^2\end{pmatrix}\cdot\\&\begin{pmatrix}\Theta\cdot\left(I_1(\xi,\zeta)-I_2(u,v)\right)\cdot\left(I_{1u}(\xi,\zeta)\xi_z+I_{1v}(\xi,\zeta)\zeta_z\right)+\\(1-\Theta)\cdot\left(I_{1u}(\xi,\zeta)-I_{2u}(u,v)\right)\cdot\left(I_{1uu}(\xi,\zeta)\xi_z+I_{1uv}(\xi,\zeta)\zeta_z\right)+\\(1-\Theta)\cdot\left(I_{1v}(\xi,\zeta)-I_{2v}(u,v)\right)\cdot\left(I_{1vu}(\xi,\zeta)\xi_z+I_{1vv}(\xi,\zeta)\zeta_z\right)\end{pmatrix}-\\&\varphi\cdot div\left(\Psi'(\left|\nabla z\right|^2)\cdot\nabla z\right)\end{aligned} \tag{2.3.11}$$

It means, that there is some moment of time we will achieve the steady state, when function $z(u,v)$ does not change in time: $\dfrac{\partial z}{\partial t}\xrightarrow[t\to\infty]{}0$. This is an iterative numerical scheme, and as usual it takes a lot of time to gain the steady state. In general the equation that should be solved is pretty big and it takes a lot of computations. When we need real-time performance, we should use some more sophisticated methods, like multigrid [Bra77], or problem linearization, which we will consider in the next chapters.

In case when the object is situated too close to the observer, and thus the displacement of this object is very large, the variational method can stuck in local minima instead of converging to the global one. In that case we may not achieve the

desired solution. To solve this problem we use a *coarse-to-fine level* strategy [BAK91]. We scale down the original stereo image to some coarse level and apply our method to this image. We have chosen such a down-scale level that our problem had unique solution close to the global minimum. This solution is used as the initial data for the next coarse level, which is closer to the fine one. We continue in such a manner until we finally reach the original scale. In the section 4.7 we will consider this process in detail.

2.4 Summary

In this chapter we have presented the general structure of variational methods for depth-map reconstruction, we have explained how to design energy functionals as well as the meaning of its parts: the data term and the smoothness term; how to derive the Euler-Lagrange equation from the energy functional. We have also briefly discussed constancy assumptions that help us construct more reliable data terms, penalizing functions which downweight outliers and lead to a non-linear diffusion process for smoothing the depth-map, but preserve edges and the problem of numerical solution. Moreover, the coarse-to-fine level strategy was described to avoid that the method gets stuck in local minima when we deal with short distant objects.

In the next chapters we will refer to these foundations and consider these principles and approaches in more detail.

Chapter 3

Matching Process

Nearly everyone has used a camera and is familiar with its basic functionality: you indicate your desire to record an image of the world (usually by pressing a button), and the image is recorded onto a piece of film. One of the simplest devices for taking photographs in real world is called the *pinhole camera*. The main principles of the pinhole camera are described and its *projection matrix* which is derived in [PH04].

In this chapter we will consider the modeling of data term and smoothness terms more closely. Constancy assumptions are the core of any data term. Constructing and combining together constancy assumptions requires prior knowledge about the imaging device, which describes image quality, about the scene illumination, and object surfaces which describe incident light reflection. We will discuss in detail the advantages and the shortcomings of different constancy assumptions for the data term, which are frequently used in the literature.

The design of the smoothness term is closely related to the type of the filling-in effect, which in its turn related to the diffusion process. Therefore, the different smoothness constraints are classified in accordance with the induced diffusion process. To show the connection between regularization methods and diffusion filters, we should discuss in this chapter different types of diffusivities and corresponding types of diffusion.

The control of the matching process gives us a number of advantages like automatic controlling the quality of computations, quick error recovery, opportunity to speed up the solver, and others. We will discuss these controlling techniques as well in this chapter.

3.1 Correspondence

We assume that the imaging system follows the pinhole model [PH04]. A pinhole camera consists of a light-tight box with a tiny hole at one end (figure 3.1). When the hole is uncovered, light enters this hole and falls on a piece of photographic paper that is affixed to the other end of the box. Although most cameras are substantially more complex than the pinhole camera, it is a convenient point for simulation.

Therefore, the task of the camera simulator is to take a point on the image and generate *rays* along which light is known to contribute to that image location. Because a ray consists of an origin point and a direction vector, this is particularly simple for the pinhole camera model of figure 3.1: it simply uses the near plane for the origin, and the vector from the eye to the near plane as the ray's direction.

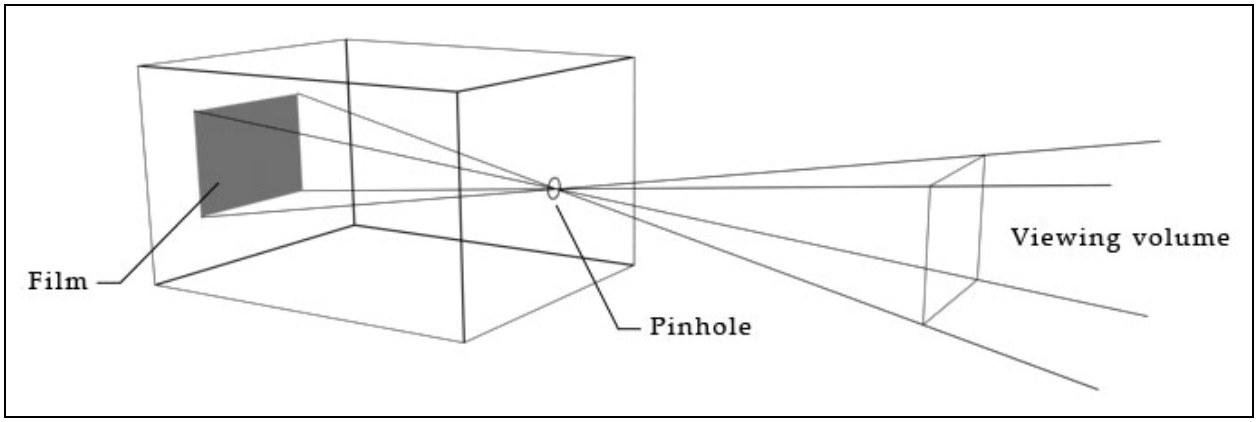

Figure 3.1: A pinhole camera.

Another way to think about the pinhole camera is to place the film in front of the pinhole, at the same distance (figure 3.2) Note that connecting the hole to the film defines exactly the same viewing volume as before. Of course, this is not a practical way to build a real camera, but for simulation purpose it is a convenient abstraction. When the film (or image) plane is in front of the pinhole, the pinhole is frequently referred to as the *eye*.

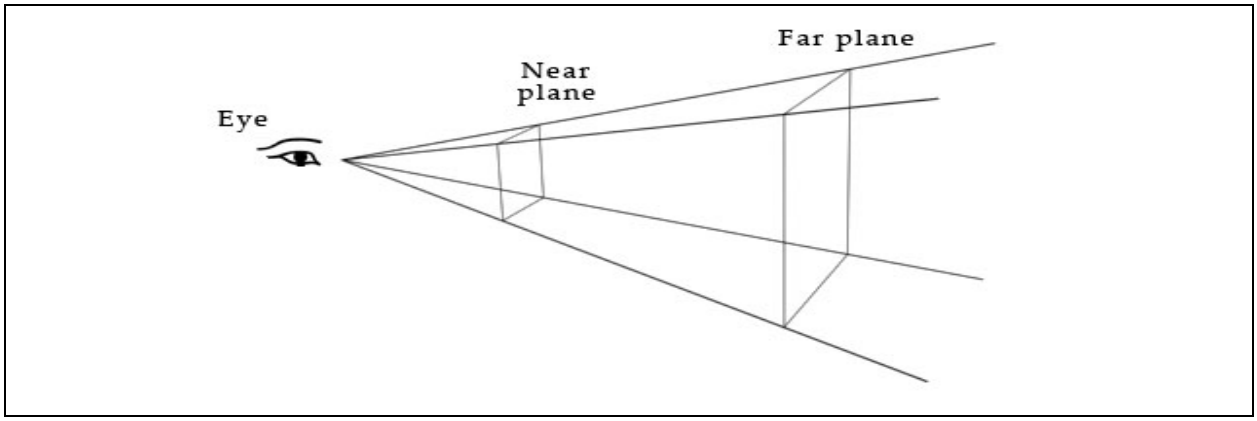

Figure 3.2: When we simulate a pinhole camera, we place the film in front of the hole at the near plane, and the hole is renamed the "eye".

To project a three dimensional point $M(x,y,z)$ from real world into a two dimensional point $m(u,v)$ on a camera's film we need to transform 3D points coordinates (x,y,z) via a *projection matrix* into 2D point coordinates (u,v). The projection matrix describes a pinhole camera, its position and direction and has dimensions 3×4. There are three coordinate systems involved – camera, image and world [HJ94].

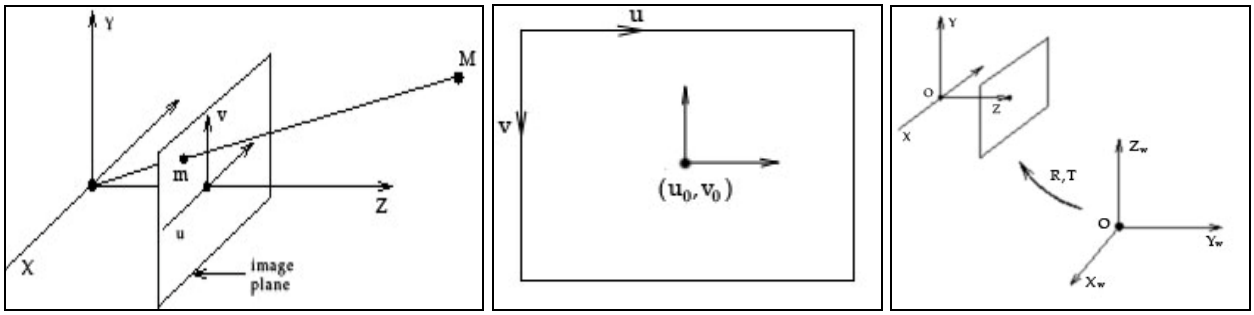

Figure 3.3: Coordinate system's transformation: **Left:** Perspective projection of a 3D point M onto image plane; **Centre:** The image coordinate system transformation; **Right:** Rotation and translation projection from world coordinates onto object coordinates. ([WMW07])

Camera. This coordinate system transformation constitutes the perspective projection (figure 3.3 left). This projection can be expressed as a linear mapping between homogeneous coordinates:

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}. \tag{3.1.1}$$

Image. The image coordinate system transformation gives us the intrinsic camera parameter description matrix J. This matrix provides the transformation between an image point and a ray in Euclidean 3D space. Matrix J is a 3×3 upper triangular matrix, called camera calibration matrix:

$$J = \begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.1.2}$$

and its inverse matrix J^{-1} is:

$$J^{-1} = \begin{pmatrix} \frac{1}{\alpha_u} & 0 & -\frac{u_0}{\alpha_u} \\ 0 & \frac{1}{\alpha_v} & -\frac{v_0}{\alpha_v} \\ 0 & 0 & 1 \end{pmatrix}, \tag{3.1.3}$$

where α_u, α_v are horizontal and vertical pixel sizes in [pixels/length] correspondingly and (u_0, v_0) is the *principal point*, which is the point where the optic axis intersects the image plane [Rob95].

Once matrix J is known the camera is termed *calibrated*. A calibrated camera is a direct sensor, able to measure the direction of rays like a 2D protractor.

World. The world coordinate system transformation constitutes the extrinsic camera's parameters description matrix D. This matrix provides the Euclidean transformation between the camera and world coordinates:

$$D = \begin{pmatrix} R_{1,1} & R_{1,2} & R_{1,3} & t_x \\ R_{2,1} & R_{2,2} & R_{2,3} & t_y \\ R_{3,1} & R_{3,2} & R_{3,3} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{3.1.4}$$

where $R_{i,j}$ are components of a rotation matrix R and t_x, t_y, t_z are coordinates of the camera in Cartesian coordinate system. We consider cameras that are not rotated, i.e. $\begin{cases} R_{i,j} = 0; & i \neq j \\ R_{i,j} = 1; & i = j \end{cases}$. Let us rewrite the matrix D with respect to this assumption:

$$D = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.1.5}$$

and its inverse matrix:

$$D^{-1} = \begin{pmatrix} 1 & 0 & 0 & -t_x \\ 0 & 1 & 0 & -t_y \\ 0 & 0 & 1 & -t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{3.1.6}$$

Now we can construct the projection matrix of our pinhole camera, which will consist of intrinsic camera parameters matrix J and extrinsic camera parameters D:

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \begin{bmatrix} J 0^3 \end{bmatrix} D \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}. \tag{3.1.7}$$

To find pixels coordinates u_i, v_i of the image produced by camera i which correspond to pixel coordinates u_j, v_j of image produced by camera j, we have to write down the correspondence equation:

$$\begin{pmatrix} su_j \\ sv_j \\ s \end{pmatrix} = \left[J_j 0^3\right] D_j D_i^{-1} \begin{bmatrix} zJ_i^{-1} \begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix} \\ 1 \end{bmatrix}, \tag{3.1.8}$$

where s is a scaling parameter and z is the depth-map $z(u,v)$ For more details see [RD96].

Let us substitute (3.1.2), (3.1.3) and (3.1.5), (3.1.6) into (3.1.8) and derive two formulas for coordinate's expression (2.2.2):

$$\begin{pmatrix} su_j \\ sv_j \\ s \end{pmatrix} = \begin{pmatrix} \alpha_u^j & 0 & u_0^j & 0 \\ 0 & \alpha_v^j & v_0^j & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & t_x^j \\ 0 & 1 & 0 & t_y^j \\ 0 & 0 & 1 & t_z^j \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & -t_x^i \\ 0 & 1 & 0 & -t_y^i \\ 0 & 0 & 1 & -t_z^i \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{bmatrix} z \begin{pmatrix} \frac{1}{\alpha_u^i} & 0 & -\frac{u_0^i}{\alpha_u^i} \\ 0 & \frac{1}{\alpha_v^i} & -\frac{v_0^i}{\alpha_v^i} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix} \\ 1 \end{bmatrix}$$

$$= \begin{pmatrix} \alpha_u^j & 0 & u_0^j & 0 \\ 0 & \alpha_v^j & v_0^j & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & t_x^j \\ 0 & 1 & 0 & t_y^j \\ 0 & 0 & 1 & t_z^j \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & -t_x^i \\ 0 & 1 & 0 & -t_y^i \\ 0 & 0 & 1 & -t_z^i \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} z\frac{u_i - u_0^i}{\alpha_u^i} \\ z\frac{v_i - v_0^i}{\alpha_v^i} \\ z \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} \alpha_u^j & 0 & u_0^j & 0 \\ 0 & \alpha_v^j & v_0^j & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & t_x^j \\ 0 & 1 & 0 & t_y^j \\ 0 & 0 & 1 & t_z^j \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} z\frac{u_i - u_0^i}{\alpha_u^i} - t_x^i \\ z\frac{v_i - v_0^i}{\alpha_v^i} - t_y^i \\ z - t_z^i \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} \alpha_u^j & 0 & u_0^j & 0 \\ 0 & \alpha_v^j & v_0^j & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} z\frac{u_i - u_0^i}{\alpha_u^i} + t_x^j - t_x^i \\ z\frac{v_i - v_0^i}{\alpha_v^i} + t_y^j - t_y^i \\ z + t_z^j - t_z^i \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} z\alpha_u^j \dfrac{u_i - u_0^i}{\alpha_u^i} + \alpha_u^j \Delta t_x + z u_0^j + u_0^j \Delta t_z \\ z\alpha_v^j \dfrac{v_i - v_0^i}{\alpha_v^i} + \alpha_v^j \Delta t_y + z v_0^j + v_0^j \Delta t_z \\ z + \Delta t_z \end{pmatrix}.$$

$$\begin{cases} u_j = \dfrac{z\alpha_u^j \dfrac{u_i - u_0^i}{\alpha_u^i} + \alpha_u^j \Delta t_x}{z + \Delta t_z} + u_0^j; \\ v_j = \dfrac{z\alpha_v^j \dfrac{v_i - v_0^i}{\alpha_v^i} + \alpha_v^j \Delta t_y}{z + \Delta t_z} + v_0^j. \end{cases} \tag{3.1.9}$$

We assume that pixel dimensions of any used camera is equal, i.e. $\alpha_u^i = \alpha_u^j$ and $\alpha_v^i = \alpha_v^j$, $\forall i, j$. Moreover, we assume that $u_0^i = 0$ and $v_0^i = 0$, $\forall i$. And now let us first consider the case when cameras shifted relative to each other only in x-direction equal to Δt_x. Now we can rewrite (3.1.9) in a much simpler form:

$$\begin{cases} u_j = u_i + \dfrac{\alpha_u \Delta t_x}{z}; \\ v_j = v_i. \end{cases} \tag{3.1.10}$$

3.2 Grey-value constancy assumption

Constancy assumptions are the core of any data term. In the literature we can find a lot of different constancy assumption developed for one or another particular condition [HS81], [TP84], [BBPW04]. Let us describe them briefly. The *constancy assumption on image brightness* or grey-value constancy assumption is an assumption that the brightness of all the objects in scene is the same from any angle of view. This constancy assumption is the most popular one, since it was developed for any motion type and gives us a big amount of valuable data for small displacements. The major lack of this approach is a big sensitivity to illumination changes. The *constancy assumption on image derivatives* or gradient constancy assumption considers not a picture's intensity in a pixel, but its derivative. Such an approach gives us very good results on illumination changes and on translational, divergent and slow rotational motion type. Another constancy assumption, based on higher order derivatives is called *constancy of the Hessian*. Both these derivatives – driven methods have the same disadvantage – they are

very inaccurate for fast rotational motion type. To eliminate this shortcoming, motion invariant constancy assumptions were developed. The *constancy of the gradient norm* or gradient magnitude constancy assumption is based on idea of creation motion invariant image features from "oriented" derivatives instead of imposing constancy on the (spatial) brightness gradient and therewith on its orientation. Correspondingly we can find rotation invariant features in Hessian – its trace and determinant: *constancy of the trace / determinant of the Hessian*. The constancy of the trace of the Hessian gives us the Laplacian constancy assumption, and the constancy of the determinant of the Hessian gives us the Hessian determinant constancy assumption. When working with colour images, we have to take in account the entire colour channels of image and the *RGB colour brightness* constancy assumption was developed.

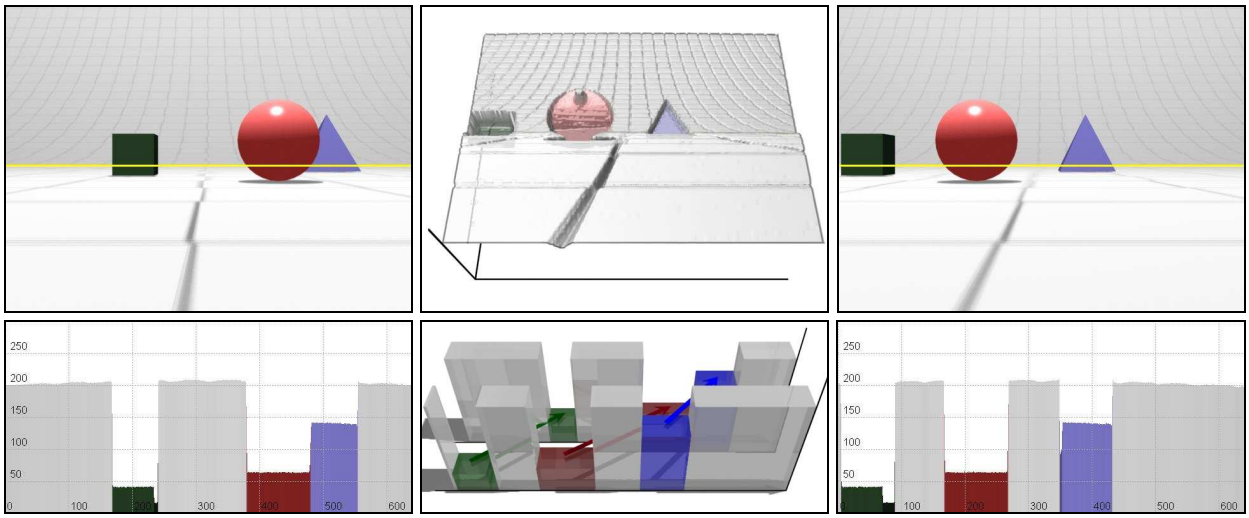

Figure 3.4: "Geometric primitives" scene: **Top Left:** Picture achieved by the left camera; **Top Centre:** Picture achieved by the right camera and represented as a 3D surface; **Top Right:** Picture achieved by the right camera; **Bottom Left**: Diagram of the light intensity of the 250-th line of the top left image; **Bottom Centre**: Correlation of the intensity diagrams; **Bottom Right**: Diagram of the light intensity of the 250-th line of the top right image.

The grey-value constancy assumption will be the base assumption in our model. All the pictures of the stereo image are shot at the same moment of time; it means that we will not meet the problem of illumination changes. Let us consider the figure 3.4, where we can see the left and the right pictures of the stereo-image. As we know the photo-picture could be represented as a function, which takes as arguments the position of a point (u,v) and returns the light intensity: $Intensity = I(u,v)$. Using such a function, we can build a 3 dimensional surface for each picture (see the top centre illustration of the figure 3.4). Such a representation is very useful: we can make a cut if this surface with the plane (yellow line in figure 3.4), and than plot the diagram of light intensity in this cut (see bottom left and right illustrations of figure 3.4). As usual we deal with discrete pictures, where the light intensity is encoded within [0; 255] interval, where 0 means "no light" and represent black color, and 255 represents white color in grayscale images. As we can see on these diagrams we have blocks with the same light intensity, or brightness. For example, the green block, which represents the cube of our scene, has intensity of 45 in 250-th cut-line on both pictures. The same we can observe with red

block that have the same brightness on both pictures. That was the main idea of the brightness constancy assumption. At the bottom centre illustration we can observe how these two diagrams are correlated.

Let us, taking into account (3.1.10), formulate mathematically this assumption as follows:

$$I_1(u,v) - I_2(u + \frac{\alpha_u \Delta t_x}{z}, v) = 0\,. \tag{3.2.1}$$

Unfortunately, the constraint on $\frac{\alpha_u \Delta t_x}{z}$ is rather inconvenient since it is *nonlinear* and *implicit*. When we construct a data term, we should overcome these problems. This is done via two different techniques: first order Taylor approximation [AK02] and linear interpolation method. Both of them are described in detail in section 4.4 as well as their advantages and shortcomings.

Using the first order Taylor approximation we substitute the term $I_2(u + \frac{\alpha_u \Delta t_x}{z}, v)$ with $I_2(u,v) + \frac{\alpha_u \Delta t_x}{z} I_{2u}(u,v)$. Thus we have:

$$I_1(u,v) - I_2(u,v) - \frac{\alpha_u \Delta t_x}{z} I_{2u}(u,v) = 0\,. \tag{3.2.2}$$

The data term built on this assumption:

$$Data\_term_{brightness} = \left(I_1(u,v) - I_2(u,v) - \frac{\alpha_u \Delta t_x}{z} I_{2u}(u,v) \right)^2. \tag{3.2.3}$$

3.3 Gradient constancy assumption

When we have no changes in illumination, the grey-value constancy assumption is very suitable. But it still depends on the assumption that all the objects in our scene have lambertian surfaces. But in real life we have to use more suitable illumination model. On the figure 3.5 we observe the case when brightness constancy assumption will not bring any success.

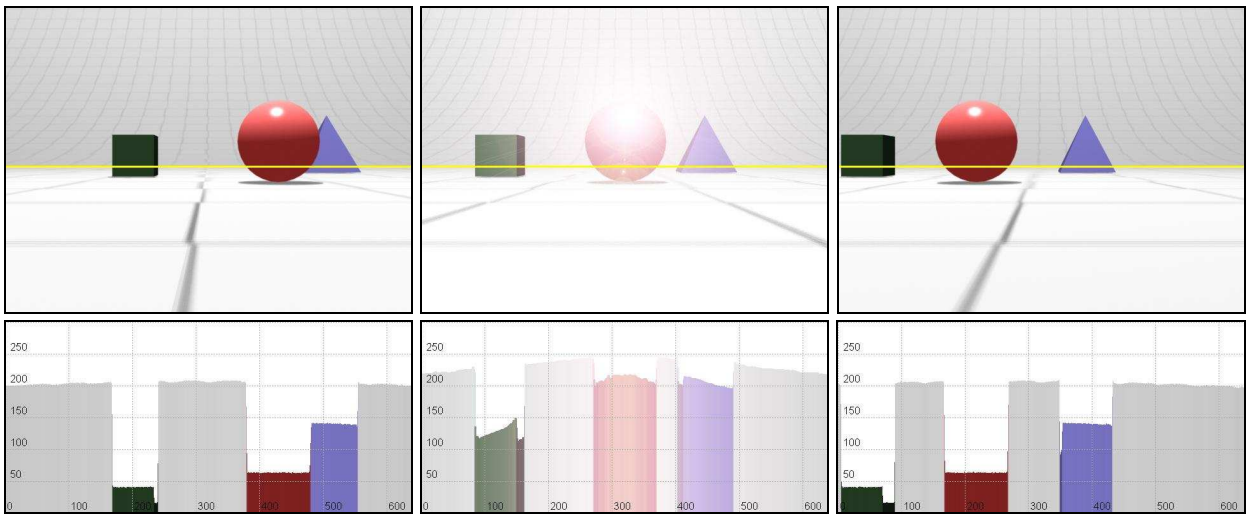

Figure 3.5: "Geometric primitives" scene: **Top Row:** Pictures achieved by the left, centre and right cameras correspondingly; **Bottom Row**: Diagram of the light intensity of the 250-th line of the images from top row correspondingly.

Let us consider the Phong illumination model, which was suggested by Bui Tuong Phong in 1973 [PH04]. According to his model, the reflection from a surface is divided into three subcomponents: *specular* reflection, *diffuse* reflection, and *ambient* reflection (see figure 3.6).

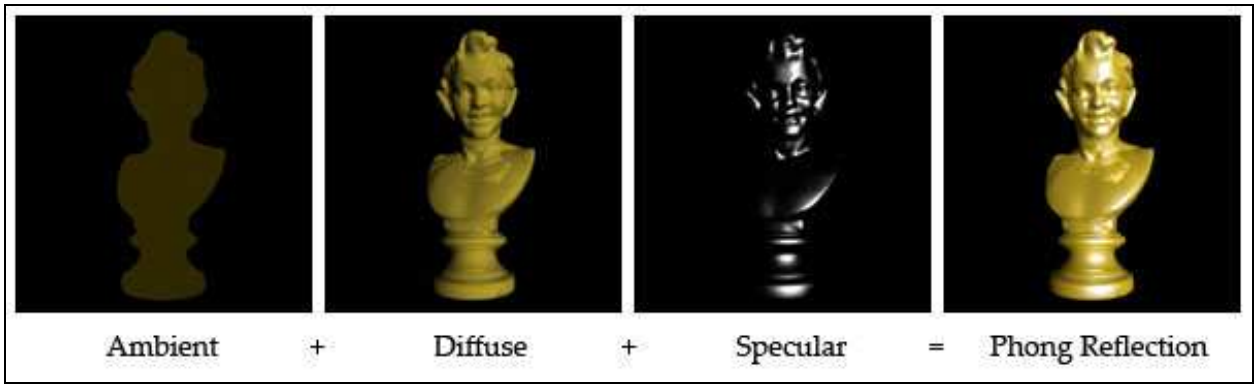

Figure 3.6: Graphical representation of Phong lightning equation.

Mathematically this equation can be written as follows:

$$Intensity = I_{ambient} + (\bar{l} \cdot \bar{n}) I_{Diffuse} + (\bar{r} \cdot \bar{v})^{\alpha} I_{Specular}, \qquad (3.3.1)$$

where $\bar{l}$ is a vector, directed from a point of the surface to the light source and $\bar{r}$ its reflected vector, $\bar{n}$ is a normal vector to the point of the surface and $\bar{v}$ is a vector directed from the point of surface to the camera (figure 3.7).

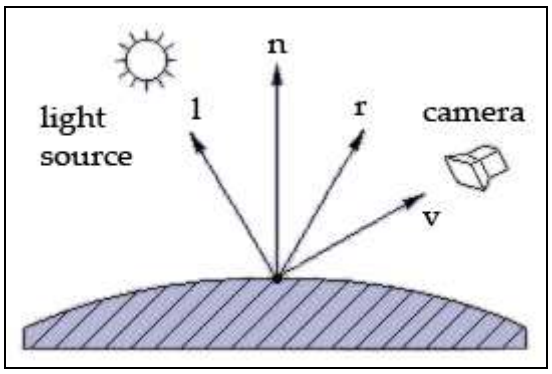

Figure 3.7: Vectors used in the Phong lighting equation of a surface point.

As we can see the specular reflection term is depended on the camera's position. That's why the centre camera on the figure 3.5 is blinded by the spot of the reflecting light, and the picture became too bright. Now the intensity of green block at the bottom left diagram of figure 3.5 (equal to 45) does not equal to the same green block at the bottom centre diagram of figure 3.5 (equal to 150). Thus these diagrams became totally incomparable.

To enhance our results we will add one more constancy assumption to our data term: the illumination invariant gradient constancy assumption. At the top row of the figure 3.8 we observe the result of fuzzy edge detector, applied to pictures of our stereo image. The fuzzy edge detector shows scaled to [0; 255] gradient magnitude in each pixel and represents derivatives of the image in corresponding diagrams. The main point of this idea is that under changing of the illumination conditions the object's edges on pictures are preserved.

At the bottom row of figure 3.8 we see the scaled gradient magnitude that represents the derivatives of the 250-th line of original pictures. They are pretty big at the object's boundaries, and almost didn't suffer from the blinding of the centre camera – we can easily correlate them. The gradient constancy assumption, built on this information helps us greatly in combination with brightness constancy assumption.

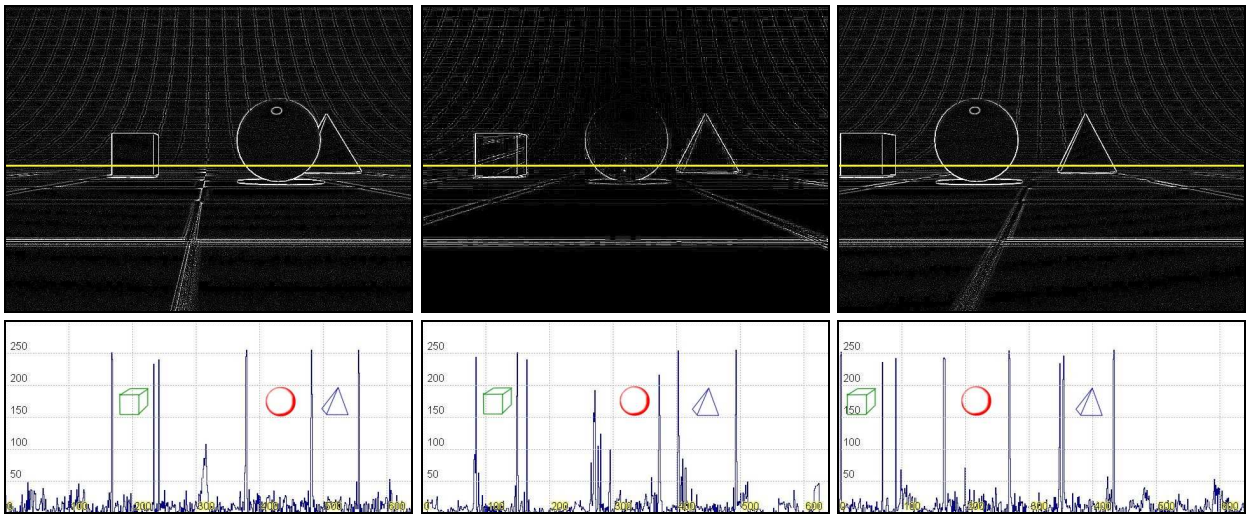

Figure 3.8: "Geometric primitives" scene: **Top Row:** Fuzzy edge detector applied to the pictures achieved by the left, centre and right cameras correspondingly; **Bottom Row**: derivatives of the light intensity of the 250-th line of pictures achieved by the left, the centre and the right cameras correspondingly.

The gradient assumption with respect to (3.1.10) obtains the following form:

$$\nabla I_1(u,v) - \nabla I_2(u + \frac{\alpha_u \Delta t_x}{z}, v) = 0\,. \tag{3.3.2}$$

Since the spatial gradient is a vector with two components, we obtain two constraints this time. They are given by

$$\begin{cases} I_{1u}(u,v) - I_{2u}(u + \frac{\alpha_u \Delta t_x}{z}, v) = 0; \\ I_{1v}(u,v) - I_{2v}(u + \frac{\alpha_u \Delta t_x}{z}, v) = 0. \end{cases} \tag{3.3.3}$$

Using the same first-order Taylor approximation method like in brightness constancy constraint to get rid of implicitly, we achieve:

$$\begin{cases} I_{1u}(u,v) - I_{2u}(u,v) - \frac{\alpha_u \Delta t_x}{z} I_{2uu}(u,v) = 0; \\ I_{1v}(u,v) - I_{2v}(u,v) - \frac{\alpha_u \Delta t_x}{z} I_{2vu}(u,v) = 0. \end{cases} \tag{3.3.4}$$

Squaring and adding them together produces the needed part of the data term:

$$\begin{aligned} Data\_term_{gradient} = &\left(I_{1u}(u,v) - I_{2u}(u,v) - \frac{\alpha_u \Delta t_x}{z} I_{2uu}(u,v) \right)^2 + \\ &\left(I_{1v}(u,v) - I_{2v}(u,v) - \frac{\alpha_u \Delta t_x}{z} I_{2vu}(u,v) \right)^2. \end{aligned} \tag{3.3.5}$$

3.4 Diffusion process

Diffusion is a term from physics: *diffusion* is the net action of matter (particles or molecules), heat, momentum, or light whose end is to minimize a concentration gradient. A diffusion process is characterized by two standings. The first one is that the diffusion process always preserves the mass of matter [Wei98]. And the second is that the diffusion process equilibrates differences of matter concentration. These standings are easy to describe with two formulas:

1. *Fick's law* describes the equilibration of concentration differences: $j = -g \cdot \nabla u$.
   Concentration gradient ∇u creates flux j, and g is a diffusion tensor.
2. *Continuity equation* describes conservation of mass: $\partial_t u = div(g \cdot \nabla u)$.

In such a way, we have derived the diffusion equation:

$$\partial_t u = div(g \cdot \nabla u). \tag{3.4.1}$$

In a case of linear diffusion (when we do not consider local structure of matter, i.e. the diffusion tensor is equivalent to identity) we obtain the simplest diffusion process: $\partial_t u = \Delta u$.

To understand how it works, imagine a cap of milk. You can shake this cap somehow to create waves on the milk surface. But as soon as you put this cup on a table, the milk will start settling down and in some seconds you will have no waves anymore:

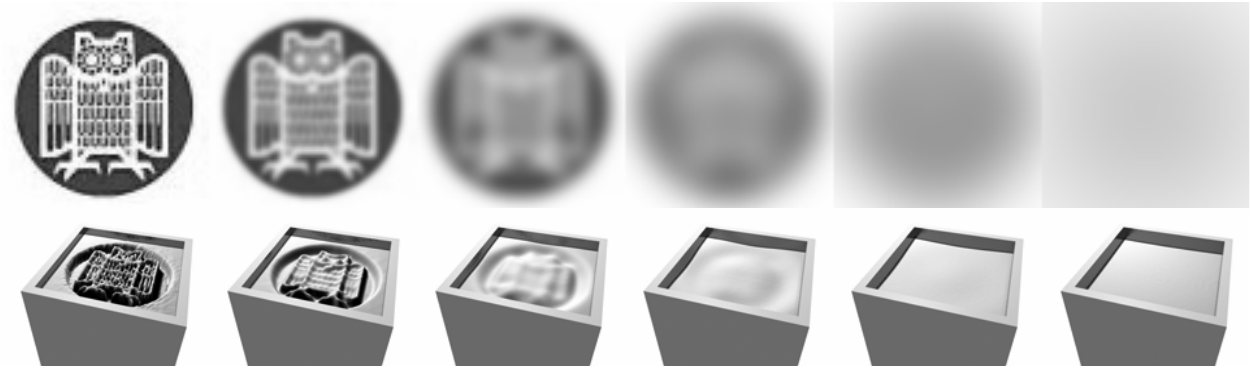

Figure 3.9: UdS logo. Linear diffusion example: **From left to right:** original state and linear diffusion process with time.

You still have the same amount of milk (mass preservation), but the difference between any two drops of milk at the surface is minimal – you have flat surface (equilibrium of concentration differences). In image processing, diffusion is very useful for image enhancement. For example – for denoising an image. When we want to get rid of high frequencies (noise or some small details) but preserve edges from blurring. Unfortunatly it is almost impossible to do this using simple linear diffusion.

So, let us consider nonlinear diffusion. This process avoids delocalization and blurring of edges. It is described with the following equation:

$$\partial_t u = div(g(|\nabla u|) \cdot \nabla u) \quad on \quad \Omega \times (0,\infty) \tag{3.4.2}$$

with stable initial and boundary conditions:

$$\begin{aligned} &u(x,0) = f(x) \quad on \quad \Omega; \\ &\partial_n u = 0 \quad on \quad \partial\Omega \times (0,\infty). \end{aligned} \tag{3.4.3}$$

Function $g()$ takes as an argument a *fuzzy edge detector* $|\nabla u|$ and should be chosen as a decreasing nonnegative function. It means that on the edges, where the image derivatives are high, low values of the function $g()$ will inhibit diffusion. Now let us

suppose that we want to smooth the UdS, preserving its edges with help of diffusion process. And let us, with help of the following illustration, compare the best result of eliminating noise with linear diffusion process and, nonlinear diffusion process:

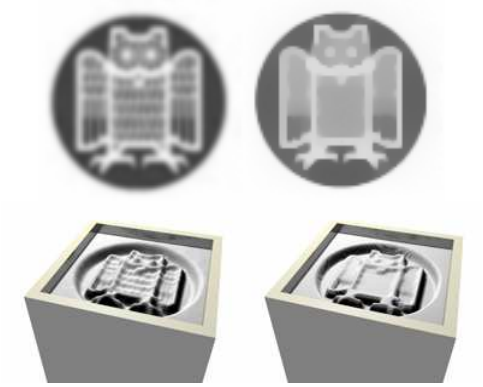

Figure 3.10: UdS logo: **Left:** Linear diffusion. **Right:** Nonlinear diffusion.

We can clearly see that in the right column, our owl has no plumage anymore but the edges are very good preserved [ALM92].

Singular diffusion filters lead to a piecewise constant images as it was shown in [WWS05]. As function $g()$, diffusivity, we use singular diffusivities. As a prototype for a class of singular diffusivities we consider the family:

$$g(|\nabla u|) = \frac{1}{|\nabla u|^p}; \quad p \geq 1. \tag{3.4.4}$$

In this chapter we will consider two singular diffusivities. *Total variation* (TV) diffusivity, that has interesting properties such as finite extinction time and shape-preserving qualities – the case when $p = 1$: $g(|\nabla u|) = \frac{1}{|\nabla u|}$. For $p > 1$ the diffusion not only preserves edges but even enhances them. *Balanced forward-backward* (BFB) diffusivity – the case when $p = 2$: $g(|\nabla u|) = \frac{1}{|\nabla u|^2}$. The most problems with singular diffusivities are that it is possible to have very small values of $|\nabla u|$ and in such cases our function $g()$ becomes unbounded. That implies numerical instability, and failure of solver. As a result, iterative numerical schemes may reveal slow convergence, and in general numerical errors can be amplified. In order to eliminate all these problems, it is common to regularize the diffusivity function by replacing it by the bounded diffusivity:

$$g(|\nabla u|) = \frac{1}{\left(|\nabla u|^2 + \varepsilon^2\right)^{p/2}}; \quad p \geq 1. \tag{3.4.5}$$

As summary for the diffusion process description, let us consider the diffusion process on color images in figure 3.11

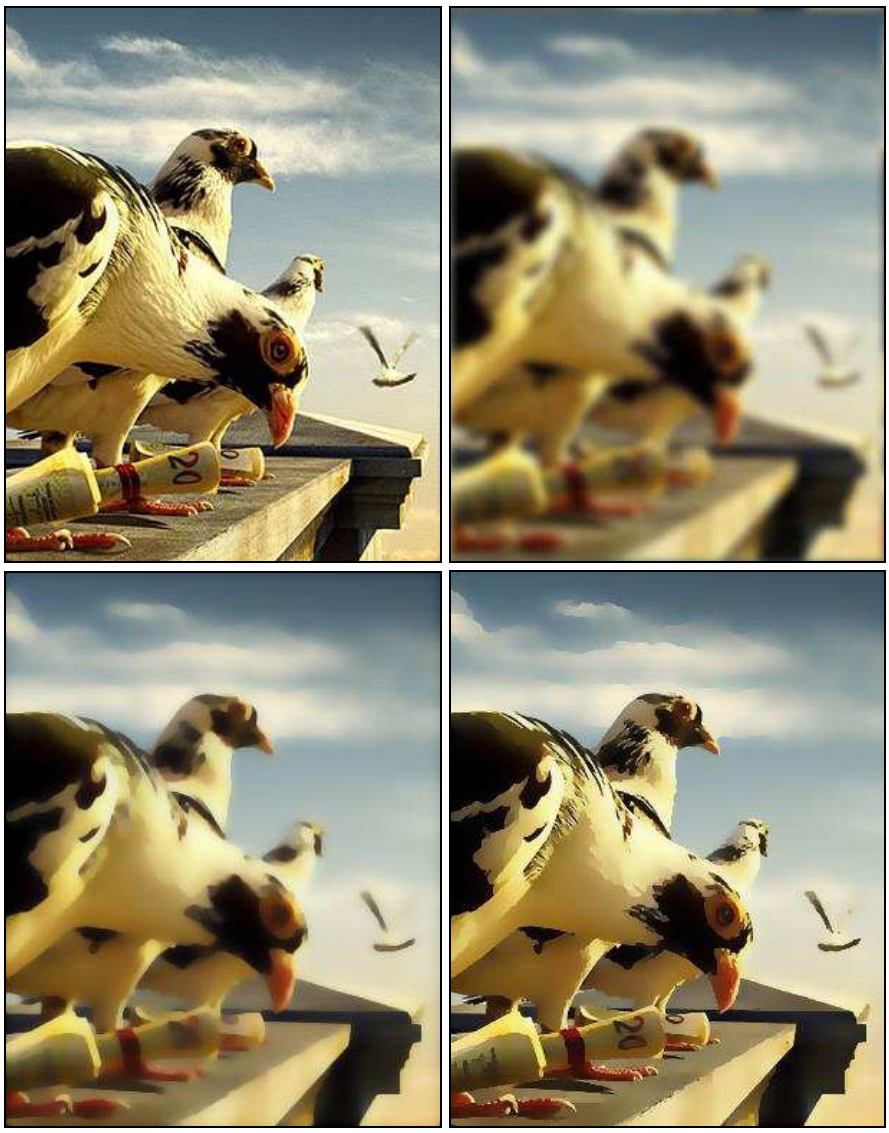

Figure 3.11: "Carrier-pigeon" scene: **Top Left:** Original image. **Top Right:** Linear diffusion with periodic boundary conditions. **Bottom Left**: Total variation diffusion; **Bottom Right**: Balanced forward – backward diffusion. (The original photo is taken from [WF07] and processed by the thesis author's freeware application [WPX07]).

Figure 3.11 demonstrates linear, total variation and balanced forward–backward diffusion. The elimination of noise with edge preservation (and even enhancement in case of BFB diffusion) is very good seen on Figure 3.12. Gradient $|\nabla u|$ is very sensitive to high frequencies, and all the pictures show only contours of the objects in scene. These images are the results of fuzzy edge detector processor with original image as initial data. At the top left picture we can observe not only contours but also a lot of high frequencies where spatial derivatives are high. At the top right image we have get rid of high frequencies, but all the edges are extremely blurred – we have lost a lot important information about objects geometry. At the bottom row we see no high frequencies, and all the object edges are good preserved. In case of balanced forward-backward diffusion, we can even notice, that the fuzzy edge detector gave us more bright boundaries than in case of total variation diffusion, and even in case of original image.

Figure 3.12: "Carrier-pigeon" scene under the fuzzy edge detector: **Top Left:** Original image. **Top Right:** Linear diffusion with periodic boundary conditions. **Bottom Left**: Total variation diffusion; **Bottom Right**: Balanced forward – backward diffusion. (The original photo is original taken from [WF07] and processed the thesis author's freeware application [WPX07]).

3.5 Preserving discontinuities

Let us consider formulae (2.3.5) and (2.3.9) more close. Comparing them, we can derive a connection between the smoothness term and the smoothness term of the Euler-Lagrange equation:

$$Smoothness\_term(z_u, z_v) = \varphi \cdot \Psi(|\nabla z|^2), \tag{3.5.1}$$

$$Smoothness\_term_{Euler-Lagrange} = 2\varphi \cdot div\left(\Psi'(|\nabla z|^2) \cdot \nabla z\right). \tag{3.5.2}$$

The diffusion process, that will take place during the solution, and that is introduced by the smoothness term, is determined by the penalizing function Ψ. In

general function Ψ takes two parameters: $\Psi(\nabla I, \nabla z)$, and its derivative $\Psi'(\nabla I, \nabla z)$ is called *diffusivity*. Let us discuss briefly describe different diffusivities, described in the literature [Bru06], [DKA95], [GR92] and corresponding diffusion processes.

- *Homogeneous diffusion* is the simplest form of diffusion, based on linear diffusion. As a consequence, the resulting solution – depth-map will be homogeneously blurred and semantically important edges may be dislocated. As a penalizing function is used the Tikhonov regularizer: $\Psi(\nabla I, \nabla z) = |\nabla z|^2$, which corresponds to the homogeneous diffusion: $u_t = \Delta u$.

- *Linear isotropic diffusion* is a more advanced diffusion process, which is image – driven. It should be constructed in such a way, to respect discontinuities in initial stereo image by reducing smoothing at image boundaries. Let function $g(s)$ is a decreasing smooth function, then the penalizing function will be written as: $\Psi(\nabla I, \nabla z) = g(|\nabla I|^2) \cdot |\nabla z|^2$. Such a penalizing function leads us to the linear isotropic diffusion: $u_t = div\left(g(|\nabla I|^2)\nabla z\right)$. The most disadvantages appear when we deal with high textured objects, which have more image boundaries. This gives over segmented depth-maps.

- *Linear anisotropic diffusion* is also image-driven diffusion, but in contrast to the linear isotropic diffusion, can provide smoothing along image edges. It is done with help of a regularized projection matrix $D(\nabla I)$ on $\nabla I^{\perp}$, which in contrast to normal gradient is definite as follows: $\nabla I^{\perp} = \begin{pmatrix} I_v \\ -I_u \end{pmatrix}$. The projection matrix $D(\nabla I)$ has form: $D(\nabla I) = \frac{1}{|\nabla I|^2 + 2\lambda^2}(\nabla I^{\perp} \nabla I^{\perp T} + \lambda^2 I)$. The penalizing function $\Psi(\nabla I, \nabla z)$ now has the following look: $\Psi(\nabla I, \nabla z) = \nabla z^T D(\nabla I) \nabla z$ with corresponding diffusion process: $u_t = div\left(D(\nabla I)\nabla z\right)$. To understand how it works, we should realize that matrix $D(\nabla I)$ has two eigenvectors ∇f and $\nabla f^{\perp}$ with eigenvalues $\lambda_1(|\nabla I|) = \frac{\lambda^2}{|\nabla I|^2 + 2\lambda^2}$ and $\lambda_2(|\nabla I|) = \frac{|\nabla I|^2 + \lambda^2}{|\nabla I|^2 + 2\lambda^2}$- The isotropic behaviour at flat image regions, i.e. when $|\nabla I| \to 0$: $\lim_{|\nabla I| \to 0} \lambda_1(|\nabla I|) = \frac{1}{2}$, $\lim_{|\nabla I| \to 0} \lambda_2(|\nabla I|) = \frac{1}{2}$. And the anisotropic behaviour at image edges, i.e. when $|\nabla I| \to \infty$: $\lim_{|\nabla I| \to \infty} \lambda_1(|\nabla I|) = 0$, $\lim_{|\nabla I| \to \infty} \lambda_2(|\nabla I|) = 1$. This approach gives better results than isotropic image-driven one, but still suffers from high-textured regions in images.

- *Nonlinear isotropic diffusion* is very close to the linear isotropic diffusion, but instead of using boundary information from the initial image, it uses feedback from the evolution – still calculating depth-map. So, it is a depth-driven diffusion process, which determines by the following penalizing function: $\Psi(\nabla I, \nabla z) = \Psi(|\nabla z|^2)$. Here function Ψ can be any increasing, differentiable, convex in its argument, and bounded. The corresponding isotropic nonlinear diffusion process: $u_t = div\left(\Psi'(|\nabla z|^2)\nabla z\right)$.

• *Nonlinear anisotropic diffusion* is also exists for depth-driven process. It also has advantage of smoothing along edges of depth-map. It uses penalizing function as a trace of the matrix: $\Psi(\nabla I, \nabla z) = tr\Psi(\nabla z \nabla z^T)$. Such an approach leads to the anisotropic nonlinear diffusion: $u_t = div\big(D(\nabla z)\cdot\nabla z\big)$, where $D(\nabla u) = \Psi'(\nabla z \nabla z^T)$. If v_1, v_2 are the eigenvectors of $\nabla z \nabla z^T$ and μ_1, μ_2 the corresponding eigenvalues, then $D(\nabla u)$ has eigenvectors v_1, v_2 with eigenvalues $\Psi'(\mu_1)$, $\Psi'(\mu_2)$ and this is exactly the desired anisotropy.

It is very difficult to use image-driven diffusion processes, since we deal with stereo-images, which are represented by a number of pictures. In this paper we use the homogeneous diffusion process and the nonlinear isotropic diffusion process. Linear diffusion is the simplest among diffusion processes, and could be applied to the wide variety of variational problems. Substituting the Tichonov regularizer

$$\Psi_{Tichonov}(s^2) = s^2 \tag{3.5.3}$$

into the formulae (3.5.1) and (3.5.2) we will have:

$$Smoothness\_term(z_u, z_v) = \varphi \cdot |\nabla z|^2, \tag{3.5.4}$$

$$Smoothness\_term_{Euler-Lagrange} = 2\varphi \cdot div(\nabla z) = 2\varphi \cdot \Delta z. \tag{3.5.5}$$

Using the nonlinear isotropic diffusion is a great improvement. It preserves and can even enhance edges in depth-map, and thus helps to segment different objects in scene. As penalizing function we will take the Charbonnier regularizer [CABB94]:

$$\Psi_{Charbonnier}(s^2) = 2\lambda^2\sqrt{1+\frac{s^2}{\lambda^2}} - 2\lambda^2 \tag{3.5.6}$$

and the Perona-Malik regularizer

$$\Psi_{Perona-Malik}(s^2) = \lambda^2 \ln(\lambda^2 + s^2) - \lambda^2 \ln(\lambda^2). \tag{3.5.7}$$

Let us first consider the Charbonnier regularizer. To understand how in works we should write down its diffusivity function $\Psi'(s^2)$ and its flux function $\Psi'(s^2)\cdot s$. Its diffusivity function:

$$\Psi'_{Charbonnier}(s^2) = \frac{\lambda}{\sqrt{\lambda^2 + s^2}} \tag{3.5.8}$$

and its flux function:

$$\Phi_{Charbonnier}(s) = \frac{\lambda s}{\sqrt{\lambda^2 + s^2}}. \tag{3.5.9}$$

Now let us consider the plots of these functions:

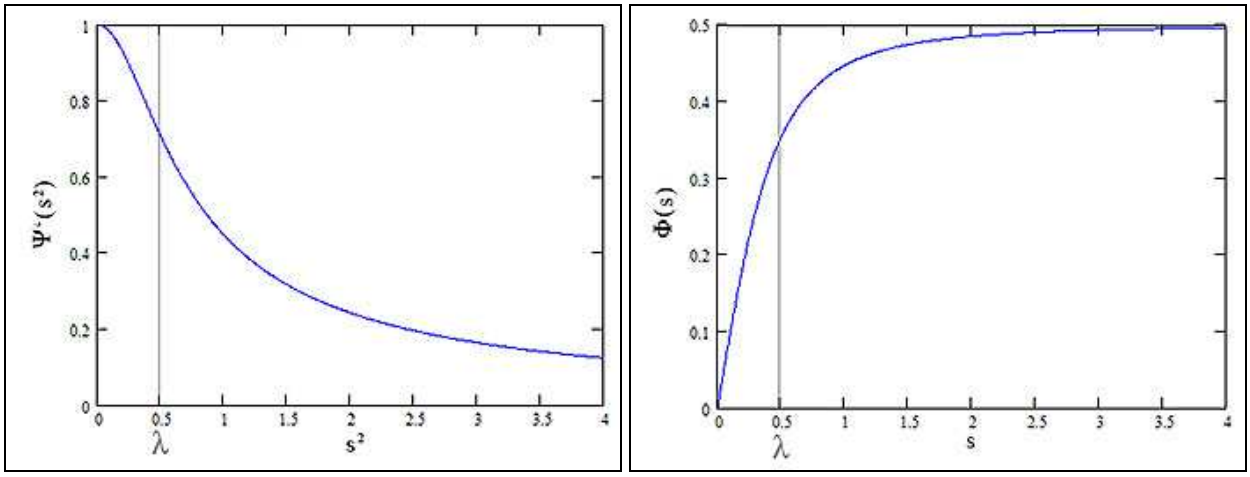

Figure 3.13: Charbonnier regularization: **Left:** Corresponding diffusivity. **Top Right:** Corresponding flux function.

If we think about 1D diffusion process, we have the following equation for the nonlinear isotropic equation:

$$u_t = \left(\Psi'(z_u^2)z_u\right)_u = \left(\Phi(z_u)\right)_u = \Phi'(z_u)z_{uu}\,. \tag{3.5.10}$$

The diffusivity function is a monotonically decreasing function, see figure 3.13. From formula (3.5.10) we can see, that when we have an edge in out image, than its derivative becomes larger, and thus the diffusivity becomes smaller, attenuating diffusion. Also from the figure 3.13 we observe, that the flux function is monotonically increasing, it means that $\Phi'(s) > 0, \quad \forall s \geq 0$. Thus the right part of equation (3.5.10) is always positive and we have always smoothing of forward diffusion.

Now let us consider the Perona-Malik regularizer [PM90] and compare it with the Charbonnier regularizer. The derivative of the Perona-Malik regularizer is:

$$\Psi'_{Perona-Malik}(s^2) = \frac{\lambda^2}{\lambda^2 + s^2} \tag{3.5.11}$$

and its flux function:

$$\Phi_{Perona-Malik}(s) = \frac{\lambda^2 s}{\lambda^2 + s^2}\,. \tag{3.5.12}$$

These functions are plotted at the figure 3.14.

In contrast to the Charbonnier penalizer, the diffusivity function corresponded to the Perona-Malik penalizer falls faster (compare the cross points of the graph and lambda-line). And, what is more important, the flux function now is not monotonic. We observe that $\Phi'(s) > 0$ for $s < \lambda$ and $\Phi'(s) < 0$ for $s > \lambda$. It means that for $s < \lambda$ we have

forward diffusion process and thus smoothing, and for $s > \lambda$ we have backward diffusion process and thus edge-enchasing.

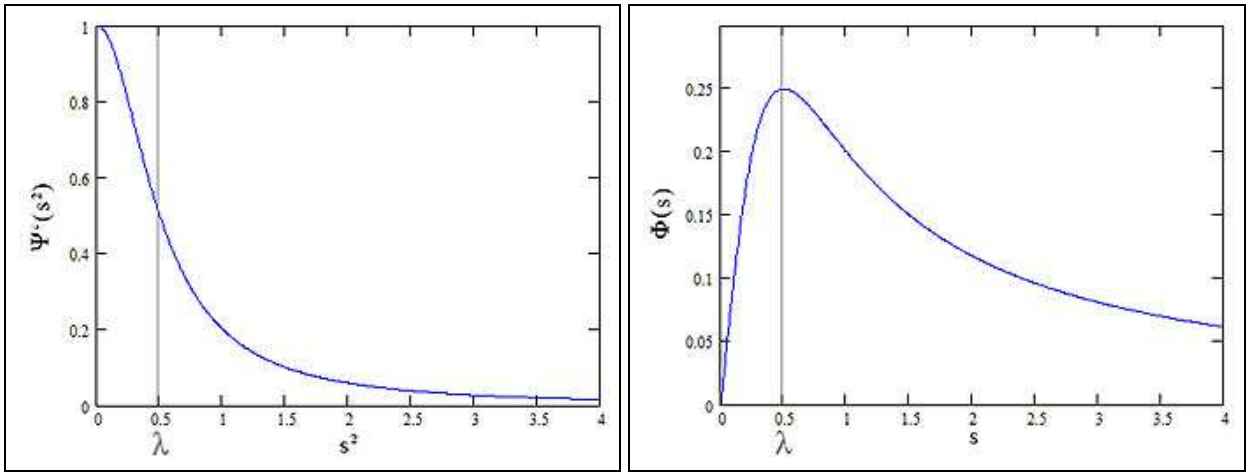

Figure 3.14: Perona-Malik regularization: **Left:** Corresponding diffusivity. **Top Right:** Corresponding flux function.

Discussing formulae (3.5.8) and (3.5.11) we must say that Charbonnier penalizer corresponds to the total variation diffusion, and Perona-Malik penalizer corresponds to the balanced forward–backward diffusion [BWSW03]. These theoretical computations explain the results, illustrated in figure 3.14.

3.6 Controlling the matching process

To control the matching process during the computation is a very important task, since even a small error at the beginning of international process can affect the final result significantly. As usual, scientists before starting the calculations define the setup control parameters (time step, smoothing parameters, etc.), which are kept fixed during the computations, start process and after that, gaining the result, estimate the relative error. Nonlinear variational methods can be very slow since they need time-marching numerical schemes that as usual waste too much time to converge. And such an approach, when we should wait till the end of calculations, makes them even more slow and useless for industry.

Another very important role of the controlling the matching process is follows: if we find a method capable to estimate the speed of method's convergence and if we know how to influence upon it, it appears to be possible us to interact with the iteration process on run.

It is possible to control the convergence of variational method during the computation via estimating the energy functional after each (or after each N) iteration. This gives us a possibility of immediate computational error recognition and recovery. More flexibility gives us the possibility of automatic picking up setup control parameters. This approach, combined with controlling the matching process allows us to change these setup parameters automatically not only on setup and calculation

beginning, but even during the calculation. First of all, it gives the possibility to try different control parameters for problem iteration, and consequently fix up numerical problems on run. Second, it speeds up the convergence of the method greatly.

The energy functional (2.1.4) we can rewrite in the following form:

$$Sum^{(i)} = \sum_{u=0}^{width} \sum_{v=0}^{height} Data\_term(u,v,z^{(i)}) + Smothness\_term(z_u^{(i)}, z_v^{(i)}), \qquad (3.6.1)$$

where subscript $^{(i)}$ denotes the iteration number. During the computations, solving the Euler-Lagrange equation, on each iteration we are finding the function $z^{(i)}(u,v)$. In ideal case function $z^{(\infty)}(u,v)$ is our depth-map, and the functional (2.1.4), if the smoothness term was chosen properly, is minimal. By other words:

$$Sum^{(i)} \xrightarrow[i\to\infty]{} 0. \qquad (3.6.2)$$

Also in ideal case we will have monotonous convergence, which can be written as two mathematical formulas:

$$Sum^{(i)} > Sum^{(i+j)}, \quad \forall i, j \in N, \qquad (3.6.3)$$

$$Sum^{(i)} - Sum^{(i+j)} > Sum^{(i+k)} - Sum^{(i+k+j)}, \quad \forall i, j, k \in N, \qquad (3.6.4)$$

The formula (3.6.3) says us than the row $Sum^{(i)}$ is *strictly monotonically decrescent*: each subsequent member is smaller then preceding. The formula (3.6.4) guaranties us that the row $Sum^{(i)}$ converges to some definite value. We check both criteria on violation and thus have powerful tool of analyzing and controlling the matching process.

The energy functional representation (3.6.1) gives us also the reach material for relaxation methods, which definitely help to speed up convergence. For the further information refer to [Ter86].

Anyway, the representation (3.6.1) has some disadvantages. First of all it includes the smoothness term, which we need only for numerical solution of the Euler-Lagrange equation. The most important parts are the constancy assumptions, since they are the core of any energy functional. With this idea, using the main constancy assumption on image brightness, we can write down another formula for the row $Sum^{(i)}$:

$$Sum^{(i)} = \sum_{u=0}^{width} \sum_{v=0}^{height} \left(I_1(u,v) - I_2(u + \frac{\alpha_u \Delta t_x}{z^{(i)}}, v) \right)^2. \qquad (3.6.5)$$

Now let us discuss the technique of the controlling the matching process more closely. We have two criteria for it, let us call formula (3.6.3) the *main* criteria and formula (3.6.4) the *secondary* criteria. The violation of the main criteria we consider as a calculation mistake. In this case we try to repeat the last iteration with new parameters.

The violation of the secondary criteria we consider as an acceptable mistake. Having in mind the main criteria, we can formulate the concept of the convergence speed:

$$Speed_{conv} = \frac{Sum^{(i)} - Sum^{(i+j)}}{j}, \quad \forall i, j \in N. \tag{3.6.6}$$

In case of the convergence speed becomes smaller than a certain threshold, we may try to play with the parameters again, to find a better vector of minimization. The most interesting question is about the way how to change these parameters. Particularly, when we vary such constants like ρ or λ we receive another problem so the formula (3.6.5) will not give comparable $Sum^{(i)}$, but the point here – is to recover the calculation error in the case of the main criteria violation and to gain the more fast convergence in the case of $Speed_{conv}$ became too small. Note that, using the fine-to-coarse levels strategy or warping technique leads to the whole problem changing and use the solution from the previous problem as initial data for the new one. Approximately the same we observe with the controlling technique.

The criteria for the stopping the iteration process and the parameters varying strategy depend on the according algorithms and the implementation. In this thesis we use a heuristic parameters determination and adaptation block.

3.7 Summary

In this chapter we have discussed important theoretical topics that constitute the basis for stereo problems in image processing. The most important of them are mathematical models of cameras and scene illumination, diffusion processes and gradient filtering theory. Using matrix transformations and mathematical description of camera, we have derived correspondence equation, which binds any pixel of one picture of the stereo image with the pixel of another picture through the real 3D point of the world. Moreover this equation binds the pixel size with the real metric system.

We have discussed the theory of diffusion processes and the theory of illumination to design and build data terms and smoothness terms for our energy functional. Also we have marked what kind of results are gained with using different data and smoothness terms and the ways how to control the solver during computations for fast error recovery and correcting the way the solution process flows.

In the next chapters we will precede more close to practice, build and discuss numerical schemes, build the solver, achieve first results, and than step by step improve them.

Chapter 4

Depth-Map Reconstruction with Two Cameras

In this chapter we will come from the theory to the practice. Four different methods of the depth-map reconstruction are waiting to be considered and their advantages and shortcomings to be discussed. As a powerful tool to anlize the regularization behavior of different smoothness terms, the concept of an *RnB pyramid* is presented. Moreover the techniques, aforementioned in previous chapters, like multi scale technique and the technique that modifies the data term such that it becomes more robust will be described in detail. We will also describe which practical case needs a particular technique or method. Then we come to the numerical schemes – the paragraph, where we will step by step derive and explain the working numerical schemes for linear and non linear Euler – Lagrange equations. All the discussions and conclusions in this chapter are about the two camera case; they are the fundament for multi – view depth-map reconstruction, which will be considered in the next chapter.

4.1 Depth-driven method vs. disparity-driven method

For the reconstruction of the depth-map it is possible to use two different approaches. They are the depth-driven approach and the disparity-driven approach. To explain the difference between these approaches, let us consider the first equation from the system (3.1.10) for two cameras:

$$u_2 = u_1 + \frac{\alpha_u \Delta t_x}{z(u_1, v_1)}, \tag{4.1.1}$$

here the function $z(u_1, v_1)$ is the depth-map and the fraction $\frac{\alpha_u \Delta t_x}{z(u_1, v_1)}$ is the disparity between two pixels from the different cameras, constituted by the projection of one world point. By other words, let us suppose that a world point $M(x, y, z)$ projects to the pixel $m_1(u_1, v_1)$ of the first camera and to the pixel $m_2(u_2, v_2)$ of the second camera. Accordingly to the system of equations (3.1.10) $v_1 = v_2$ (since the cameras are shifted among each other only horizontally) and the value of $\frac{\alpha_u \Delta t_x}{z(u_1, v_1)}$ is the disparity in pixels between u_1 and u_2.

The idea of the disparity-driven approach is to calculate the disparity map or *optic flow* field between the cameras and then calculate from it the depth-map, instead of calculating the depth-map directly. If we denote though $z'(u_1, v_1)$ the disparity map, then the equation (4.1.1) will be rewritten as follows:

$$u_2 = u_1 + z'(u_1, v_1) \tag{4.1.2}$$

and to achieve the depth-map from the calculated disparity map we will us the formula:

$$z(u, v) = \frac{\alpha_u \Delta t_x}{z'(u, v)}. \tag{4.1.3}$$

The disparity driven method has some ponderable advantages. First of all, it leads to the Euler – Lagrange equation which is linear, while the depth driven method leads to the non-linear equation [BCAB95]. Let us consider the energy functionals and the corresponding Euler – Lagrange equations. For the simplicity let us use only the grey-value constancy assumption in the data term. The energy functional for the depth driven approach:

$$E(z) = \iint_{\Omega} \left(I_1(u, v) - I_2\left(u + \frac{\alpha_u \Delta t_x}{z(u, v)}, v\right) \right)^2 + \varphi \cdot \Psi(|\nabla z|^2) du dv \tag{4.1.4}$$

and its Euler – Lagrange equation:

$$-I_{2u}\left(u + \frac{\alpha_u \Delta t_x}{z(u, v)}, v\right) \frac{\alpha_u \Delta t_x}{z^2} \left(I_1(u, v) - I_2\left(u + \frac{\alpha_u \Delta t_x}{z(u, v)}, v\right) \right) + \varphi \cdot div\left(\Psi'(|\nabla z|^2) \cdot \nabla z\right) = 0. \tag{4.1.5}$$

This Euler – Lagrange equation still has implicitness in function I_2 and its derivative, and we can get rid of it, using the techniques of first order Taylor expansion or linear interpolation described in paragraphs 4.2 and 4.3. But we can already see that

due to the term $\frac{\alpha_u \Delta t_x}{z^2}$ it will be no possibility to build a linear numerical scheme. Now let us write the energy functional for the disparity driven approach:

$$E(z') = \iint_{\Omega} \left(I_1(u,v) - I_2(u + z'(u,v), v)\right)^2 + \varphi \cdot \Psi(|\nabla z|^2) du dv \qquad (4.1.6)$$

and its Euler – Lagrange equation:

$$I_{2u}(u + z'(u,v), v)\left(I_1(u,v) - I_2(u + z'(u,v), v)\right) + \varphi \cdot div\left(\Psi'(|\nabla z'|^2) \cdot \nabla z'\right) = 0, \qquad (4.1.7)$$

here we have got rid of the term $\frac{\alpha_u \Delta t_x}{z^2}$ and now it is possible to choose a linear numerical scheme for solving this equation.

Another advantage of the disparity-driven method becomes clear, if we look more carefully at the smoothness terms of the energy functionals. While theirs data terms are different, the corresponding smoothness terms are still almost the same. Pay attention that in case of the depth-driven approach we are smoothing the depth-map, while in the disparity-driven approach we are smoothing the disparity. Let us consider this difference in details. At this point to discuss the advantages and shortcomings of the approaches we will build a simple model – an artificial disparity field and the corresponding depth-map (figure 4.1).

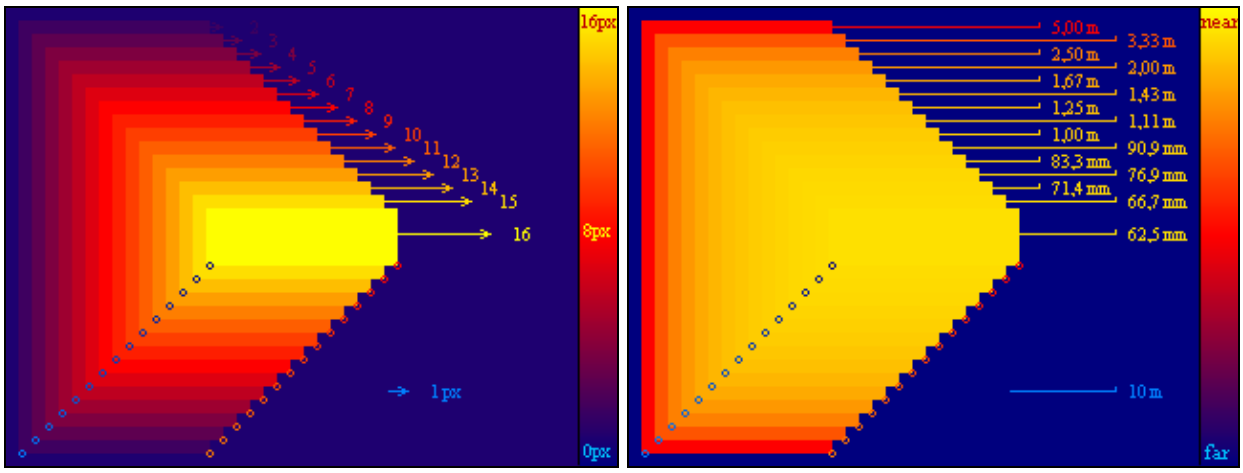

Figure 4.1: RnB Pyramid: **Left:** Disparity domain; **Right:** Depth domain.

An *RnB pyramid* is a number of half – overlapping layers with different disparity or in other words with different depth – distance from an observer. The pyramid illustrates disparity/depth differences (gradients) between a layer and the background in the *red* point row and between consequent layers in the *blue* point row. Since in stereo images which are represented as a number of pictures, we deal with the pixels' displacements, we built the RnB pyramid in disparity domain in such a way, that the displacement between any two adjacent layers is constant, and then translated it into the depth domain.

Now let us consider in plots and tables the differences between red and blue point rows of the RnB pyramid in depth domain and disparity domain. If we have N layers L_i where $i \in [1;\ N]$ and L_1 is the background layer, we can define the disparity and depth values as follows:

$$z'_i = i \cdot \Delta d_i, \tag{4.1.8}$$

$$z_i = \frac{\alpha_u \Delta t_x}{i \cdot \Delta d_i}, \tag{4.1.9}$$

where Δd_i is the shift of layer L_i in pixels. Note, we use the constant layers shift: $\Delta d_i = \Delta d$, $\forall i \in [1;\ N]$ ($\Delta d = 1_{pixel}$ and $\alpha_u \Delta t_x = 10$ in figure 4.1).

From formulae (4.1.8) and (4.1.9) we can derive the formula (4.1.3) what proves the correctness of the RnB pyramid principle. Moreover, using the formula (4.1.3) we can write down the equation, characterizing the correlation between the gradients on the RnB pyramid in depth and disparity domains:

$$\left|\nabla z\right|^2 = \left|\nabla\left(\frac{\alpha_u \Delta t_x}{z'}\right)\right|^2 = \left|-\frac{\alpha_u \Delta t_x}{(z')^2}\nabla z'\right|^2 = \frac{\alpha_u^2 \Delta t_x^2}{(z')^4} \cdot \left|\nabla z'\right|^2. \tag{4.1.10}$$

Now let us return to the RnB pyramid. Since all the points of the red and blue point rows lie on the corner-edge of the RnB pyramid and the directional derivatives are equal there, we can write $\left|\nabla z\right|^2 = z_x^2 + z_y^2 = 2z_x^2$. The gradient in the blue point row we calculate with the help of the following formulae:

$$\left|\nabla z'_i\right|^2 = 2\left(z'_{i+1} - z'_i\right)^2 = 2\left((i+1)\Delta d - i\Delta d\right)^2 = 2\Delta d^2, \tag{4.1.11}$$

$$\left|\nabla z_i\right|^2 = 2\left(z_{i+1} - z_i\right)^2 = 2\left(\frac{\alpha_u \Delta t_x}{(i+1)\Delta d} - \frac{\alpha_u \Delta t_x}{i\Delta d}\right)^2 = 2\left(\frac{\alpha_u \Delta t_x}{i(i+1)\Delta d}\right)^2 \tag{4.1.12}$$

and the gradient in the red point row we will calculate by means of the next pair of formulae:

$$\left|\nabla z'_i\right|^2 = 2\left(z'_{i+1} - z'_1\right)^2 = 2\left((i+1)\Delta d - \Delta d\right)^2 = 2\left(i\Delta d\right)^2, \tag{4.1.13}$$

$$\left|\nabla z_i\right|^2 = 2\left(z_{i+1} - z_1\right)^2 = 2\left(\frac{\alpha_u \Delta t_x}{(i+1)\Delta d} - \frac{\alpha_u \Delta t_x}{\Delta d}\right)^2 = 2\left(\frac{i\alpha_u \Delta t_x}{(i+1)\Delta d}\right)^2. \tag{4.1.14}$$

Now let us illustrate the behavior of the gradient magnitude on the edges of far and near layers. Using formulae (4.1.11) – (4.1.14) we can build the table 4.1, substituting for the *far* column values of gradient magnitude with $i = 1$, and for *near* column values of gradient magnitude with $i \to \infty$. With the help of the color we distinguish gradient magnitudes in the red or blue point rows.

Table 4.1: Gradient magnitudes on the RnB pyramid.

	$\lvert\nabla z_i\rvert^2 / \lvert\nabla z'_i\rvert^2$	Far	Near
Disparity domain	$2\Delta d^2$	*const* 1	*const* 1
	$2(i\Delta d)^2$	*const* 1	∞
Depth domain	$2\left(\frac{\alpha_u \Delta t_x}{i(i+1)\Delta d}\right)^2$	*const* 2	*0*
	$2\left(\frac{i\alpha_u \Delta t_x}{(i+1)\Delta d}\right)^2$	*const* 2	*4 const* 2

At table 4.1 we can see that the gradient is constant in the blue point row of the RnB pyramid in disparity domain and the gradient in the red point row linearly increases from layer to layer. Indeed, if we look once more at figure 4.1 we can see that the contrast between layers is constant and the edge between the 16-th layer and first one is much mire bigger than contrast between second and first layers. In depth domain we have a different picture. Here the gradient in blue point row not constant but has decreasing character from far to near layers and the gradient in red point row, as well as in disparity domain, increases but is bounded. Comparing these results with the figure 4.1 we make sure that the contrast between consequent layers decreases when coming from far to near layers and it is almost impossible to distinguish edge between the 16-th and 15-th layer. And the contrast between a layer and background is very good distinguished by an unarmed eye, it increases when coming closer to an observer and the contrast between 16-th and 15-th layer not bigger than four times of contrast between the second and the first layer.

We would like to compare the values of $const_1$ and $const_2$ to understand the difference of the gradient behaviour on far situated objects. For this purpose we will derive the gradient correlation equation like (4.1.10) from equations (4.1.11), (4.1.12) for the blue point row:

$$\left|\nabla z_i\right|^2 = 2\left(\frac{\alpha_u \Delta t_x}{i(i+1)\Delta d}\right)^2 = \left(\frac{\alpha_u \Delta t_x}{i(i+1)\Delta d^2}\right)^2 \cdot 2\Delta d^2 = \left(\frac{\alpha_u \Delta t_x}{i(i+1)\Delta d^2}\right)^2 \cdot \left|\nabla z'_i\right|^2 \qquad (4.1.15)$$

and from equations (4.1.13), (4.1.14) for the red point row:

$$\left|\nabla z_i\right|^2 = 2\left(\frac{i\alpha_u \Delta t_x}{(i+1)\Delta d}\right)^2 = \left(\frac{\alpha_u \Delta t_x}{(i+1)\Delta d^2}\right)^2 \cdot 2(i\Delta d)^2 = \left(\frac{\alpha_u \Delta t_x}{(i+1)\Delta d^2}\right)^2 \cdot \left|\nabla z'_i\right|^2 . \qquad (4.1.16)$$

For the far layer, i.e. $i = 1$ we achieve the same formula for (4.1.15) and (4.1.16):

$$|\nabla z_1|^2 = \left(\frac{\alpha_u \Delta t_x}{2\Delta d^2}\right)^2 \cdot |\nabla z'_1|^2, \tag{4.1.17}$$

consequently

$$const_2 = \left(\frac{\alpha_u \Delta t_x}{2\Delta d^2}\right)^2 \cdot const_1 . \tag{4.1.18}$$

Now it is very important to investigate the smoothness process on red and blue point rows of the pyramid. Let us put the gradient magnitudes from formulae (4.1.11) – (4.1.14) in the Charbonnier regularizer (3.5.6) and plot it to show the contribution of the smoothness term to the energy functional:

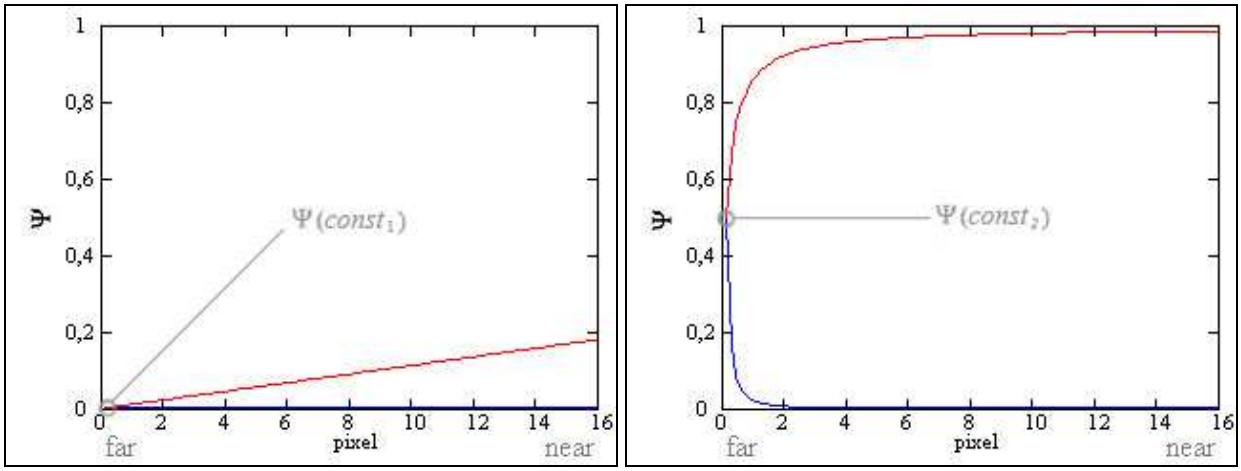

Figure 4.2: Charbonnier penalizer on gradient magnitude ($\lambda = 0{,}025$): **Left:** Argument: gradient magnitudes in the blue point row (blue graph) and the red point row (red graph) of the RnB pyramid in disparity domain; **Right:** Argument: gradient magnitudes in the blue point row (blue graph) and the red point row (red graph) of the RnB pyramid in depth domain.

As we can see from the figure 4.2, the contribution to the energy functional of the smoothness term in depth domain is much larger than the contribution of the smoothness term in disparity domain. Moreover, the difference between values of red and blue graphs in depth domain for 16 pixels layer shift reaches almost the value of 1, when the same difference in disparity domain less then *0,2*. It means, that the average deviation of the gradient magnitude in depth domain almost 5 times bigger than in disparity domain. Due to such a big deviation we are not afforded to reduce the smoothness parameter φ too much. Also the plots prove the idea, described by formulae (4.1.10) and (4.1.17): when an object is far situated, the disparity in its stereo image z' is pretty small, then the value of the term $\frac{\alpha_u^2 \Delta t_x^2}{(z')^4}$ becomes unbounded large and therefore especially at far objects the contribution to the energy functional of the smoothness term becomes too large in comparison with the contribution of the data term. In such a way all the far distant objects will be blurred into the background. Indeed, looking back to our RnB pyramid from figure 4.1, where we have $\Delta d = 1_{pixel}$ and

$\alpha_u \Delta t_x = 10$ and substituting these values into the formula (4.1.18), we calculate that $const_2 = 25 \cdot const_1$! Anyway we have to have in mind that the nonlinear behavior more problematic than scale.

To summarize this discussion, let us represent the results of our research of the smoothness process on the RnB pyramid in table 4.2:

Table 4.2: Smoothness process on edges of the RnB pyramid.

		Far	near
Disparity domain		*Edge preservation*	*Edge preservation*
		Edge preservation	*Small blurring*
Depth domain		***Overblurring***	*Edge preservation*
		Overblurring	***Overblurring***

and some examples at the figure 4.3:

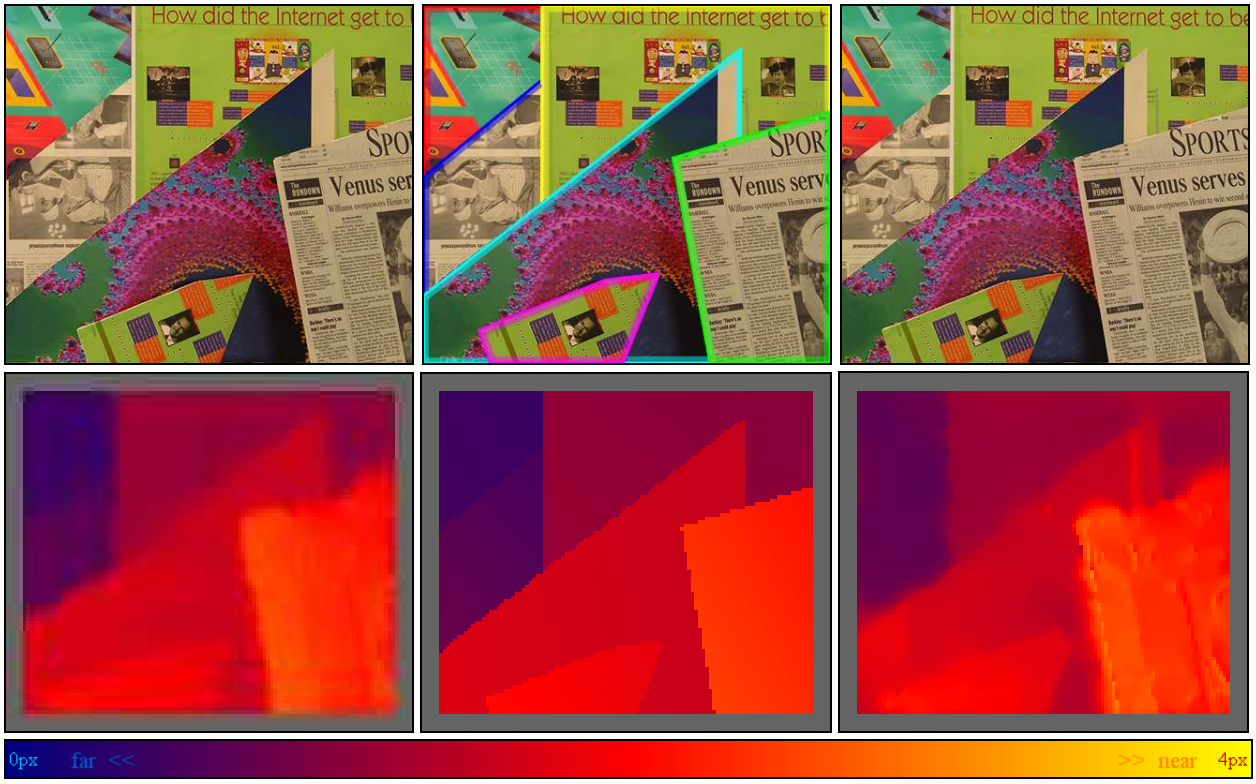

Figure 4.3: "Poster" scene**: Top Left:** The first picture of the scene; **Top Centre:** Segmented object on the scene; **Top Right:** The eighth picture of the scene; **Bottom Left:** Result achieved with the disparity-driven method; **Bottom Centre:** True solution; **Bottom Right:** Result achieved with the depth-driven method. (The original scene taken from [WMU07]).

Figure 4.3 illustrates the practical difference between depth-driven and disparity-driven approaches. At the bottom left image we can see that edge between the red and blue segments is preserved (for the color of segments refer to the top centre picture of the figure 4.1), while at the bottom right picture we observe this edge totally overblurred. At other hand the figure demonstrates that the nearest object in the "Poster" scene is not so blurry with the depth-driven method like in result achieved by the disparity-driven method. So the nearest layer does not look like whole segment. As a conclusion we can recommend to use the disparity-driven method for the depth-map reconstruction in further work, as it is the method, which has more advantages.

4.2 First order Taylor expansion

As we have discussed in the third chapter, the most problems in data term we have due to implicitness of the first term in formula (3.2.1). We have mentioned that there are several methods to overcome this problem and one of them is the very popular method of first-order Taylor expansion. To explain how it works, let us consider the Taylor series.

In mathematics, the Taylor series is a representation of a function as an infinite sum of terms, calculated from the values of its derivatives at a single point. It may be regarded as a limit of the Taylor *polynomials*[5]. Taylor series are named in honor of English mathematician *Brook Taylor* (August 18, 1685 – November 30, 1731).

The Taylor series of a real or complex function f that is infinitely differentiable in a neighborhood of a real or complex number a, is the power series:

$$f(x) = f(a) + \sum_{k=1}^{\infty} \frac{f^{(k)}(a)}{k!}(x-a)^k \,. \qquad (4.2.1)$$

The Taylor series need not in general be a convergent series, but often it is. The limit of a convergent Taylor series need not in general be equal to the function value $f(x)$, but often it is. If $f(x)$ is equal to its Taylor series in a neighborhood of a, it is called to be *analytic* in this neighborhood. If $f(x)$ is equal to its Taylor series everywhere it is called *entire* [WW07].

A Taylor series can be used to calculate the value of an entire function in every point, if the value of the function, and of all its derivatives, is known at a single point. Uses of the Taylor series for entire functions include the partial sums of the series can be used as approximations of the entire function. These approximations are good if sufficiently many terms are included.

Let us rewrite (4.2.1) in a form, than is more suitable for our problem:

[5] The expressions that is constructed from one or more variables and constants, using only the operations of addition, subtraction, multiplication and constant positive whole number exponents.

$$f(x+\Delta x) = f(x) + \sum_{k=1}^{\infty} \frac{f^{(k)}(x)}{k!} \Delta x^k \ . \tag{4.2.2}$$

Here we approximate the function f in a point $x+\Delta x$ within a neighborhood with radius Δx. We assume that we deal with analytic function in this neighborhood and the neighborhood's radius is small enough to neglect all the terms in the Taylor series except the first one, i.e. we assume that that the function f is sufficiently smooth and the value of Δx is sufficiently small. This approach gives us the first-order Taylor approximation of a function:

$$f(x+\Delta x) = f(x) + \Delta x \cdot f'(x), \tag{4.2.3}$$

or in two dimensional case:

$$f(x+\Delta x, y) = f(x,y) + \Delta x \cdot f_x(x,y). \tag{4.2.4}$$

In such a way we have come to the formula (3.2.2). Now let us write down the simplest energy functional, that is used in this paper:

$$E(z) = \iint_{\Omega} \left(I_1(u,v) - I_2(u,v) - \frac{\alpha_u \Delta t_x}{z} I_{2u}(u,v) \right)^2 + \varphi \cdot \left| \nabla z \right|^2 du dv \tag{4.2.5}$$

and its Euler – Lagrange equation:

$$0 = -I_{2u}(u,v) \frac{\alpha_u \Delta t_x}{z^2} \left(I_1(u,v) - I_2(u,v) - \frac{\alpha_u \Delta t_x}{z} I_{2u}(u,v) \right) + \varphi \cdot \Delta z \ . \tag{4.2.6}$$

We got rid of implicitness, but the Euler – Lagrange equation is still non linear. So, we can find the solution to the equation (4.2.6) as a steady-time state of the following differential equation:

$$\frac{\partial z}{\partial t} = -I_{2u} \frac{\alpha_u \Delta t_x}{z^2} \left(I_1 - I_2 - \frac{\alpha_u \Delta t_x}{z} I_{2u} \right) + \varphi \cdot \Delta z, \tag{4.2.7}$$

here we omit the functions I arguments for simplicity and introduce an artificial time parameter t.

To consider the disparity–driven method with the first-order Taylor approximation, let us rewrite the energy functional (4.1.6) using the formula (4.2.4) and Tichonov regularizer in smoothness term:

$$E(z') = \iint_{\Omega} \left(I_1(u,v) - I_2(u,v) - z'(u,v) I_{2u}(u,v) \right)^2 + \varphi \cdot \left| \nabla z' \right|^2 du dv \tag{4.2.8}$$

and its Euler – Lagrange equation:

$$I_{2u}(u,v)\big(I_1(u,v)-I_2(u,v)-z'(u,v)I_{2u}(u,v)\big)+\varphi\cdot\Delta z=0\,. \qquad (4.2.9)$$

Equations like (4.2.7) are solved with help of time-marching numerical schemes and equations like (4.2.9) being linear, are possible to solve with the help of fast linear numerical schemes. We will consider this numerical schemes and discretization techniques in paragraphs 4.5 and 4.6.

The method of the first order Taylor expansion has some ponderable disadvantages. First of all it follows the assumptions that the value $\left|\frac{\alpha_u\Delta t_x}{z}\right|$ (or $|z'|$) is sufficiently small, which is not obliged to be always true. This limitation leads us to another problem with implementing the method of coarse levels, which implies big disparities at the finer levels. This problem is described in detail in paragraph 4.7.

4.3 Linear interpolation of data term

Another approach to get rid of implicitness in data term is the method of linear interpolation. This method related closer to numerical schemes and deals with the discrete data. In practice our pictures of a stereo image $I_i(u,v)$ are represented as a two-dimensional arrays, or discrete functions: $I_i(u,v):\mathrm{N}^+\times\mathrm{N}^+\mapsto R$. By another words, arguments of these functions in discrete case must be integers greater or equal to zero. The value $\frac{\alpha_u\Delta t_x}{z}\in Q$ is rational, hence the value of $u+\frac{\alpha_u\Delta t_x}{z}\in Q$ is rational as well. To overcome this problem we apply the linear interpolation [Mei02].

The main idea of the method of linear interpolation is to represent the term $\frac{\alpha_u\Delta t_x}{z}\in Q$ as a sum of two numbers: integer $A\in\mathrm{N}^+$ and $b\in R$, such that $|b|<1$. Thus we have:

$$I\left(u+\frac{\alpha_u\Delta t_x}{z},v\right)=I\big(u+A+b,v\big), \qquad (4.3.1)$$

using the linear interpolation we have:

$$I\big(u+A+b,v\big)=(1-|b|)\cdot I(u+A,v)+|b|\cdot I(u+A+sign(b),v)\,. \qquad (4.3.2)$$

This approach admits to still having implicitness, but it excludes any kind of limitation on value of $\frac{\alpha_u\Delta t_x}{z}$ and could be considered as a kind of "incremental calculation". Moreover we proof that the method of linear interpolation is more general, and hence more useful than the method of first order Taylor expansion.

Theorem 4.1: if the value of $\frac{\alpha_u \Delta t_x}{z}$ is small enough, then method of linear interpolation is identical to the first order Taylor expansion.

Proof: if $\left|\frac{\alpha_u \Delta t_x}{z}\right| << 1$, then, since $\frac{\alpha_u \Delta t_x}{z} = A + b$, we have $A = 0$ and $b = \frac{\alpha_u \Delta t_x}{z}$.

Now we can write:

$$\begin{aligned}
&I(u + \frac{\alpha_u \Delta t_x}{z}, v) \\
&= I(u + A + b, v) \\
&= (1 - |b|) \cdot I(u + A, v) + |b| \cdot I(u + A + sign(b), v) \\
&= (1 - |b|) \cdot I(u, v) + |b| \cdot I(u + sign(b), v) \\
&= I(u, v) + |b| \cdot \left(I(u + sign(b), v) - I(u, v)\right) \\
&= I(u, v) + sign(b) \cdot |b| \cdot I_u(u, v) \\
&= I(u, v) + b \cdot I_u(u, v) \\
&= I(u, v) + \frac{\alpha_u \Delta t_x}{z} \cdot I_u(u, v).
\end{aligned}$$

Now let us write the energy functional and then derive its Euler – Lagrange equation with data term of the Euler – Lagrange equation interpolated with help of the linear interpolation:

$$E(z) = \iint_{\Omega} \left(I_1(u, v) - I_2\left(u + \frac{\alpha_u \Delta t_x}{z}, v\right)\right)^2 + \varphi \cdot |\nabla z|^2 \, du dv \tag{4.3.3}$$

and its Euler – Lagrange equation:

$$0 = -I_{2u}\left(u + \frac{\alpha_u \Delta t_x}{z}, v\right) \frac{\alpha_u \Delta t_x}{z^2} \left(I_1(u, v) - I_2\left(u + \frac{\alpha_u \Delta t_x}{z}, v\right)\right) + \varphi \cdot \Delta z \,. \tag{4.3.4}$$

Now we are ready to apply the linear interpolation to the data term of the Euler – Lagrange equation. Since we calculate the derivatives via finite difference numerical scheme, we know, that the function $I_{2u}\left(u + \frac{\alpha_u \Delta t_x}{z}, v\right)$ is a piecewise constant function, so we assume that $I_{2u}\left(u + \frac{\alpha_u \Delta t_x}{z}, v\right) = I_{2u}(u + A + b, v) \approx I_{2u}(u + A, v)$:

$$\begin{aligned}
&- I_{2u}(u + A, v) \frac{\alpha_u \Delta t_x}{z^2} \left(I_1(u, v) - (1 - |b|) I_2(u + A, v) - |b| I_2(u + A + sign(b), v)\right) \\
&+ \varphi \cdot \Delta z = 0.
\end{aligned} \tag{4.3.5}$$

As in (4.2.6) we have got a nonlinear equation. So we will use the time-marching numerical scheme to solve.

$$-I_{2u}(u+A,v)\frac{\alpha_u \Delta t_x}{z^2}\left(I_1(u,v)-(1-|b|)I_2(u+A,v)-|b|I_2(u+A+sign(b),v)\right) + \varphi\cdot\Delta z = \frac{\partial z}{\partial t} \tag{4.3.6}$$

To consider the disparity–driven approach with the linear interpolation method, let us rewrite the energy functional (4.1.6) with Tichonov regularizer in smoothness term:

$$E(z') = \iint_{\Omega}\left(I_1(u,v)-I_2(u+z'(u,v),v)\right)^2 + \varphi\cdot|\nabla z'|^2\,dudv \tag{4.3.7}$$

and its Euler – Lagrange equation:

$$I_{2u}(u+z'(u,v),v)\left(I_1(u,v)-I_2(u+z'(u,v),v)\right)+\varphi\cdot\Delta z'=0\,. \tag{4.3.8}$$

Now we define $z'(u,v)=A'+b'$ and apply formula (4.3.2):

$$I_{2u}(u+A',v)\left(I_1(u,v)-(1-|b'|)I_2(u+A',v)-|b'|I_2(u+A'+sign(b'),v)\right)+\varphi\cdot\Delta z'=0\,. \tag{4.3.9}$$

Let us now summarize the last two paragraphs. We have derived four different Euler–Lagrange equations: (4.2.6), (4.2.9), (4.3.5) and (4.3.9) and we can combine the data terms of the Euler – Lagrange equations in the following table:

Table 4.3: Data terms of Euler – Lagrange equations.

	First-order Taylor approximation	Linear interpolation
Depth driven	$-I_{2u}\frac{\alpha_u\Delta t_x}{z^2}\left(I_1-I_2-\frac{\alpha_u\Delta t_x}{z}I_{2u}\right)$	$-I_{2u}(u+A)\frac{\alpha_u\Delta t_x}{z^2}\cdot$ $\left(I_1-(1-b)I_2(u+A)-bI_2(u+A+1)\right)$
Disparity driven	$I_{2u}\left(I_1-I_2-z'I_{2u}\right)$	$I_{2u}(u+A')\left(I_1-(1-b')I_2(u+A')-b'I_2(u+A'+1)\right)$

Results, according to the table 4.3 are depicted at figure 4.5. The background object in the scene (red layer in the top centre picture of figure 4.5) has shifted less than 1 pixel and the nearest object in the scene (cyan layer in the top centre picture of figure 4.5) has shifted more than 2 pixels. At the left middle and bottom pictures of the figure 4.5 we observe that the first order Taylor approximation, as it was predicted, can not handle with big disparities for the depth-driven method as well as for the disparity-driven method: the cyan layer does not seen at all, and the yellow and magenta layers have numerous mistakes – outliers values, which are depicted with bright yellow or dark

blue colours at the figure. The results, achieved with the linear interpolation method show good performance. The depth driven method is characterized by the overblurred background as it was described in the first paragraph of this chapter. We can conclude that the linear interpolation method may deal with much more bigger displacements than the first-order Taylor approximation. Experiments show that the linear interpolation method successfully handles displacement till 4 pixels, while the first-order Taylor approximation can give good results with displacements not bigger than 2 pixels and require more accurate and heavy coarse levels technique than the linear interpolation method.

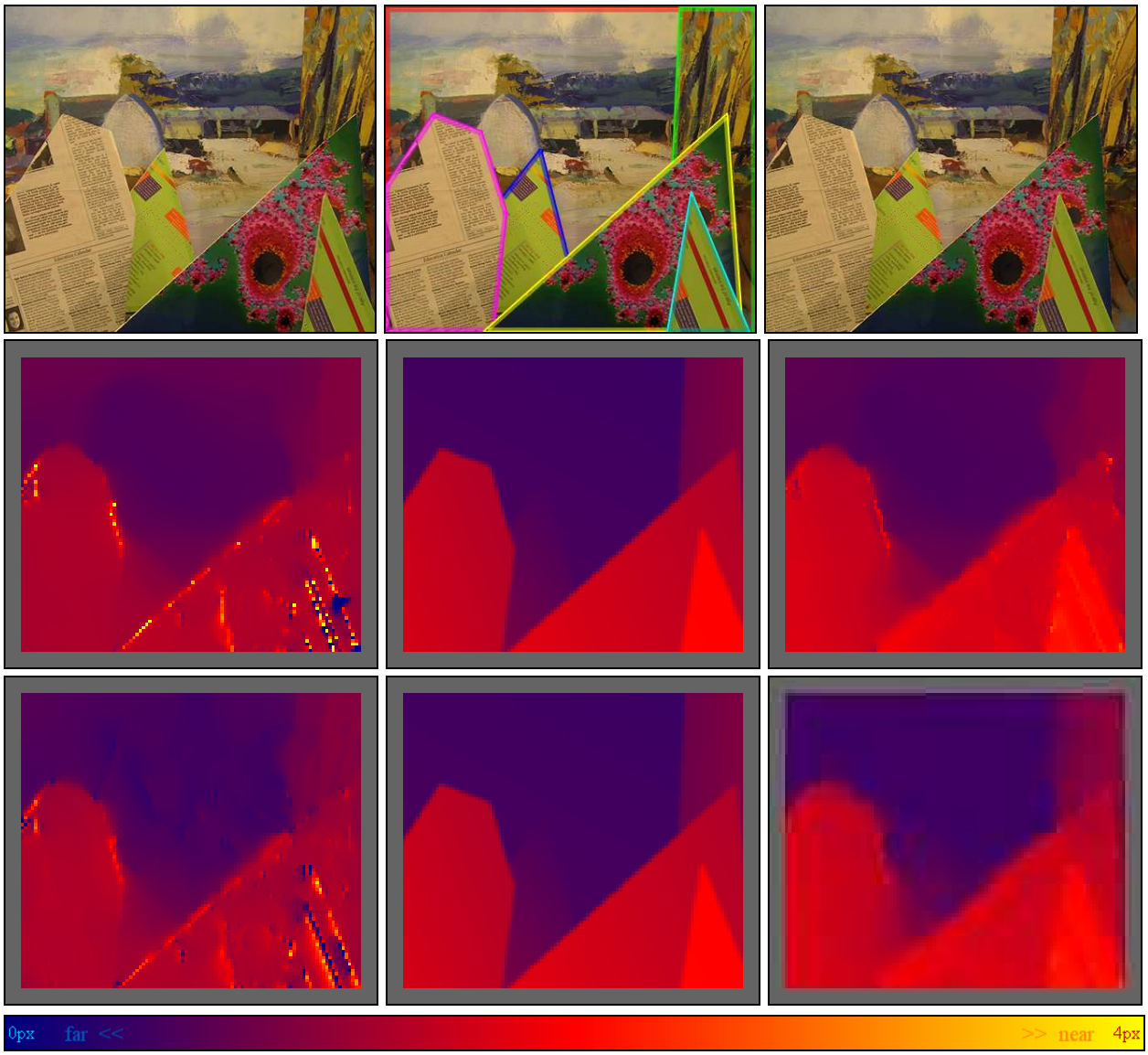

Figure 4.4: "Barn1" scene**: Top Left:** The first picture of the scene; **Top Centre:** Segmented object on the scene; **Top Right:** The eighth picture of the scene; **Middle Left:** Result achieved with the depth-driven method and first-order Taylor approximation; **Middle Centre:** True solution; **Middle Right:** Result achieved with the depth-driven method and linear interpolation; **Bottom Left:** Result achieved with the disparity-driven method and first-order Taylor approximation; **Bottom Centre:** True solution; **Bottom Right:** Result achieved with the disparity-driven method and linear interpolation. (The original scene is taken from [WMU07]).

4.4 Penalization in the data term

To control outliers in data term we will use a very popular technique of data term penalization. After we have constructed the data term, using suitable constancy assumptions, it is possible to make it more robust. Till this moment we considered data terms that penalize deviations from constancy assumptions in a quadratic way, to modify it to be more robust we will use non quadratic penalisation strategy which renders the estimation more robust with respect to violations of the model assumption like appearing or occluding objects. The main idea of this strategy is to penalize outliers less severely than in quadratic setting.

For the regularization of data term we will use penalizing functions like (2.3.4), or particularly saying the same penalizing functions that we already use for smoothness term: Tichonov penalizer (3.5.3), Charbonnier penalizer (3.5.6) and Perona-Malik penalizer (3.5.7). The only difference between the penalizing functions of data and smoothness terms will be the index: "d" for data term and "s" for smoothness term:

$$\Psi_o(s^2) = \sqrt{s^2 + \lambda_o^2}\ ,\ o \in \{d, s\}\,, \tag{4.4.1}$$

or by another words, different parameter λ.

Let us plot in figure 4.5 the graphs of the corresponding penalizing functions. We know that the quadratic and Charbonnier regularizers are convex in s, what leads to an opportunity for building simple globally convergent algorithms [AL95]. Apart from them we also have Perona-Malik regularizer which, as we can see, is not convex, and using it in data term we can not guarantee well-posedness for the problem of the depth-map reconstruction. Anyway such penalizers are more robust and result in energy functionals that have multiple minima [BA96]. Also pay your attention at the range of the co-domain of the functions plotted in figure 4.5.

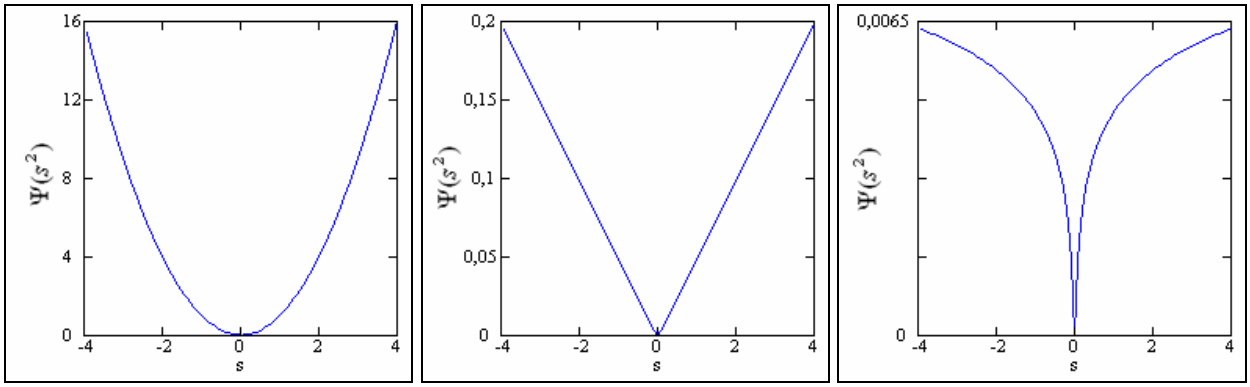

Figure 4.5: Comparison of different penalizing functions: **Left:** Tichonov (quadratic); **Centre:** Charbonnier (total variation) ($\lambda = 0{,}025$); **Right:** Perona-Malik (balanced forward-backward) ($\lambda = 0{,}025$);

We will consider the whole data term as one entity that represents all constancy assumptions that are imposed on the image data. It means that all assumptions are robustified jointly – by applying a single robust function to the complete data term.

Such an approach is called *joint robustification*. Schematically we can illustrate the technique of data term regularization in the following formula:

$$E(z)=\iint_{\Omega}\Psi_d\left(\sum_i Constancy\_assumption_i\right)+\varphi\cdot\Psi_s(|\nabla z|^2)dudv\,. \tag{4.4.2}$$

4.5 Time marching numerical scheme

Now let us build the implicit time marching numerical scheme for our problem. First of all let us write the energy functional with the data term built on grey-value constancy assumption and gradient constancy assumption with help of regularization and the smoothness term built with the help of a regularization function:

$$E(z)=\iint_{\Omega}\Psi\begin{pmatrix}\Theta\cdot\left|I_1(u,v)-I_2(u+\frac{\alpha_u\Delta t_x}{z},v)\right|^2+\\(1-\Theta)\cdot\left|\nabla I_1(u,v)-\nabla I_2(u+\frac{\alpha_u\Delta t_x}{z},v)\right|^2\end{pmatrix}+\varphi\cdot\Psi(|\nabla z|^2)dudv \tag{4.5.1}$$

we decompose the second component of the data term:

$$\begin{aligned}&\left|\nabla I_1(u,v)-\nabla I_2(u+\frac{\alpha_u\Delta t_x}{z},v)\right|^2=\\&\left|\begin{pmatrix}I_{1u}(u,v)\\I_{1v}(u,v)\end{pmatrix}-\begin{pmatrix}I_{2u}(u+\frac{\alpha_u\Delta t_x}{z},v)\\I_{2v}(u+\frac{\alpha_u\Delta t_x}{z},v)\end{pmatrix}\right|^2=\\&\left|\begin{pmatrix}I_{1u}(u,v)-I_{2u}(u+\frac{\alpha_u\Delta t_x}{z},v)\\I_{1v}(u,v)-I_{2v}(u+\frac{\alpha_u\Delta t_x}{z},v)\end{pmatrix}\right|^2=\\&\left(I_{1u}(u,v)-I_{2u}(u+\frac{\alpha_u\Delta t_x}{z},v)\right)^2+\left(I_{1v}(u,v)-I_{2v}(u+\frac{\alpha_u\Delta t_x}{z},v)\right)^2.\end{aligned} \tag{4.5.2}$$

Using the formula (4.5.2) we can rewrite the energy functional (4.5.1) in the following form:

$$E(z)=\iint\limits_{\Omega}\Psi\left(\begin{array}{l}\Theta\cdot\left(I_1(u,v)-I_2(u+\frac{\alpha_u\Delta t_x}{z},v)\right)^2+\\(1-\Theta)\cdot\left(I_{1u}(u,v)-I_{2u}(u+\frac{\alpha_u\Delta t_x}{z},v)\right)^2+\\(1-\Theta)\cdot\left(I_{1v}(u,v)-I_{2v}(u+\frac{\alpha_u\Delta t_x}{z},v)\right)^2\end{array}\right)+\varphi\cdot\Psi(|\nabla z|^2)dudv\,. \quad (4.5.3)$$

Now let us apply the first-order Taylor expansion technique and rewrite the energy functional (4.5.3) omitting the function argument indexing:

$$E(z)=\iint\limits_{\Omega}\Psi\left(\begin{array}{l}\Theta\cdot\left(I_1-I_2-\frac{\alpha_u\Delta t_x}{z}I_{2u}\right)^2+\\(1-\Theta)\cdot\left(I_{1u}-I_{2u}-\frac{\alpha_u\Delta t_x}{z}I_{2uu}\right)^2+\\(1-\Theta)\cdot\left(I_{1v}-I_{2v}-\frac{\alpha_u\Delta t_x}{z}I_{2vu}\right)^2\end{array}\right)+\varphi\cdot\Psi(|\nabla z|^2)dudv\,. \quad (4.5.4)$$

And now we are ready to derive the Euler – Lagrange equation in meaning of the time marching scheme:

$$\begin{array}{l}\frac{\partial z}{\partial t}=-\Psi'\left(\begin{array}{l}\Theta\cdot\left(I_1-I_2-\frac{\alpha_u\Delta t_x}{z}I_{2u}\right)^2+\\(1-\Theta)\cdot\left(I_{1u}-I_{2u}-\frac{\alpha_u\Delta t_x}{z}I_{2uu}\right)^2+\\(1-\Theta)\cdot\left(I_{1v}-I_{2v}-\frac{\alpha_u\Delta t_x}{z}I_{2vu}\right)^2\end{array}\right)\cdot\\\left(\begin{array}{l}\Theta\cdot I_{2u}\frac{\alpha_u\Delta t_x}{z^2}\left(I_1-I_2-\frac{\alpha_u\Delta t_x}{z}I_{2u}\right)+\\(1-\Theta)\cdot I_{2uu}\frac{\alpha_u\Delta t_x}{z^2}\left(I_{1u}-I_{2u}-\frac{\alpha_u\Delta t_x}{z}I_{2uu}\right)+\\(1-\Theta)\cdot I_{2vu}\frac{\alpha_u\Delta t_x}{z^2}\left(I_{1v}-I_{2v}-\frac{\alpha_u\Delta t_x}{z}I_{2vu}\right)\end{array}\right)+\\\varphi\cdot div\left(\Psi'(|\nabla z|^2)\cdot\nabla z\right)\end{array} \quad (4.5.5)$$

Now let us discretize the data term of the Euler – Lagrange equation and the smoothness term of the Euler – Lagrange equation separately, and then combine the results into the desired numerical scheme. To precede let us first of all introduce some notations for simplicity. We will replace the long diffusivity function by the short notation:

$$\Psi'(\left|\nabla z_{i,j}\right|^2) = g_{i,j}. \tag{4.5.6}$$

Next, let z^* will be the next depth-map state, i.e. the state at the next iteration step, and z is the current state of the depth-map, i.e. the state at the current time. Then the forward approximation of the time derivative could be written as follows:

$$\frac{\partial z}{\partial t} \approx \frac{z^*_{i,j} - z_{i,j}}{\tau}, \tag{4.5.7}$$

where τ is the time step and indexes i, j denote the current depth-map element, or by another words the array indexes.

We will start with the data term of the Euler – Lagrange equation discretization. Here for the simplicity, we designate the whole argument of the diffusivity function in data term of the Euler – Lagrange equation through one term:

$$\begin{aligned} Forma_{i,j} = \Theta \cdot \left((I_1)_{i,j} - (I_2)_{i,j} - \frac{\alpha_u \Delta t_x}{z_{i,j}} (I_{2u})_{i,j} \right)^2 + \\ (1-\Theta) \cdot \left((I_{1u})_{i,j} - (I_{2u})_{i,j} - \frac{\alpha_u \Delta t_x}{z_{i,j}} (I_{2uu})_{i,j} \right)^2 + \\ (1-\Theta) \cdot \left((I_{1v})_{i,j} - (I_{2v})_{i,j} - \frac{\alpha_u \Delta t_x}{z_{i,j}} (I_{2vu})_{i,j} \right)^2. \end{aligned} \tag{4.5.8}$$

And now let us write down the discretized data term of the Euler – Lagrange equation:

$$\begin{aligned} Data\_term_{Euler-Lagrange} = g_{data}(Forma_{i,j}) \cdot \\ \left(\begin{array}{l} \Theta \cdot (I_{2u})_{i,j} \frac{\alpha_u \Delta t_x}{z_{i,j}^2} \left((I_1)_{i,j} - (I_2)_{i,j} - \frac{\alpha_u \Delta t_x}{z_{i,j}} (I_{2u})_{i,j} \right) + \\ (1-\Theta) \cdot (I_{2uu})_{i,j} \frac{\alpha_u \Delta t_x}{z_{i,j}^2} \left((I_{1u})_{i,j} - (I_{2u})_{i,j} - \frac{\alpha_u \Delta t_x}{z_{i,j}} (I_{2uu})_{i,j} \right) + \\ (1-\Theta) \cdot (I_{2vu})_{i,j} \frac{\alpha_u \Delta t_x}{z_{i,j}^2} \left((I_{1v})_{i,j} - (I_{2v})_{i,j} - \frac{\alpha_u \Delta t_x}{z_{i,j}} (I_{2vu})_{i,j} \right) \end{array} \right). \end{aligned} \tag{4.5.9}$$

Now let us proceed with the smoothness term of the Euler – Lagrange equation discretization. For this purpose let us rewrite it in the following form:

$$\begin{aligned} Smoothness\_term_{Euler-Lagrange} = -\alpha \cdot div\left(\Psi'(\left|\nabla z_{i,j}\right|^2) \cdot \nabla z_{i,j} \right) = \\ -\alpha \cdot div\left(g_{i,j} \cdot \nabla z_{i,j} \right) = -\alpha \cdot \left(\partial_u \left(g_{i,j} \cdot (z_{i,j})_u \right) + \partial_v \left(g_{i,j} \cdot (z_{i,j})_v \right) \right); \end{aligned} \tag{4.5.10}$$

the next step is to discretize the term $\partial_u\left(g_{i,j}\cdot(z_{i,j})_u\right)$:

$$\partial_u\left(g_{i,j}\cdot(z_{i,j})_u\right)=\partial_u\left(g_{i+1/2,j}\cdot\frac{z_{i+1,j}-z_{i,j}}{\Delta u}\right)= \\ \frac{g_{i+1,j}+g_{i,j}}{2}\cdot\frac{z_{i+1,j}-z_{i,j}}{\Delta u}-\frac{g_{i,j}+g_{i-1,j}}{2}\frac{z_{i,j}-z_{i-1,j}}{\Delta u}. \tag{4.5.11}$$

The term $\partial_v\left(g_{i,j}\cdot(z_{i,j})_v\right)$ we discretize in the same way. Now we assume that the grid steps in u and v directions are equal to 1: $\Delta u=\Delta v=1$; so we can rewrite the formula (4.5.10):

$$Smoothness\_term_{Euler-Lagrange}= \\ -\varphi\cdot\begin{pmatrix}\frac{g_{i+1,j}+g_{i,j}}{2}(z_{i+1,j}-z_{i,j})-\frac{g_{i,j}+g_{i-1,j}}{2}(z_{i,j}-z_{i-1,j})+\\ \frac{g_{i,j+1}+g_{i,j}}{2}(z_{i,j+1}-z_{i,j})-\frac{g_{i,j}+g_{i,j-1}}{2}(z_{i,j}-z_{i,j-1})\end{pmatrix}= \\ -\frac{\varphi}{2}\cdot\begin{pmatrix}(g_{i+1,j}+g_{i,j})z_{i+1,j}+(g_{i,j}+g_{i-1,j})z_{i-1,j}+(g_{i,j+1}+g_{i,j})z_{i,j+1}\\ +(g_{i,j}+g_{i,j-1})z_{i,j-1}-(4g_{i,j}+g_{i+1,j}+g_{i-1,j}+g_{i,j+1}+g_{i,j-1})z_{i,j}\end{pmatrix}. \tag{4.5.12}$$

Of course when we are using the Tichonov diffusivity where $g_{i,j}=1:\ \forall i,j$, we get the standard Laplacian discretization:

$$Smoothness\_term_{Euler-Lagrange}= \\ -\frac{\varphi}{2}\cdot\left(2z_{i+1,j}+2z_{i-1,j}+2z_{i,j+1}+2z_{i,j-1}-8z_{i,j}\right)= \\ -\varphi\cdot(z_{i+1,j}-2z_{i,j}+z_{i-1,j}+z_{i,j+1}-2z_{i,j}+z_{i,j+1}). \tag{4.5.13}$$

Now, using the discrete versions of the data and smoothness terms of the Euler – Lagrange equation (4.5.6), (4.5.9) and taking all the terms $z_{i,j}$ in its smoothness term from the next time step we can rewrite the discrete version of the equation (4.5.5):

$$\frac{z^*_{i,j}-z_{i,j}}{\tau}=-g_{data}(Forma_{i,j})\cdot$$
$$\left(\begin{array}{l}\Theta\cdot(I_{2u})_{i,j}\dfrac{\alpha_u\Delta t_x}{z^2_{i,j}}\left((I_1)_{i,j}-(I_2)_{i,j}-\dfrac{\alpha_u\Delta t_x}{z_{i,j}}(I_{2u})_{i,j}\right)+\\ (1-\Theta)\cdot(I_{2uu})_{i,j}\dfrac{\alpha_u\Delta t_x}{z^2_{i,j}}\left((I_{1u})_{i,j}-(I_{2u})_{i,j}-\dfrac{\alpha_u\Delta t_x}{z_{i,j}}(I_{2uu})_{i,j}\right)+\\ (1-\Theta)\cdot(I_{2vu})_{i,j}\dfrac{\alpha_u\Delta t_x}{z^2_{i,j}}\left((I_{1v})_{i,j}-(I_{2v})_{i,j}-\dfrac{\alpha_u\Delta t_x}{z_{i,j}}(I_{2vu})_{i,j}\right)\end{array}\right)+$$
$$\frac{\varphi}{2}\cdot\left(\begin{array}{l}(g_{i+1,j}+g_{i,j})z_{i+1,j}+(g_{i,j}+g_{i-1,j})z_{i-1,j}+(g_{i,j+1}+g_{i,j})z_{i,j+1}\\ +(g_{i,j}+g_{i,j-1})z_{i,j-1}-(4g_{i,j}+g_{i+1,j}+g_{i-1,j}+g_{i,j+1}+g_{i,j-1})z^*_{i,j}\end{array}\right). \tag{4.5.14}$$

Rearranging all the $z^*_{i,j}$ to the left part of the equation (4.5.14) and all the $z_{i,j}$ to the right part we achieve the modified explicit time marching numerical scheme:

$$z^*_{i,j}=\frac{z_{i,j}+\tau\cdot\left(\begin{array}{l}-g_{data}(Forma_{i,j})\cdot\\ \left(\begin{array}{l}\Theta\cdot(I_{2u})_{i,j}\dfrac{\alpha_u\Delta t_x}{z^2_{i,j}}\left((I_1)_{i,j}-(I_2)_{i,j}-\dfrac{\alpha_u\Delta t_x}{z_{i,j}}(I_{2u})_{i,j}\right)+\\ (1-\Theta)\cdot(I_{2uu})_{i,j}\dfrac{\alpha_u\Delta t_x}{z^2_{i,j}}\left((I_{1u})_{i,j}-(I_{2u})_{i,j}-\dfrac{\alpha_u\Delta t_x}{z_{i,j}}(I_{2uu})_{i,j}\right)+\\ (1-\Theta)\cdot(I_{2vu})_{i,j}\dfrac{\alpha_u\Delta t_x}{z^2_{i,j}}\left((I_{1v})_{i,j}-(I_{2v})_{i,j}-\dfrac{\alpha_u\Delta t_x}{z_{i,j}}(I_{2vu})_{i,j}\right)\end{array}\right)+\\ \dfrac{\varphi}{2}\cdot\left(\begin{array}{l}(g_{i+1,j}+g_{i,j})z_{i+1,j}+(g_{i,j}+g_{i-1,j})z_{i-1,j}+\\ (g_{i,j+1}+g_{i,j})z_{i,j+1}+(g_{i,j}+g_{i,j-1})z_{i,j-1}\end{array}\right)\end{array}\right)}{1+\dfrac{\varphi\tau}{2}\cdot(4g_{i,j}+g_{i+1,j}+g_{i-1,j}+g_{i,j+1}+g_{i,j-1})}. \tag{4.5.15}$$

The analogous semi-implicit time marching numerical scheme for the non linear Euler – Lagrange equation but for the linear interpolation method could be achieved from the equation (4.5.15) using the table 4.3. Having in mind (4.3.1) we can write:

$$z_{i,j}^{*} = \frac{z_{i,j} + \tau \cdot \left(\begin{array}{l} -g_{data}(Forma_{i,j}) \cdot \\ \left(\begin{array}{l} \Theta \cdot (I_{2u})_{i+A,j} \dfrac{\alpha_u \Delta t_x}{z_{i,j}^2} \cdot \\ \left((I_1)_{i,j} - (1-|b|) \cdot (I_2)_{i+A,j} - |b| \cdot (I_{2u})_{i+A+sign(b),j} \right) + \\ (1-\Theta) \cdot (I_{2uu})_{i+A,j} \dfrac{\alpha_u \Delta t_x}{z_{i,j}^2} \cdot \\ \left((I_{1u})_{i,j} - (1-|b|) \cdot (I_{2u})_{i+A,j} - |b| \cdot (I_{2uu})_{i+A+sign(b),j} \right) + \\ (1-\Theta) \cdot (I_{2vu})_{i+A,j} \dfrac{\alpha_u \Delta t_x}{z_{i,j}^2} \cdot \\ \left((I_{1v})_{i,j} - (1-|b|) \cdot (I_{2v})_{i+A,j} - |b| \cdot (I_{2vu})_{i+A+sign(b),j} \right) \end{array} \right) + \\ \dfrac{\varphi}{2} \cdot \left(\begin{array}{l} (g_{i+1,j} + g_{i,j}) z_{i+1,j} + (g_{i,j} + g_{i-1,j}) z_{i-1,j} + \\ (g_{i,j+1} + g_{i,j}) z_{i,j+1} + (g_{i,j} + g_{i,j-1}) z_{i,j-1} \end{array} \right) \end{array} \right)}{1 + \dfrac{\varphi \tau}{2} \cdot (4 g_{i,j} + g_{i+1,j} + g_{i-1,j} + g_{i,j+1} + g_{i,j-1})}. \quad (4.5.16)$$

In this notation the term $Forma_{i,j}$ will accept the look:

$$\begin{array}{l} Forma_{i,j} = \Theta \cdot \left((I_1)_{i,j} - (1-|b|) \cdot (I_2)_{i+A,j} - |b| \cdot (I_{2u})_{i+A+sign(b),j} \right)^2 + \\ (1-\Theta) \cdot \left((I_{1u})_{i,j} - (1-|b|) \cdot (I_{2u})_{i+A,j} - |b| \cdot (I_{2uu})_{i+A+sign(b),j} \right)^2 + \\ (1-\Theta) \cdot \left((I_{1v})_{i,j} - (1-|b|) \cdot (I_{2v})_{i+A,j} - |b| \cdot (I_{2vu})_{i+A+sign(b),j} \right)^2. \end{array} \quad (4.5.17)$$

The discretized smoothness term of the Euler – Lagrange equation was not changed at all, we only corrected the data term of the Euler – Lagrange equation. The process of discretizing and building the numerical scheme for the linear interpolation method is identical to the same process for the first order Taylor expansion.

4.6 SOR numerical scheme

When we are going to discretize a linear differential equation, it is possible for its numerical scheme use the Successive Over Relaxation technique or SOR scheme, which speeds up the convergence of the numerical algorithm greatly. We will focus on the use of an iterative method which lends itself to the opportunity to apply Cartesian topology. The simplest of iterative techniques is the Jacobi scheme and we will start the derivation of our numerical scheme according to it. While the Jacobi iteration scheme is very simple and parallelizable its slow convergent rate however renders it impractical for any "real world" applications. One way to speed up the convergent rate would be to

"over predict" the new solution by linear extrapolation. This is the main idea of the Successive Over Relaxation scheme.

Now let us build a linear numerical scheme for our problem. First of all let us write the energy functional with the data term built on grey-value constancy assumption and gradient constancy assumption with help of regularization and the smoothness term built with the help of a regularization function:

$$E(z) = \iint_{\Omega} \Psi \begin{pmatrix} \Theta \cdot \left| I_1(u,v) - I_2(u+z',v) \right|^2 + \\ (1-\Theta) \cdot \left| \nabla I_1(u,v) - \nabla I_2(u+z',v) \right|^2 \end{pmatrix} + \varphi \cdot \Psi(\left|\nabla z'\right|^2) du dv ; \qquad (4.6.1)$$

like before in (4.5.2) we decompose the second component of the data term:

$$\begin{aligned} &\left| \nabla I_1(u,v) - \nabla I_2(u+z',v) \right|^2 = \\ &\left(I_{1u}(u,v) - I_{2u}(u+z',v) \right)^2 + \left(I_{1v}(u,v) - I_{2v}(u+z',v) \right)^2 . \end{aligned} \qquad (4.6.2)$$

Using the formula (4.6.2) we can rewrite the energy functional (4.6.1) in the following form:

$$E(z) = \iint_{\Omega} \Psi \begin{pmatrix} \Theta \cdot \left(I_1(u,v) - I_2(u+z',v) \right)^2 + \\ (1-\Theta) \cdot \left(I_{1u}(u,v) - I_{2u}(u+z',v) \right)^2 + \\ (1-\Theta) \cdot \left(I_{1v}(u,v) - I_{2v}(u+z',v) \right)^2 \end{pmatrix} + \varphi \cdot \Psi(\left|\nabla z'\right|^2) du dv . \qquad (4.6.3)$$

Now let us apply the first-order Taylor expansion technique and rewrite the energy functional (4.6.3) omitting the function argument indexing:

$$E(z) = \iint_{\Omega} \Psi \begin{pmatrix} \Theta \cdot \left(I_1 - I_2 - z' I_{2u} \right)^2 + \\ (1-\Theta) \cdot \left(I_{1u} - I_{2u} - z' I_{2uu} \right)^2 + \\ (1-\Theta) \cdot \left(I_{1v} - I_{2v} - z' I_{2vu} \right)^2 \end{pmatrix} + \varphi \cdot \Psi(\left|\nabla z'\right|^2) du dv . \qquad (4.6.4)$$

Then we are ready to derive the Euler – Lagrange equation:

$$\begin{aligned} &\Psi' \begin{pmatrix} \Theta \cdot \left(I_1 - I_2 - z' I_{2u} \right)^2 + \\ (1-\Theta) \cdot \left(I_{1u} - I_{2u} - z' I_{2uu} \right)^2 + \\ (1-\Theta) \cdot \left(I_{1v} - I_{2v} - z' I_{2vu} \right)^2 \end{pmatrix} \cdot \\ &\begin{pmatrix} \Theta \cdot I_{2u} \left(I_1 - I_2 - z' I_{2u} \right) + \\ (1-\Theta) \cdot I_{2uu} \left(I_{1u} - I_{2u} - z' I_{2uu} \right) + \\ (1-\Theta) \cdot I_{2vu} \left(I_{1v} - I_{2v} - z' I_{2vu} \right) \end{pmatrix} + \\ &\varphi \cdot div\left(\Psi'(\left|\nabla z'\right|^2) \cdot \nabla z \right) = 0. \end{aligned} \qquad (4.6.5)$$

Like in the previous paragraph we will consider the data and smoothness terms of the Euler – Lagrange equation separately. We will use the same notation (4.5.6) from the paragraph 4.5 and the same discrete smoothness term of the Euler – Lagrange equation (4.5.12). Discretizing the data term of the Euler – Lagrange equation we will use the designation similar to (4.5.8):

$$\begin{aligned} &Forma'_{i,j} = \Theta \cdot \left(\left(I_1\right)_{i,j} - \left(I_2\right)_{i,j} - z'_{i,j} \left(I_{2u}\right)_{i,j} \right)^2 + \\ &(1-\Theta) \cdot \left(\left(I_{1u}\right)_{i,j} - \left(I_{2u}\right)_{i,j} - z'_{i,j} \left(I_{2uu}\right)_{i,j} \right)^2 + \\ &(1-\Theta) \cdot \left(\left(I_{1v}\right)_{i,j} - \left(I_{2v}\right)_{i,j} - z'_{i,j} \left(I_{2vu}\right)_{i,j} \right)^2. \end{aligned} \tag{4.6.6}$$

In such a way let us write down the discretized data term of the Euler – Lagrange equation:

$$\begin{aligned} &Data\_term_{Euler-Lagrange} = g_{data}(Forma'_{i,j}) \cdot \\ &\left(\begin{aligned} &\Theta \cdot \left(I_{2u}\right)_{i,j} \left(\left(I_1\right)_{i,j} - \left(I_2\right)_{i,j} - z'_{i,j} \left(I_{2u}\right)_{i,j} \right) + \\ &(1-\Theta) \cdot \left(I_{2uu}\right)_{i,j} \left(\left(I_{1u}\right)_{i,j} - \left(I_{2u}\right)_{i,j} - z'_{i,j} \left(I_{2uu}\right)_{i,j} \right) + \\ &(1-\Theta) \cdot \left(I_{2vu}\right)_{i,j} \left(\left(I_{1v}\right)_{i,j} - \left(I_{2v}\right)_{i,j} - z'_{i,j} \left(I_{2vu}\right)_{i,j} \right) \end{aligned} \right). \end{aligned} \tag{4.6.7}$$

Now, using the discrete versions of the data and smoothness terms of the Euler – Lagrange equation (4.6.7), (4.5.12) we can rewrite the discrete version of the Euler – Lagrange equation (4.6.5):

$$\begin{aligned} &0 = g_{data}(Forma'_{i,j}) \cdot \\ &\left(\begin{aligned} &\Theta \cdot \left(I_{2u}\right)_{i,j} \left(\left(I_1\right)_{i,j} - \left(I_2\right)_{i,j} - z'_{i,j} \left(I_{2u}\right)_{i,j} \right) + \\ &(1-\Theta) \cdot \left(I_{2uu}\right)_{i,j} \left(\left(I_{1u}\right)_{i,j} - \left(I_{2u}\right)_{i,j} - z'_{i,j} \left(I_{2uu}\right)_{i,j} \right) + \\ &(1-\Theta) \cdot \left(I_{2vu}\right)_{i,j} \left(\left(I_{1v}\right)_{i,j} - \left(I_{2v}\right)_{i,j} - z'_{i,j} \left(I_{2vu}\right)_{i,j} \right) \end{aligned} \right) + \\ &\frac{\varphi}{2} \cdot \left(\begin{aligned} &(g_{i+1,j} + g_{i,j}) z'_{i+1,j} + (g_{i,j} + g_{i-1,j}) z'_{i-1,j} + (g_{i,j+1} + g_{i,j}) z'_{i,j+1} \\ &+ (g_{i,j} + g_{i,j-1}) z'_{i,j-1} - (4g_{i,j} + g_{i+1,j} + g_{i-1,j} + g_{i,j+1} + g_{i,j-1}) z'_{i,j} \end{aligned} \right). \end{aligned} \tag{4.6.8}$$

Having in mind notation (4.5.7) we will build Jacobi iteration scheme by choosing all the terms $z'_{i,j}$ from the discretized Euler – Lagrange equation except the argument of $g_{data}()$ function from the next iteration step:

$$
\begin{aligned}
0 = {} & g_{data}(Forma'_{i,j}) \cdot \\
& \begin{pmatrix}
\Theta \cdot (I_{2u})_{i,j} \left((I_1)_{i,j} - (I_2)_{i,j} \right) - \Theta \cdot z'^*_{i,j} (I_{2u})^2_{i,j} + \\
(1-\Theta) \cdot (I_{2uu})_{i,j} \left((I_{1u})_{i,j} - (I_{2u})_{i,j} \right) - (1-\Theta) \cdot z'^*_{i,j} (I_{2uu})^2_{i,j} + \\
(1-\Theta) \cdot (I_{2vu})_{i,j} \left((I_{1v})_{i,j} - (I_{2v})_{i,j} \right) - (1-\Theta) \cdot z'^*_{i,j} (I_{2vu})^2_{i,j}
\end{pmatrix} + \\
& \frac{\varphi}{2} \cdot \begin{pmatrix}
(g_{i+1,j} + g_{i,j}) z'_{i+1,j} + (g_{i,j} + g_{i-1,j}) z'_{i-1,j} + \\
(g_{i,j+1} + g_{i,j}) z'_{i,j+1} + (g_{i,j} + g_{i,j-1}) z'_{i,j-1}
\end{pmatrix} - \\
& \frac{\varphi}{2} \cdot (4 g_{i,j} + g_{i+1,j} + g_{i-1,j} + g_{i,j+1} + g_{i,j-1}) z'^*_{i,j}
\end{aligned}
$$

and rearrange them to the left side of the equation (4.6.8):

$$
\begin{aligned}
z'^*_{i,j} = {} & \frac{\dfrac{\varphi}{2} \cdot \begin{pmatrix}
(g_{i+1,j} + g_{i,j}) z'_{i+1,j} + (g_{i,j} + g_{i-1,j}) z'_{i-1,j} + \\
(g_{i,j+1} + g_{i,j}) z'_{i,j+1} + (g_{i,j} + g_{i,j-1}) z'_{i,j-1}
\end{pmatrix}}{} \\
& g_{data}(Forma'_{i,j}) \cdot \begin{pmatrix}
\Theta \cdot (I_{2u})^2_{i,j} + \\
(1-\Theta) \cdot (I_{2uu})^2_{i,j} + \\
(1-\Theta) \cdot (I_{2vu})^2_{i,j}
\end{pmatrix} + \\
& \frac{\varphi}{2} \cdot (4 g_{i,j} + g_{i+1,j} + g_{i-1,j} + g_{i,j+1} + g_{i,j-1}) +
\end{aligned}
\qquad (4.6.9)
$$

This time it will not be so easy to derive the analogues Jacobi iteration scheme for the linear interpolation method because we have to split z' onto A' and b' and take b' from the next time step, leaving A' at current time step. We should continue step by step, beginning from the (4.6.8). According to the method of the linear interpolation we chose two numbers A' and b' such that $A'+b' = z'_{i,j}$ and described in paragraph 4.3. Now, using the table 4.3, and having in mind (4.3.1) we can rewrite the equation (4.6.8):

$$
\begin{aligned}
0 = {} & g_{data}(Forma'_{i,j}) \cdot \\
& \begin{pmatrix}
\Theta \cdot (I_{2u})_{i+A,j} \left((I_1)_{i,j} - (1-|b'|) \cdot (I_2)_{i+A,j} - |b'| \cdot (I_2)_{i+A+sign(b),j} \right) + \\
(1-\Theta) \cdot (I_{2uu})_{i+A,j} \left((I_{1u})_{i,j} - (1-|b'|) \cdot (I_{2u})_{i+A,j} - |b'| \cdot (I_{2u})_{i+A+sign(b),j} \right) + \\
(1-\Theta) \cdot (I_{2vu})_{i+A,j} \left((I_{1v})_{i,j} - (1-|b'|) \cdot (I_{2v})_{i+A,j} - |b'| \cdot (I_{2v})_{i+A+sign(b),j} \right)
\end{pmatrix} + \\
& \frac{\varphi}{2} \cdot \begin{pmatrix}
(g_{i+1,j} + g_{i,j}) z'_{i+1,j} + (g_{i,j} + g_{i-1,j}) z'_{i-1,j} + (g_{i,j+1} + g_{i,j}) z'_{i,j+1} \\
+ (g_{i,j} + g_{i,j-1}) z'_{i,j-1} - (4 g_{i,j} + g_{i+1,j} + g_{i-1,j} + g_{i,j+1} + g_{i,j-1})(A'+b')
\end{pmatrix}.
\end{aligned}
\qquad (4.6.10)
$$

Then we build Jacobi iteration scheme by choosing all the terms b' from the discretized Euler – Lagrange equation except the argument of $g_{data}()$ function from the next iteration step:

$$\begin{aligned}
&0 = g_{data}(Forma'_{i,j}) \cdot \\
&\left(\begin{array}{l} \Theta \cdot (I_{2u})_{i+A,j} \left((I_1)_{i,j} - (I_2)_{i+A,j} \right) - \Theta \cdot b^{*\prime} \cdot (I_{2u})^2_{i+A,j} + \\ (1-\Theta) \cdot (I_{2uu})_{i+A,j} \left((I_{1u})_{i,j} - (I_{2u})_{i+A,j} \right) - (1-\Theta) \cdot b^{*\prime} \cdot (I_{2uu})^2_{i+A,j} + \\ (1-\Theta) \cdot (I_{2vu})_{i+A,j} \left((I_{1v})_{i,j} - (I_{2v})_{i+A,j} \right) - (1-\Theta) \cdot b^{*\prime} \cdot (I_{2vu})^2_{i+A,j} \end{array} \right) + \\
&\frac{\varphi}{2} \cdot \left(\begin{array}{l} (g_{i+1,j} + g_{i,j}) z'_{i+1,j} + (g_{i,j} + g_{i-1,j}) z'_{i-1,j} + (g_{i,j+1} + g_{i,j}) z'_{i,j+1} \\ + (g_{i,j} + g_{i,j-1}) z'_{i,j-1} - (4g_{i,j} + g_{i+1,j} + g_{i-1,j} + g_{i,j+1} + g_{i,j-1}) A' \end{array} \right) - \\
&\frac{\varphi b^{*\prime}}{2} \cdot (4g_{i,j} + g_{i+1,j} + g_{i-1,j} + g_{i,j+1} + g_{i,j-1}).
\end{aligned} \tag{4.6.11}$$

Deriving the equation (4.6.11) we used the proof of the theorem 4.1. Now let us rearrange all the b' to the left side of the equation:

$$\begin{aligned}
b^{*\prime} = &\frac{\begin{array}{l} \frac{\varphi}{2} \cdot \left(\begin{array}{l} (g_{i+1,j} + g_{i,j}) z'_{i+1,j} + (g_{i,j} + g_{i-1,j}) z'_{i-1,j} + (g_{i,j+1} + g_{i,j}) z'_{i,j+1} \\ + (g_{i,j} + g_{i,j-1}) z'_{i,j-1} - (4g_{i,j} + g_{i+1,j} + g_{i-1,j} + g_{i,j+1} + g_{i,j-1}) A' \end{array} \right) \\ g_{data}(Forma'_{i,j}) \cdot \left(\begin{array}{l} \Theta \cdot (I_{2u})_{i+A,j} \left((I_1)_{i,j} - (I_2)_{i+A,j} \right) + \\ (1-\Theta) \cdot (I_{2uu})_{i+A,j} \left((I_{1u})_{i,j} - (I_{2u})_{i+A,j} \right) + \\ (1-\Theta) \cdot (I_{2vu})_{i+A,j} \left((I_{1v})_{i,j} - (I_{2v})_{i+A,j} \right) \end{array} \right) + \end{array}}{\begin{array}{l} g_{data}(Forma'_{i,j}) \cdot \left(\begin{array}{l} \Theta \cdot (I_{2u})^2_{i+A,j} + \\ (1-\Theta) \cdot (I_{2uu})^2_{i+A,j} + \\ (1-\Theta) \cdot (I_{2vu})^2_{i+A,j} \end{array} \right) + \\ \frac{\varphi}{2} \cdot (4g_{i,j} + g_{i+1,j} + g_{i-1,j} + g_{i,j+1} + g_{i,j-1}) \end{array}}.
\end{aligned} \tag{4.6.12}$$

In this notation the term $Forma'_{i,j}$ will accept the look:

$$\begin{aligned}
&Forma'_{i,j} = \Theta \cdot \left((I_1)_{i,j} - (1 - |b'|) \cdot (I_2)_{i+A,j} - |b'| \cdot (I_2)_{i+A+sign(A),j} \right)^2 + \\
&(1-\Theta) \cdot \left((I_{1u})_{i,j} - (1 - |b'|) \cdot (I_{2u})_{i+A,j} - |b'| \cdot (I_{2uu})_{i+A+sign(A),j} \right)^2 + \\
&(1-\Theta) \cdot \left((I_{1v})_{i,j} - (1 - |b'|) \cdot (I_{2v})_{i+A,j} - |b'| \cdot (I_{2vu})_{i+A+sign(A),j} \right)^2.
\end{aligned} \tag{4.6.13}$$

And the new value of the disparity field we can find using the following formula:

$$z^{*\prime} = A' + b^{*\prime}. \tag{4.6.14}$$

Pay your attention that the value of $b^{*\prime}$ is not obliged to satisfy the criteria: $|b^{*\prime}| < 1$.

To come now to the SOR numerical scheme we make the following steps:

1. Make initial guess for $z'_{i,j}$ at all interior points (i, j).
2. Define a scalar w_n $(0 < w_n < 2)$.
3. Apply equation (4.6.9) or (4.6.12) to all interior points (i, j) and call it $\bar{z}'_{i,j}$.
4. $z^{*\prime}_{i,j} = w_n \bar{z}'_{i,j} + (1 - w_n) z'_{i,j}$.
5. Stop if prescribed convergence threshold is reached, otherwise continue on next step.
6. $z'_{i,j} = z^{*\prime}_{i,j}$.
7. Go to Step 2.

Note in the above that setting $w_n = 1$ recovers the Jacobi scheme while $w_n < 1$ underrelaxes the solution. Ideally, the choice of w_n would be such that it provides the optimal rate of convergence and is not restricted to a fixed constant. As a matter of fact, an effective choice of w_n, known as the Chebyshev acceleration, is defined as [WMW07]:

$$w_n = \begin{cases} 0 & for \quad n = 0; \\ \dfrac{2}{2 - \rho^2} & for \quad n = 1; \\ \dfrac{4}{4 - \rho^2 w_1} & for \quad n = 2; \\ \dfrac{4}{4 - \rho^2 w_{q-1}} & for \quad n = q > 2. \end{cases} \tag{4.6.15}$$

Note that the numerical scheme (4.6.12), (4.6.13) with $A' = 0$ and $b' = z'$ becomes identical to the numerical scheme (4.6.9), (4.6.6). The same we observe with the time-marching numerical scheme: the scheme (4.5.16), (4.6.17) with $A = 0$ and $b = z$ turns into the scheme (4.5.15), (4.5.8).

4.7 Coarse-to-fine levels technique

The coarse-to-fine strategy follows two main aims. The firs aim is to solve the problem of the multiple minima in the energy functional and the problem of avoidance of local minima during the iteration process. And the second aim is to tackle the problem of large displacements. First we shell tell how to build the coarse instances of a picture and discuss the strategies of implementation and after that we shell consider the aims of the coarse-to-fine levels technique in detail.

For the purpose of the implementation technique the literature offers us two different strategies: the *scale-space focusing method* that considers the problem at different smoothness scales, keeping the picture's resolution unchanged; and the *multiresolution technique* that considers the problem at different resolution levels [MP98]. In this paper

we will use the multiresolution technique since it is much more efficient from the computational point of view. The only lack of this technique that we should take care about aliasing problem during downscaling a picture. But this lack could be easily eliminated with help of the bilinear interpolation.

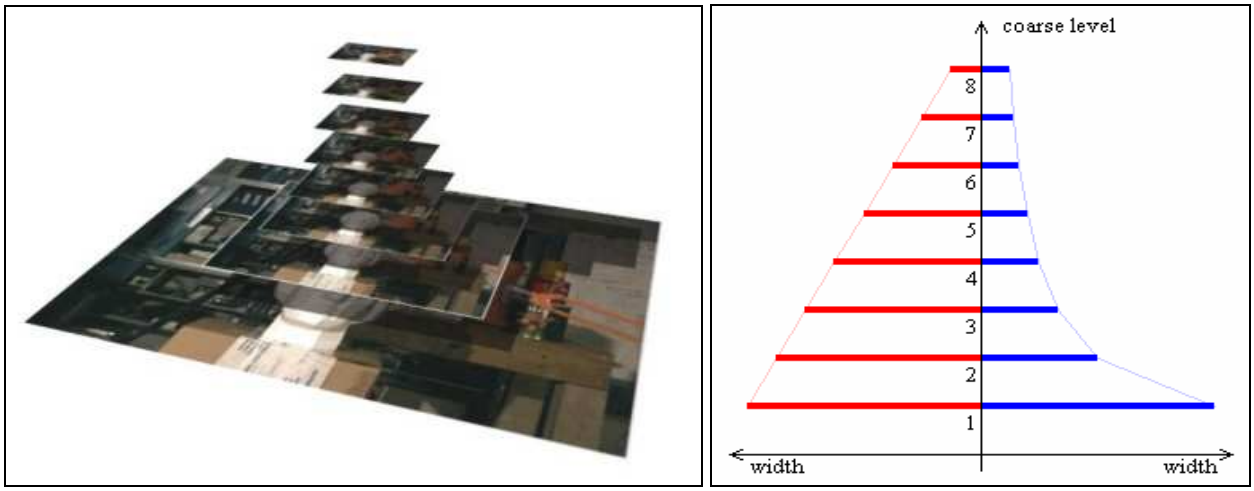

Figure 4.6: Multiresolution coarse levels pyramids: **Left:** 2D example; **Right:** Comparison of the arithmetic pyramid (red) and the geometric pyramid (blue).

At the figure 4.6 we can see an example of a coarse levels pyramid, built from a single image and its downscaled instances. At the top of this pyramid the coarsest level is situated and at the bottom – the fine level, the original picture. If we have N coarse levels, then the fine level will be the first coarse level and the coarsest one will be the N-th coarse level. The depth-map reconstruction process becomes with this technique an iteration process: we reconstruct a depth-map on a coarse level image and then use this reconstructed depth-map as initial map for the next coarse level. As initial map for the coarsest level we take a plane which has constant depth values for all the objects in scene, which is better satisfies the Euler – Lagrange equation. This plane could be chosen by the simple looking though the suitable depth values.

When we have the original picture, to build the pyramid, illustrated in figure 4.6, it is enough to choose the number of coarse levels and the dimensions of the picture in the coarsest level. It is very important to get the first depth-map from the coarsest level, because it will be a fundament for all the following computations. That's why the best choice of the dimensions of picture at the coarsest level is such choice where the largest displacement in optic flow will be sufficiently small.

Let the width of the original picture will be designated through $width_1$ and the width of the picture at the coarsest level – $width_N$. Then the resolution of the picture at coarse level i could be found with the help of the following formulae:

$$\begin{aligned} width_i &= width_1 - (i-1)\frac{width_1 - width_N}{N-1}; \\ height_i &= height_1 - (i-1)\frac{height_1 - height_N}{N-1}. \end{aligned} \qquad (4.7.1)$$

As we have mentioned at the very coarse levels we build the fundament for all the subsequent calculations. Another approach for calculating the picture dimensions on a coarse level, which improves our fundament, can be expressed in the following two formulae for picture dimensions:

$$\begin{aligned} width_i &= \frac{width_1}{i}; \\ height_i &= \frac{height_1}{i}. \end{aligned} \tag{4.7.2}$$

This approach guaranties that the picture's resolution at top layers of the pyramid will increase slowly, which consequently allows us to gain better results at the beginning of iteration coarse-to-fine process. And at the bottom layers of the pyramid we have large resolution increase, what compensates the small speed at the beginning. The coarse levels pyramid, built by the rule of formulae (4.7.1) we will call *arithmetic pyramid* and the coarse levels pyramid, built with help of formulae (4.7.2) – *geometric pyramid* (figure 4.6, right picture).

The number of layers N should be chosen in such a way, that the width increment be smaller or equal to the width of the picture at the coarsest level. It will guarantee that the displacement from coarse level to coarse level will not increase more than in two pixels.

$$\frac{width_1 - width_N}{N-1} \le width_N, \tag{4.7.3}$$

or in other notation N should satisfy the criteria:

$$N \ge \frac{width_1}{width_N}. \tag{4.7.4}$$

The coarse-to-fine level technique fights the problem of the variational method getting stuck in local minima. As we know, the main idea of variational methods is to find unknown function which minimizes energy functional. Since as usual an the energy functional is not convex and may have multiple minima but only one global minimum, the initialization decides to which minimum the iteration process converges. And as usual it is a local minimum, not the desired global one. Using coarse-to-fine levels strategy makes the local minima with sufficiently small spatial extent vanish at coarser scales and can thus be avoided. How it works we can observe at figure 4.7.

When we have large displacements (more than 4 pixels) in a stereo image, none of the methods described in this chapter could handle with them, starting from a constant depth-map as initial data. The coarse-to-file levels technique starts with the variational process on a coarse instance of the original picture, where the displacement much smaller (less then 4 pixels). Coming from a coarser level to a finer level, we have the initial depth-map, calculated from the previous step, and thus we just step by step increase the displacement and recalculate the depth-map with better accuracy. As we

know the linear interpolation method precedes any large displacement, but the first order Taylor expansion method – does not. In order to use the technique of coarse-to-fine levels with the method of first order Taylor expansion we ought to use moreover the warping methods.

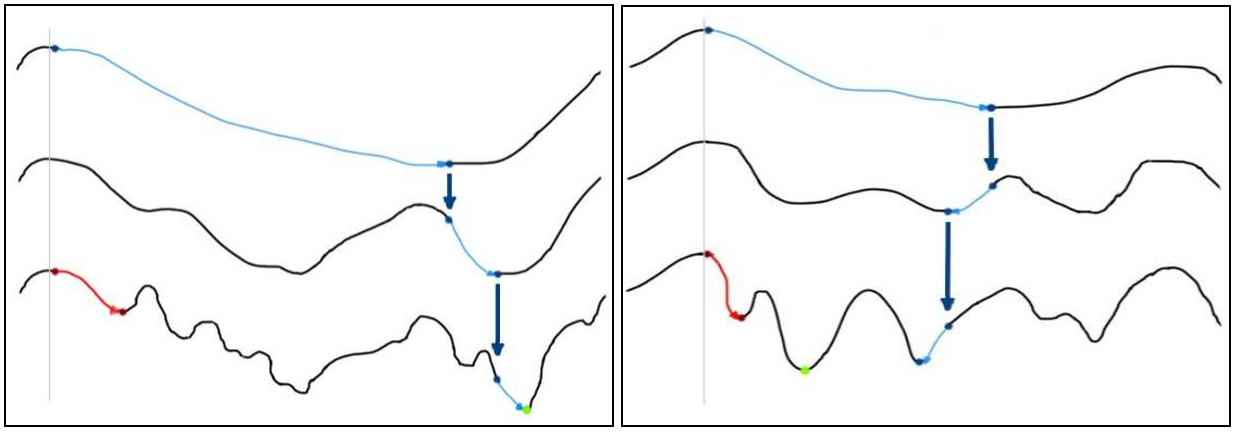

Figure 4.7: Minimization using coarse-to-file levels strategy (blue) and without (red) to the global minimum (green): **Left:** Global minimum found; **Right:** Useful local minimum found.

Warping denotes the distortion of the image sequence which is required for the compensation for the already computed motion. So far this technique has only been justified on an *algorithmic basis*: In general, it was argued that it makes sense to embed optic flow approaches for small displacements into a coarse-to-fine framework, since large displacements become smaller at coarser levels and thus allow for an accurate estimation with linearized model assumptions. This, of course, is true. However, as we have seen, this warping strategy can also be derived as hierarchical fixed point iteration for minimising the energy functional of a variational approach for large displacements, i.e. for the energy functional based on constancy assumptions without linearization. This in turn, provides a *theoretical justification* of the warping technique [BBPW04].

4.8 Summary

In this chapter we have considered four methods for the depth-map reconstruction: the depth-driven method, the disparity-driven method, the method of the first order Taylor expansion and the method of linear interpolation. We discussed the theoretical properties of these methods and proved them on certain experiments. The disparity-driven method gives better results then the depth-driven: it leads to a simpler and faster converging numerical scheme and preserves discontinuities at far-situated objects. The linear interpolation method is an extension of the first order Taylor expansion method and can deal with more then twice bigger displacements than its "small brother". Moreover the linear interpolation method does not require awkward warping methods. Thus in future work of extending our model to multiple cameras (more than 2), we will use the disparity-driven method with the linear interpolation method for linearization, coarse-to-fine levels technique to avoid local minima and handle large displacements.

Chapter 5

Multi – View Depth-Map Reconstruction

This chapter extends all the theory for two cameras to the case of multiple cameras. We discuss the multi – view model and then in the second section we derive a numerical scheme for it. Thereby we consider only the method of linear interpolation for the disparity approach as this method gives the most reliable results.

5.1 Multiple data terms

Till this moment we have being considering the depth-map reconstructions methods which are based on information from only two cameras. By other words, having any two pictures from different cameras, we can build a data term for an energy functional. In the case of N cameras, choosing any two of them, it is possible to build $(N-1)!$ different data terms. The main idea of the extension of our model for 2 cameras to the model for N cameras is to sum all the possible data terms together and thus constitute one monolith data term. As illustration, let us rewrite the energy functional (2.2.8) for case of multiple cameras:

$$E(z) = \iint_{\Omega} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left| I_i(u_i, v_i) - I_j(u_j, v_j) \right|^2 + \varphi \cdot \left| \nabla z \right|^2 du dv . \qquad (5.1.1)$$

In this paper we make some simplifications. First of all we do not consider all the $(N-1)!$ possibilities. Instead of that we chose from N cameras one *base* camera and

consider all the possible pairs of cameras with the base one. It will give us only $(N-1)$ different data terms, what will reduce computational efforts greatly.

Another simplification is that all the cameras are equidistantly situated at one horizontal line and we deal with odd number of cameras, i.e. $N = 2n+1$. So we can chose as the base camera the centre one with index $n+1$. Using the formula (3.1.10) we can rewrite the energy functional (5.1.1):

$$E(z) = \iint_{\Omega} \sum_{k=1}^{n} \left(\begin{array}{l} \left| I_{n+1}(u,v) - I_k(u-(n+1-k)\frac{\alpha_u \Delta t_x}{z}, v) \right|^2 + \\ \left| I_{n+1}(u,v) - I_{2n+2-k}(u+(n+1-k)\frac{\alpha_u \Delta t_x}{z}, v) \right|^2 \end{array} \right) + \varphi \cdot |\nabla z|^2 \, dudv \,. \qquad (5.1.2)$$

Here we start summing the data terms from the outer cameras and finish with the centre cameras (figure 5.1).

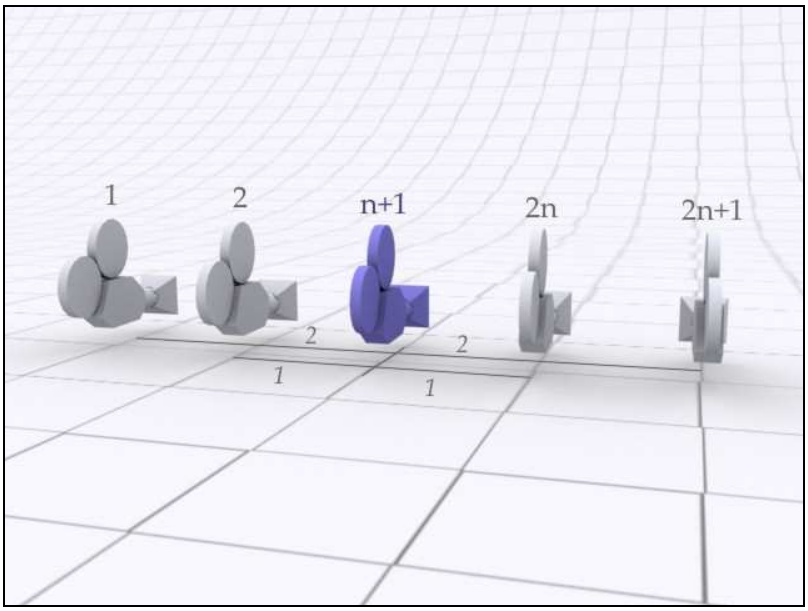

Figure 5.1: Multi – view cameras' position example.

At the figure 5.1 we observe the case of $n = 2$: we have two left, two right and one centre camera. According to the formula (5.1.2), first we consider the disparity between the 2-ns, 4-th and centre cameras and then between 1-st, 5-th and centre cameras, where the disparity is doubled.

5.2 Numerical scheme

The formulae in this paragraph will we very long and to save some paper we introduce the following notation:

$$\partial_{\arg} I_i^u \equiv \frac{\partial I_i(u,v)}{\partial \arg},\ \arg \in \{u,v\}\,. \tag{5.2.1}$$

Let us start with the energy functional for the disparity-driven approach:

$$E(z) = \iint_{\Omega} \sum_{k=1}^{n} \Psi_d \left(\begin{array}{l} \Theta \cdot \left(\begin{array}{l} \left| I_{n+1}^u - I_k^{u-(n+1-k)z} \right|^2 + \\ \left| I_{n+1}^u - I_{2n+2-k}^{u+(n+1-k)z} \right|^2 \end{array} \right) + \\ (1-\Theta) \cdot \left(\begin{array}{l} \left| \nabla I_{n+1}^u - \nabla I_k^{u-(n+1-k)z} \right|^2 + \\ \left| \nabla I_{n+1}^u - \nabla I_{2n+2-k}^{u+(n+1-k)z} \right|^2 \end{array} \right) \end{array} \right) + \varphi \cdot \Psi_s(\left|\nabla z\right|^2) du dv\,; \tag{5.2.2}$$

as usual we open the gradient notation and receive the following expression:

$$E(z) = \iint_{\Omega} \sum_{k=1}^{n} \Psi_d \left(\begin{array}{l} \Theta \cdot \left(\begin{array}{l} \left| I_{n+1}^u - I_k^{u-(n+1-k)z} \right|^2 + \\ \left| I_{n+1}^u - I_{2n+2-k}^{u+(n+1-k)z} \right|^2 \end{array} \right) + \\ (1-\Theta) \cdot \left(\begin{array}{l} \left| \partial_u I_{n+1}^u - \partial_u I_k^{u-(n+1-k)z} \right|^2 + \\ \left| \partial_u I_{n+1}^u - \partial_u I_{2n+2-k}^{u+(n+1-k)z} \right|^2 \end{array} \right) + \\ (1-\Theta) \cdot \left(\begin{array}{l} \left| \partial_v I_{n+1}^u - \partial_v I_k^{u-(n+1-k)z} \right|^2 + \\ \left| \partial_v I_{n+1}^u - \partial_v I_{2n+2-k}^{u+(n+1-k)z} \right|^2 \end{array} \right) \end{array} \right) + \varphi \cdot \Psi_s(\left|\nabla z\right|^2) du dv\,. \tag{5.2.3}$$

According to the variational method we derive the Euler – Lagrange equation:

$$\sum_{k=1}^{n} g(Forma_k) \left(\begin{array}{l} \Theta \cdot \left(\begin{array}{l} -(n+1-k) \cdot \partial_u I_k^{u-(n+1-k)z} \left(I_{n+1}^u - I_k^{u-(n+1-k)z} \right) + \\ (n+1-k) \cdot \partial_u I_{2n+2-k}^{u+(n+1-k)z} \left(I_{n+1}^u - I_{2n-k}^{u+(n+1-k)z} \right) \end{array} \right) + \\ (1-\Theta) \cdot \left(\begin{array}{l} -(n+1-k) \cdot \partial_{uu} I_k^{u-(n+1-k)z} \left(\partial_u I_{n+1}^u - \partial_u I_k^{u-(n+1-k)z} \right) + \\ (n+1-k) \cdot \partial_{uu} I_{2n+2-k}^{u+(n+1-k)z} \left(\partial_u I_{n+1}^u - \partial_u I_{2n-k}^{u+(n+1-k)z} \right) \end{array} \right) + \\ (1-\Theta) \cdot \left(\begin{array}{l} -(n+1-k) \cdot \partial_{vu} I_k^{u-(n+1-k)z} \left(\partial_v I_{n+1}^u - \partial_v I_k^{u-(n+1-k)z} \right) + \\ (n+1-k) \cdot \partial_{vu} I_{2n+2-k}^{u+(n+1-k)z} \left(\partial_v I_{n+1}^u - \partial_v I_{2n-k}^{u+(n+1-k)z} \right) \end{array} \right) \end{array} \right) \tag{5.2.4}$$

$$+ \varphi \cdot div\left(\Psi'(\left|\nabla z\right|^2) \cdot \nabla z \right) = 0.$$

Here we have denoted:

$$Forma_k = \begin{pmatrix} \Theta \cdot \left(\left| I_{n+1}^{u} - I_{k}^{u-(n+1-k)z} \right|^2 + \left| I_{n+1}^{u} - I_{2n+2-k}^{u+(n+1-k)z} \right|^2 \right) + \\ (1-\Theta) \cdot \left(\left| \partial_u I_{n+1}^{u} - \partial_u I_{k}^{u-(n+1-k)z} \right|^2 + \left| \partial_u I_{n+1}^{u} - \partial_u I_{2n+2-k}^{u+(n+1-k)z} \right|^2 \right) + \\ (1-\Theta) \cdot \left(\left| \partial_v I_{n+1}^{u} - \partial_v I_{k}^{u-(n+1-k)z} \right|^2 + \left| \partial_v I_{n+1}^{u} - \partial_v I_{2n+2-k}^{u+(n+1-k)z} \right|^2 \right) \end{pmatrix}. \quad (5.2.5)$$

To apply the method of linear interpolation we chose two set numbers $A_k \in \mathrm{N}^+$, $b_k \in R$, $\forall k \in [1;n]$, such that $|b_k| < 1$ and $(n+1-k)z = A_k + b_k$. In such a manner we can also express the disparity field:

$$z = \frac{A_k + b_k}{n+1-k}; \quad (5.2.6)$$

now let us rewrite the equation (5.2.4):

$$\sum_{k=1}^{n} g(Forma_k) \begin{pmatrix} \Theta \cdot \begin{pmatrix} -(n+1-k) \cdot \partial_u I_k^{u-A_k} \left(I_{n+1}^{u} - (1-b_k) I_k^{u-A_k} - b_k I_k^{u-A_k-1} \right) + \\ (n+1-k) \cdot \partial_u I_{2n-k}^{u+A_k} \left(I_{n+1}^{u} - (1-b_k) I_{2n+2-k}^{u+A_k} - b_k I_{2n+2-k}^{u+A_k+1} \right) \end{pmatrix} + \\ (1-\Theta) \cdot \begin{pmatrix} -(n+1-k) \cdot \partial_{uu} I_k^{u-A_k} \left(\partial_u I_{n+1}^{u} - (1-b_k) \partial_u I_k^{u-A_k} - b_k \partial_u I_k^{u-A_k-1} \right) + \\ (n+1-k) \cdot \partial_{uu} I_{2n-k}^{u+A} \left(\partial_u I_{n+1}^{u} - (1-b_k) \partial_u I_{2n+2-k}^{u+A_k} - b_k \partial_u I_{2n+2-k}^{u+A_k+1} \right) \end{pmatrix} + \\ (1-\Theta) \cdot \begin{pmatrix} -(n+1-k) \cdot \partial_{vu} I_k^{u-A_k} \left(\partial_v I_{n+1}^{u} - (1-b_k) \partial_v I_k^{u-A_k} - b_k \partial_v I_k^{u-A_k-1} \right) + \\ (n+1-k) \cdot \partial_{vu} I_{2n-k}^{u+A_i} \left(\partial_v I_{n+1}^{u} - (1-b_k) \partial_v I_{2n+2-k}^{u+A_k} - b_k \partial_v I_{2n+2-k}^{u+A_k+1} \right) \end{pmatrix} \end{pmatrix} \quad (5.2.7)$$

$$+ \frac{\varphi}{2} \cdot \begin{pmatrix} (g_{i+1,j} + g_{i,j}) z_{i+1,j} + (g_{i,j} + g_{i-1,j}) z_{i-1,j} + (g_{i,j+1} + g_{i,j}) z_{i,j+1} \\ + (g_{i,j} + g_{i,j-1}) z_{i,j-1} - (4 g_{i,j} + g_{i+1,j} + g_{i-1,j} + g_{i,j+1} + g_{i,j-1}) z_{i,j} \end{pmatrix} = 0,$$

or

$$\sum_{k=1}^{n} \begin{pmatrix} g(Forma_k) \cdot \\ \begin{pmatrix} \Theta \cdot \begin{pmatrix} -(n+1-k) \cdot \partial_u I_k^{u-A_k} \left(I_{n+1}^{u} - I_k^{u-A_k} + b_k (I_k^{u-A_k} - I_k^{u-A_k-1}) \right) + \\ (n+1-k) \cdot \partial_u I_{2n-k}^{u+A_k} \left(I_{n+1}^{u} - I_{2n-k}^{u+A_k} - b_k (I_{2n-k}^{u+A_k+1} - I_{2n-k}^{u+A_k}) \right) \end{pmatrix} + \\ (1-\Theta) \cdot \begin{pmatrix} -(n+1-k) \cdot \partial_{uu} I_k^{u-A_k} \left(\partial_u I_{n+1}^{u} - \partial_u I_k^{u-A_k} + b_k (\partial_u I_k^{u-A_k} - \partial_u I_k^{u-A_k-1}) \right) + \\ (n+1-k) \cdot \partial_{uu} I_{2n-k}^{u+A_k} \left(\partial_u I_{n+1}^{u} - \partial_u I_{2n-k}^{u+A_k} - b_k (\partial_u I_{2n-k}^{u+A_k+1} - \partial_u I_{2n-k}^{u+A_k}) \right) \end{pmatrix} + \\ (1-\Theta) \cdot \begin{pmatrix} -(n+1-k) \cdot \partial_{vu} I_k^{u-A_k} \left(\partial_v I_{n+1}^{u} - \partial_v I_k^{u-A_k} + b_k (\partial_v I_k^{u-A_k} - \partial_v I_k^{u-A_k-1}) \right) + \\ (n+1-k) \cdot \partial_{vu} I_{2n-k}^{u+A_k} \left(\partial_v I_{n+1}^{u} - \partial_v I_{2n-k}^{u+A_k} - b_k (\partial_v I_{2n-k}^{u+A_k+1} - \partial_v I_{2n-k}^{u+A_k}) \right) \end{pmatrix} \end{pmatrix} \quad (5.2.8)$$

$$+ \frac{\varphi}{2} \cdot \begin{pmatrix} (g_{i+1,j} + g_{i,j}) z_{i+1,j} + (g_{i,j} + g_{i-1,j}) z_{i-1,j} + (g_{i,j+1} + g_{i,j}) z_{i,j+1} \\ + (g_{i,j} + g_{i,j-1}) z_{i,j-1} - (4 g_{i,j} + g_{i+1,j} + g_{i-1,j} + g_{i,j+1} + g_{i,j-1}) \dfrac{A_o + b_o}{n+1-o} \end{pmatrix} = 0.$$

According to the (5.2.6) it is possible to have n different expressions for the disparity field. So in the formula (5.2.8) let o be any of the values from $[1;n]$.

At this point we should agree to approximate all left cameras' $\frac{\partial I_k(u,v)}{\partial u}$ for $\forall k \in [1;n+1)$ via backward difference scheme and all right and the centre cameras' $\frac{\partial I_k(u,v)}{\partial u}$ for $\forall k \in [n+1;2n+1]$ via forward difference scheme. Thus we can write:

$$\begin{aligned}
&\sum_{k=1}^{n} g(Forma_k)\begin{pmatrix}
\Theta\cdot\begin{pmatrix}-(n+1-k)\cdot\partial_u I_k^{u-A_k}\left(I_{n+1}^{u}-I_k^{u-A_k}\right)+\\ (n+1-k)\cdot\partial_u I_{2n+2-k}^{u+A_k}\left(I_{n+1}^{u}-I_{2n+2-k}^{u+A_k}\right)\end{pmatrix}+\\
(1-\Theta)\cdot\begin{pmatrix}-(n+1-k)\cdot\partial_{uu} I_k^{u-A_k}\left(\partial_u I_{n+1}^{u}-\partial_u I_k^{u-A_k}\right)+\\ (n+1-k)\cdot\partial_{uu} I_{2n+2-k}^{u+A_k}\left(\partial_u I_{n+1}^{u}-\partial_u I_{2n+2-k}^{u+A_k}\right)\end{pmatrix}+\\
(1-\Theta)\cdot\begin{pmatrix}-(n+1-k)\cdot\partial_{vu} I_k^{u-A_k}\left(\partial_v I_{n+1}^{u}-\partial_v I_k^{u-A_k}\right)+\\ (n+1-k)\cdot\partial_{vu} I_{2n+2-k}^{u+A_k}\left(\partial_v I_{n+1}^{u}-\partial_v I_{2n+2-k}^{u+A_k}\right)\end{pmatrix}
\end{pmatrix}+\\
&-\sum_{k=1}^{n} g(Forma_k)\cdot b_k\begin{pmatrix}
\Theta\cdot\begin{pmatrix}(n+1-k)\cdot\left(\partial_u I_k^{u-A_k}\right)^2+\\ (n+1-k)\cdot\left(\partial_u I_{2n+2-k}^{u+A_k}\right)^2\end{pmatrix}+\\
(1-\Theta)\cdot\begin{pmatrix}(n+1-k)\cdot\left(\partial_{uu} I_k^{u-A_k}\right)^2+\\ (n+1-k)\cdot\left(\partial_{uu} I_{2n+2-k}^{u+A_k}\right)^2\end{pmatrix}\\
(1-\Theta)\cdot\begin{pmatrix}(n+1-k)\cdot\left(\partial_{vu} I_k^{u-A_k}\right)^2+\\ (n+1-k)\cdot\left(\partial_{vu} I_{2n+2-k}^{u+A_k}\right)^2\end{pmatrix}
\end{pmatrix}\\
&+\frac{\varphi}{2}\cdot\begin{pmatrix}(g_{i+1,j}+g_{i,j})z_{i+1,j}+(g_{i,j}+g_{i-1,j})z_{i-1,j}+(g_{i,j+1}+g_{i,j})z_{i,j+1}\\ +(g_{i,j}+g_{i,j-1})z_{i,j-1}-(4g_{i,j}+g_{i+1,j}+g_{i-1,j}+g_{i,j+1}+g_{i,j-1})\frac{A_o}{n+1-o}\end{pmatrix}\\
&-\frac{\varphi\cdot b_o}{2(n+1-o)}\cdot(4g_{i,j}+g_{i+1,j}+g_{i-1,j}+g_{i,j+1}+g_{i,j-1})=0.
\end{aligned} \tag{5.2.9}$$

Let us introduce further notations:

$$\begin{aligned}
g_r &= g_{i+1,j}+g_{i,j};\\
g_l &= g_{i,j}+g_{i-1,j};\\
g_u &= g_{i,j+1}+g_{i,j};\\
g_d &= g_{i,j}+g_{i,j-1};\\
g_c &= g_{i+1,j}+g_{i-1,j}+g_{i,j+1}+g_{i,j-1}+4g_{i,j}.
\end{aligned} \tag{5.2.10}$$

Please, pay attention that $g_c = g_r + g_l + g_u + g_d$. Now let us rewrite the equation (5.2.9):

$$\sum_{k=1}^{n} g(Forma_k)\cdot(n+1-k)b_k\begin{pmatrix}\Theta\cdot\begin{pmatrix}\left(\partial_u I_k^{u-A_k}\right)^2+\\ \left(\partial_u I_{2n+2-k}^{u+A_k}\right)^2\end{pmatrix}+\\ (1-\Theta)\cdot\begin{pmatrix}\left(\partial_{uu} I_k^{u-A_k}\right)^2+\\ \left(\partial_{uu} I_{2n+2-k}^{u+A_k}\right)^2\end{pmatrix}\\ (1-\Theta)\cdot\begin{pmatrix}\left(\partial_{vu} I_k^{u-A_k}\right)^2+\\ \left(\partial_{vu} I_{2n+2-k}^{u+A_k}\right)^2\end{pmatrix}\end{pmatrix}$$

$$+\frac{\varphi\cdot b_k}{2n(n+1-k)}\cdot g_c =$$

$$\sum_{k=1}^{n} g(Forma_k)\cdot(n+1-k)\begin{pmatrix}\Theta\cdot\begin{pmatrix}-\partial_u I_k^{u-A_k}\left(I_{n+1}^{u}-I_k^{u-A_k}\right)+\\ \partial_u I_{2n+2-k}^{u+A_k}\left(I_{n+1}^{u}-I_{2n+2-k}^{u+A_k}\right)\end{pmatrix}+\\ (1-\Theta)\cdot\begin{pmatrix}-\partial_{uu} I_k^{u-A_k}\left(\partial_u I_{n+1}^{u}-\partial_u I_k^{u-A_k}\right)+\\ \partial_{uu} I_{2n+2-k}^{u+A_k}\left(\partial_u I_{n+1}^{u}-\partial_u I_{2n+2-k}^{u+A_k}\right)\end{pmatrix}+\\ (1-\Theta)\cdot\begin{pmatrix}-\partial_{vu} I_k^{u-A_k}\left(\partial_v I_{n+1}^{u}-\partial_v I_k^{u-A_k}\right)+\\ \partial_{vu} I_{2n+2-k}^{u+A_k}\left(\partial_v I_{n+1}^{u}-\partial_v I_{2n+2-k}^{u+A_k}\right)\end{pmatrix}\end{pmatrix} \qquad (5.2.11)$$

$$+\frac{\varphi}{2n}\cdot\left(g_r z_{i+1,j}+g_l z_{i-1,j}+g_u z_{i,j+1}+g_d z_{i,j-1}-g_c\frac{A_k}{n+1-k}\right).$$

The equation (5.2.11) is linear. It means that will be fulfilled with any fixed k. Thus we can get rid of the sum on k in this equation and achieve n different equations for different k. Than we resolve the equation about b:

$$g(Forma_k)\cdot(n+1-k)\begin{pmatrix}\Theta\cdot\begin{pmatrix}-\partial_u I_k^{u-A_k}\left(I_{n+1}^{u}-I_k^{u-A_k}\right)+\\ \partial_u I_{2n+2-k}^{u+A_k}\left(I_{n+1}^{u}-I_{2n+2-k}^{u+A_k}\right)\end{pmatrix}+\\ (1-\Theta)\cdot\begin{pmatrix}-\partial_{uu} I_k^{u-A_k}\left(\partial_u I_{n+1}^{u}-\partial_u I_k^{u-A_k}\right)+\\ \partial_{uu} I_{2n+2-k}^{u+A_k}\left(\partial_u I_{n+1}^{u}-\partial_u I_{2n+2-k}^{u+A_k}\right)\end{pmatrix}+\\ (1-\Theta)\cdot\begin{pmatrix}-\partial_{vu} I_k^{u-A_k}\left(\partial_v I_{n+1}^{u}-\partial_v I_k^{u-A_k}\right)+\\ \partial_{vu} I_{2n+2-k}^{u+A_k}\left(\partial_v I_{n+1}^{u}-\partial_v I_{2n+2-k}^{u+A_k}\right)\end{pmatrix}\end{pmatrix}$$

$$\frac{b_k}{n+1-k}=\frac{+\frac{\varphi}{2n}\cdot\left(g_r z_{i+1,j}+g_l z_{i-1,j}+g_u z_{i,j+1}+g_d z_{i,j-1}-g_c\frac{A_k}{n+1-k}\right)}{g(Forma_k)\cdot(n+1-k)^2\begin{pmatrix}\Theta\cdot\begin{pmatrix}\left(\partial_u I_k^{u-A_k}\right)^2+\\ \left(\partial_u I_{2n+2-k}^{u+A_k}\right)^2\end{pmatrix}+\\ (1-\Theta)\cdot\begin{pmatrix}\left(\partial_{uu} I_k^{u-A_k}\right)^2+\\ \left(\partial_{uu} I_{2n+2-k}^{u+A_k}\right)^2\end{pmatrix}\\ (1-\Theta)\cdot\begin{pmatrix}\left(\partial_{vu} I_k^{u-A_k}\right)^2+\\ \left(\partial_{vu} I_{2n+2-k}^{u+A_k}\right)^2\end{pmatrix}\end{pmatrix}+\frac{\varphi}{2n}\cdot g_c}, \qquad (5.2.12)$$

for $\forall k \in [1;n]$.

Using the formulae (5.2.12) and (5.2.6) let us write the formula for the disparity field from the next time step:

$$z^* = \frac{A_k}{n+1-k} + \frac{+\frac{\varphi}{2n}\cdot\left(g_r z_{i+1,j} + g_l z_{i-1,j} + g_u z_{i,j+1} + g_d z_{i,j-1} - g_c \frac{A_k}{n+1-k}\right)}{g(Forma_k)\cdot(n+1-k)^2\begin{pmatrix}\Theta\cdot\begin{pmatrix}\left(\partial_u I_k^{u-A_k}\right)^2 + \\ \left(\partial_u I_{2n+2-k}^{u+A_k}\right)^2\end{pmatrix} + \\ (1-\Theta)\cdot\begin{pmatrix}\left(\partial_{uu} I_k^{u-A_k}\right)^2 + \\ \left(\partial_{uu} I_{2n+2-k}^{u+A_k}\right)^2\end{pmatrix} + \\ (1-\Theta)\cdot\begin{pmatrix}\left(\partial_{vu} I_k^{u-A_k}\right)^2 + \\ \left(\partial_{vu} I_{2n+2-k}^{u+A_k}\right)^2\end{pmatrix}\end{pmatrix} + \frac{\varphi}{2n}\cdot g_c},$$

$$g(Forma_k)\cdot(n+1-k)\begin{pmatrix}\Theta\cdot\begin{pmatrix}-\partial_u I_k^{u-A_k}\left(I_{n+1}^u - I_k^{u-A_k}\right)+ \\ \partial_u I_{2n+2-k}^{u+A_k}\left(I_{n+1}^u - I_{2n+2-k}^{u+A_k}\right)\end{pmatrix} + \\ (1-\Theta)\cdot\begin{pmatrix}-\partial_{uu} I_k^{u-A_k}\left(\partial_u I_{n+1}^u - \partial_u I_k^{u-A_k}\right)+ \\ \partial_{uu} I_{2n+2-k}^{u+A_k}\left(\partial_u I_{n+1}^u - \partial_u I_{2n+2-k}^{u+A_k}\right)\end{pmatrix} + \\ (1-\Theta)\cdot\begin{pmatrix}-\partial_{vu} I_k^{u-A_k}\left(\partial_v I_{n+1}^u - \partial_v I_k^{u-A_k}\right)+ \\ \partial_{vu} I_{2n+2-k}^{u+A_k}\left(\partial_v I_{n+1}^u - \partial_v I_{2n+2-k}^{u+A_k}\right)\end{pmatrix}\end{pmatrix}, \quad (5.2.13)$$

for $\forall k \in [1;n]$.

Where the notation (5.2.5) in terms of linear interpolation approach reads:

$$Forma_k = \begin{pmatrix}\Theta\cdot\begin{pmatrix}\left|I_{n+1}^u - (1-b_k)I_k^{u-A_k} - b_k I_k^{u-A_k-1}\right|^2 + \\ \left|I_{n+1}^u - (1-b_k)I_{2n+2-k}^{u+A_k} - b_k I_{2n+2-k}^{u+A_k+1}\right|^2\end{pmatrix} + \\ (1-\Theta)\cdot\begin{pmatrix}\left|\partial_u I_{n+1}^u - (1-b_k)\partial_u I_k^{u-A_k} - b_k \partial_u I_k^{u-A_k-1}\right|^2 + \\ \left|\partial_u I_{n+1}^u - (1-b_k)\partial_u I_{2n+2-k}^{u+A_k} - b_k \partial_u I_{2n+2-k}^{u+A_k+1}\right|^2\end{pmatrix} + \\ (1-\Theta)\cdot\begin{pmatrix}\left|\partial_v I_{n+1}^u - (1-b_k)\partial_v I_k^{u-A_k} - b_k \partial_v I_k^{u-A_k-1}\right|^2 + \\ \left|\partial_v I_{n+1}^u - (1-b_k)\partial_v I_{2n+2-k}^{u+A_k} - b_k I_{2n+2-k}^{u+A_k+1}\right|^2\end{pmatrix}\end{pmatrix}. \quad (5.2.14)$$

With the help of formulae (5.2.13), (5.2.14) and (5.2.10) we can calculate n different disparity fields, which are the *partial* solutions for the certain set of cameras (see figure 5.1) To achieve the *general* solution disparity field, which corresponds to all the defined cameras and provide the interaction between the *partial* disparity fields, we sum up weighted values of them together:

$$Z(u,v) = \sum_{k=1}^{n} t_k \cdot z_k(u,v) \text{, where } \sum_{k=1}^{n} t_k = 1; \qquad (5.2.15)$$

here $Z(u,v)$ denotes the *general* solution and $z_k(u,v)$ denote *partial* solutions. In the experiments, we always use constant weighted coefficients $t_k = 1/n$, $\forall k \in [1;n]$.

5.3 Summary

We have constructed the multi – view mathematical model for the depth-map reconstruction and derived a linear numerical scheme for it. We used the linear interpolation method for the disparity driven approach. Numerical schemes for the first order Taylor expansion method and the depth driven approach could by achieved analogously, if we apply the concept (5.1.1) to the sections 4.5 and 4.6.

The next chapter illustrates the performance and reliabilities of all the methods, described and offered in this thesis.

Chapter 6

Experimental Results

In this chapter we will present experimental results that illustrate the theory from the previous chapters. First, we will shortly discuss the error measures and the visualization techniques for the computed depth-map solution. Then, we will give a short description of the used test sequences and the actual program implementation.

In Chapter 2 we presented methods for depth-map computation, namely variational methods. These methods are based on the minimization of an energy functional, composed of a data and a smoothness terms. In Chapter 3 we developed suitable constancy assumptions for the data term and smoothness assumptions for the smoothness term and invented methods to control the matching process on run. In Chapter 4 we discussed depth-driven and disparity driven approaches and proposed to use linear interpolation instead of the first order Taylor expansion within the Euler – Lagrange equation. Moreover, we described the coarse-to-fine levels technique and the warping technique, and explained how to achieve more reliable results without them. The second, the third and the fourth parts of this chapter lead to experimental results that illustrate the theory from the aforementioned chapters. The conclusion to the thesis is given in the fifth section of this chapter.

6.1 Experimental setup

Before presenting the actual experiments and results, let us briefly discuss how we are going to assess the errors, the way we are going to visualize the solutions and the test sequences we have used in our experiments.

6.1.1 Test data sets

The main goal of this thesis is the computation of the depth-map for stereo images. As this is a relatively young research field, there are not that many suitable test data sets available. However, there are several data sets, that are well-known and there is a ground truth computed for them. In this section we give a brief introduction to the test data sets that we have chosen to use for our experiments.

It this thesis we use the stereo data sets from Middlebury University [WMB07]. These data sets include a stereo image, represented as at the least 2 photos made from different positions at the same moment of time, a true solution disparity map and also the most of data sets are supplied with an occlusion pixels map. The resolution of these stereo images is approximately 450 x 375 pixels and the objects' displacement in these pictures does not exceed 22 pixels, i.e. lies in range of [0; 22). The true solution disparity map represents the pixels' displacement field scaled by a certain *scale factor*, therefore it is an 8-bit grayscale picture with pixel values lying in range of [0; 255]. The occlusion pixels map is a 2-bit black-and-white picture, where black pixels represent occluded regions in the depth-map, and white pixels represent position of reliable data in the corresponding ground truth map (figure 6.1).

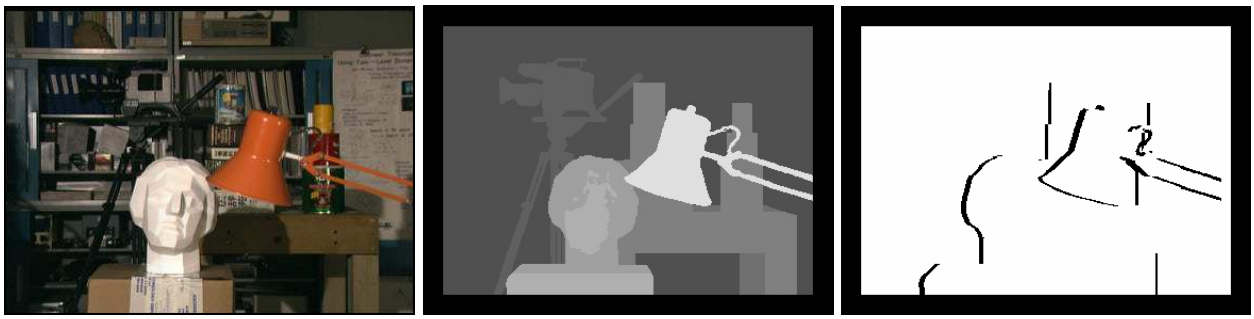

Figure 6.1: "Tsukuba" test data set: **Left:** The left frame of scene; **Centre:** Ground truth disparity map; **Right:** Occlusion pixels' map.

Among with the "Tsukuba" data set we use 6 more data sets, depicted in figure 6.2. We will present the experimental results in the same order which the thesis was written: we will start with comparing results achieved by the depth-driven and disparity-driven methods. For this purpose we take "Moebius" and "Doll" scenes. As we can see at figure 6.2, these scenes have numerous objects, and a major part of them are far situated near to the background. Evaluating the distances for such objects is the weak side of the depth-driven method and at the same time it is the strong side of the disparity-driven method. Therefore these two scenes are the best choice to illustrate the difference of the methods.

After that we will proceed with comparing the method of the first order Taylor expansion and the method of linear interpolation. Here we will use the "Art" and "Flowerpots" scenes. The "Art" scene includes different size objects in the middle range and thin brushes in the foreground. And the "Flowerpots" scene consists of massive objects at the foreground. It is very interesting to compare the reliability of the methods on thin and massive objects with big displacements.

Figure 6.2: Test data sets: **Top Left:** "Moebius" scene; **Top Centre:** "Doll" scene; **Top Right:** "Art" scene; **Bottom Left:** "Flowerpots" scene; **Bottom Centre:** "Teddy" scene; **Bottom Right:** "Cones" scene.

At the end we come to the experiments with the multi – view model on "Tsuluba", "Teddy" and "Cones" data sets. They include up to 5 pictures and the efficiency of other methods for the depth-map reconstruction for these scenes is available at [WMB07]. Talking about efficiency of a method, let us proceed to the next subsection.

6.1.2 Error estimation

As we are about to perform some experiments we need a way to measure how good or bad they are. As mentioned before, the depth-map is an array of scalar values. For every pixel we have a distance value from an observer to an object in scene. For any depth value in a pixel we have one-to-one accordance with the disparity of that pixel between pictures of a stereo image.

For our experiments we use data sets supplied with ground truth disparity maps. Having the ideal solution available, we can compare our results with it. Our solution is also a scalar array which represents either depth-map or disparity map, so we need a way to compare two disparity maps. Here we introduce the notion of the *bad pixel*. Let the solution disparity map will be denoted as $z(u,v)$ and the ground truth disparity map as $d(u,v)$, then the bad pixel is the pixel for which the following inequality is not true:

$$\left|d(u,v)-z(u,v)\right| \le \delta, \tag{6.1.1}$$

where δ is a small threshold.

As the criteria of error estimation we take the percentage of the bad pixels in the solution image. During this estimation we do not consider the occluded pixels.

6.1.3 Results visualization

We have already presented an error measurement technique for estimating our solution. Of course, this is the most precise way to measure the quality of the solution. Here follow some visualization techniques we can use.

Since human eye can distinguish only 40 grayscales, and 2,000,000 colors we represent the scaled disparity and depth maps, which are in fact 8-bit grayscale images, colored with the help of an "ice-fire" palette. We will always add to the color-coded images the palette bar in order to let the reader estimate the disparity or depth in figures.

For more clearness we will also visualize the distribution of bad pixels, which are calculated according to the formula (6.1.1). For this purpose we will draw them with the red color on a grayscale solution image. Black color will denote occluded regions, which are not used during the error estimation.

6.1.4 Implementation details

After we have presented the test sequences that we are going to use for our experiments, we should also present the current implementation used for computing the reported results.

The test program has been written in two instances. The first instance is written in GNU C++ for Linux SuSE 10.0 and the second one in Microsoft C# for Microsoft Windows XP SP2. Both instances are based on routines that perform the depth-driven and disparity-driven methods with first order Taylor expansion and linear interpolation approaches. The program is also equipped with routines that solve a linear/non-linear system of equations, using the SOR method in case of linear system of equations. Also, the coarse-to-fine technique is applied with maximum 16 levels. In case of using the first order Taylor expansion, the coarse-to-fine technique is supplied also with the warping technique.

All of the above mentioned routines were self-written and optimized for speed. The constants and parameters that were used by these routines will be presented in tables, since they are different for different methods. Except for the parameter Θ, which is always equal to the value *0,85*.

Around the above mentioned functionality was build a new one based on theory from the paragraph 3.6, that allows to get rid of coarse-to-fine levels technique, warping technique, speed up the convergence of an algorithm, have on run error recovery and make the solver human– and outer parameters– independent as much as possible.

The test program has a user-friendly interface, which allows the user to watch the automatic parameters optimization and the solution evolution during iterations. Also the interface allows manually to specify the parameters of the methods in the course of the program execution and to monitor the performance of the methods with respect to the error estimation.

6.2 Depth vs. disparity

In this section we make experiments on the "Dolls" and "Moebius" scenes. Each stereo image consist of two bitmap pictures with resolution 463 x 370 pixels. In the experiment we used geometric levels pyramid with 8 coarse levels and the coarsest level resolution about 57 x 46 pixels. The displacement of objects in the scene varies between 6 and 21 pixels, so the maximal displacement on the coarsest level does not exceed 3 pixels. The scale factor here is equal to 12. We use a linear interpolation method in these experiments that easily handles 4 pixels displacements.

6.2.1 Error estimation

We start with the "Doll" scene. Here we use the parameters, shown in the table 6.1:

Table 6.1: "Doll" scene: setup parameters.

	Depth-driven approach	Disparity-driven approach
ρ	*1,5*	*1,5*
φ	$1x10^4$	$3x10^2$
τ	$1x10^{-4}$	–
λ_d	*0,1*	*0,3*
λ_s	$1x10^{-3}$	$2,5x10^{-3}$

As we can see, with the depth-driven method we ought to use a very small time step τ in that time, while the linear numerical scheme (4.6.12) for the disparity-driven method does not require this time step at all. We shell discuss the speed of convergence of these methods in the next section of this paragraph.

At the figure 6.3 we illustrate the results of the first experiment – difference between the depth-driven and the disparity-driven method. As we can see from the table 6.1 we tried to create almost the same conditions for the experiment. The solution disparity map look similar, especially the foreground objects on them (Figure 6.3 middle right and bottom right pictures). But we can clearly see that the background of the solution image of the depth-driven approach is overblurred, while at the solution image of the disparity-driven method we can neatly distinguish the dolls' heads.

Now let us compare the middle left and the bottom left pictures of the figure 6.3. The distribution of the bad pixels again is very similar at the foreground – the red areas in the centre and bottom of the pictures somewhere larger, somewhere smaller, but look similar. And in the background, where the depth-driven method gives overblurred results – at the top of the middle left picture of the figure 6.3 we observe a large red area – the area where the depth-driven method failed to determine the correct depth. Total amount of bad pixels in the result image of the depth-driven method is near 5 times bigger than in the result image of the disparity-driven method.

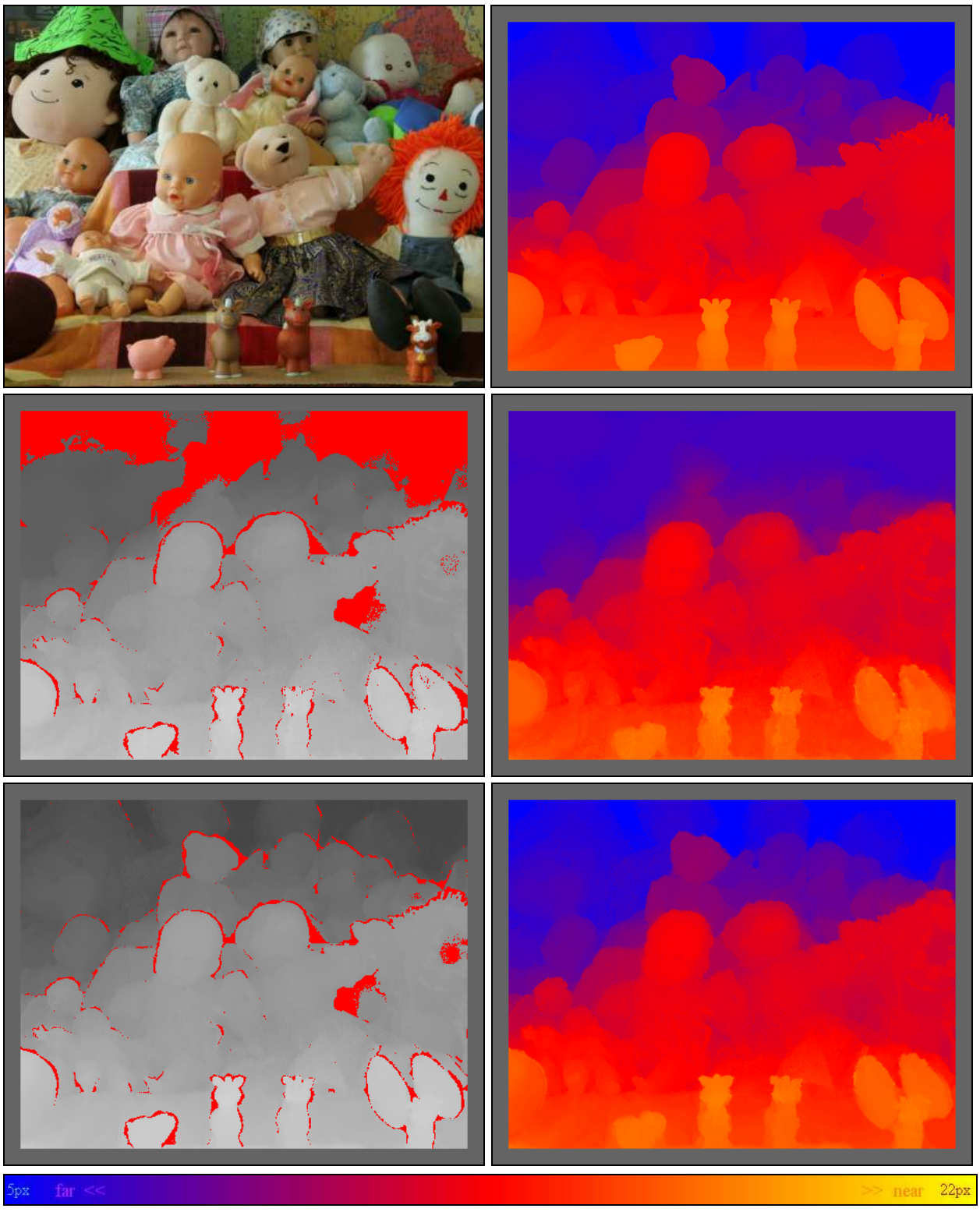

Figure 6.3: Experiment I: "Doll" scene: **Top Left:** The middle frame of the scene; **Top Right:** Ground truth disparity map; **Middle Left:** Distribution of the bad pixels in the solution achieved by the depth-driven method ($\delta = 0{,}75$); **Middle Right:** Depth-driven solution disparity map; **Bottom Left:** Distribution of the bad pixels in the solution achieved by the disparity-driven method ($\delta = 0{,}75$); **Bottom Right:** Disparity-driven solution disparity map.

At the figure 6.3 we have presented the distribution maps of the bad pixels in the solution images achieved by the investigated methods for the threshold value $\delta = 0{,}75$. The percentage of bad pixels for different δ is presented in the table 6.2.

Table 6.2: "Doll" scene: percentage of bad pixels for different δ.

δ	Depth-driven approach	Disparity-driven approach
0,5	*24,58 %*	*6,61 %*
0,75	*18,67 %*	*3,91 %*
1	*13,29 %*	*2,37 %*
1,5	*7,51 %*	*1,23 %*
2	*3,46 %*	*0,7 %*

At the table 6.2 we can see that the percentage of bad pixels tends to zero, but everywhere the mistake of the depth-driven method approximately 5 times larger than the mistake of the disparity-driven method.

For the second experiment, we took the "Moebius" scene with almost the same parameters like before, which are shown in the table 6.3:

Table 6.3: "Moebius" scene: setup parameters.

	Depth-driven approach	Disparity-driven approach
ρ	*2*	*1,5*
φ	$1x10^4$	$3x10^2$
τ	$1x10^{-4}$	–
λ_d	*0,1*	*0,3*
λ_s	$1x10^{-3}$	$2,5x10^{-3}$

At the figure 6.4 we illustrate the results of the second experiment – difference between the depth-driven and the disparity-driven methods on the "Moebius" scene. As it was expected, the results give us the same picture: for the near and reasonably far objects, the methods give very similar results, but for the far distant objects, the depth-driven does not able to estimate the distance to the objects. This phenomenon was described in section 4.1.

At the figure 6.4 we have presented the distribution maps of the bad pixels in the solution images achieved by the investigated methods for the threshold value $\delta = 0{,}75$. The percentage of bad pixels for different δ is presented in the table 6.4.

Table 6.4: "Moebius" scene: percentage of bad pixels for different δ.

δ	Depth-driven approach	Disparity-driven approach
0,5	*23,11 %*	*13 %*
0,75	*19,64 %*	*9,6 %*
1	*17,43 %*	*7,55 %*
1,5	*7,98 %*	*4,54 %*
2	*2,66 %*	*1,37 %*

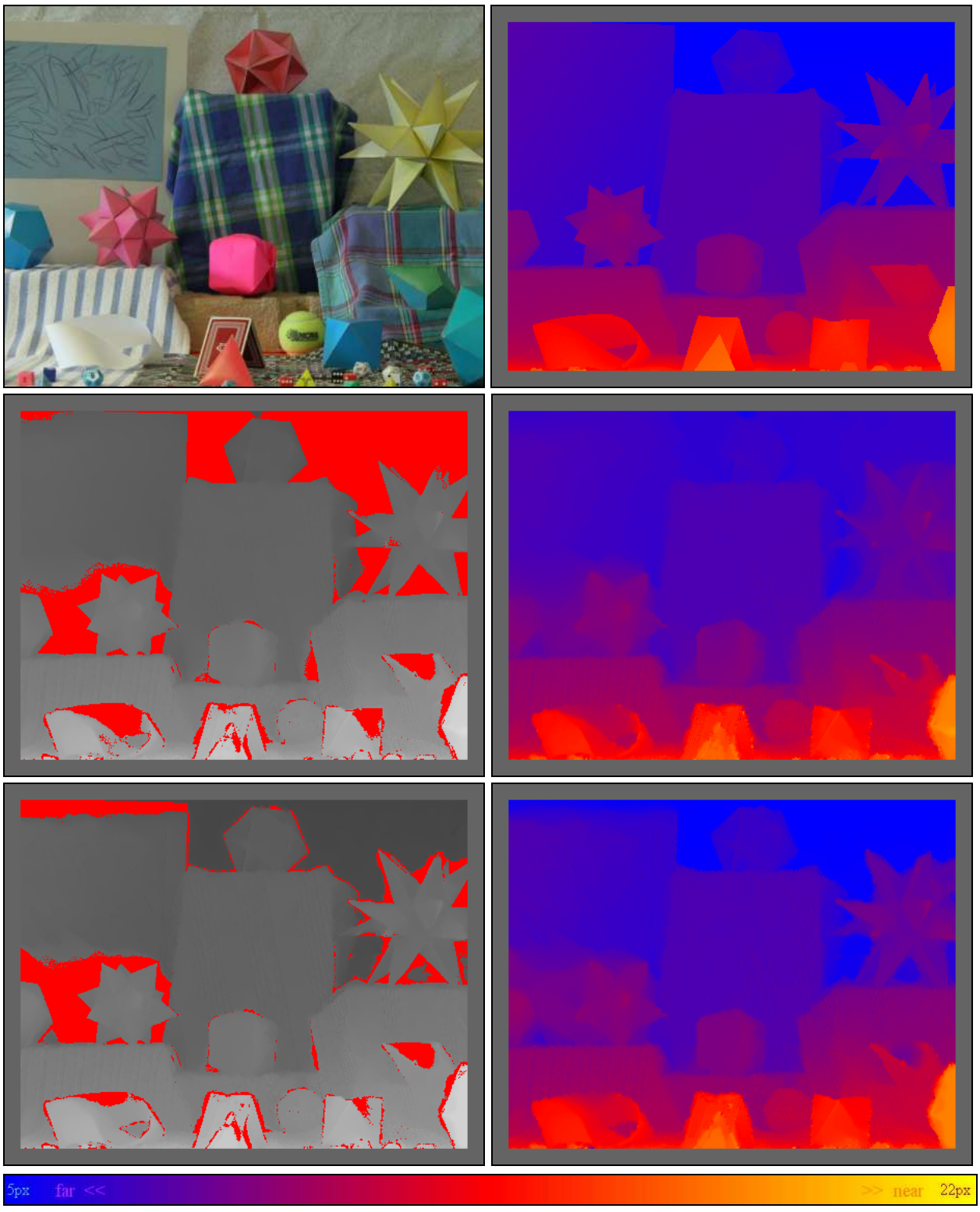

Figure 6.4: Experiment II: "Moebius" scene: **Top Left:** The middle frame of the scene; **Top Right:** Ground truth disparity map; **Middle Left:** Distribution of the bad pixels in the solution achieved by the depth-driven method ($\delta = 0{,}75$); **Middle Right:** Depth-driven solution disparity map; **Bottom Left:** Distribution of the bad pixels in the solution achieved by the disparity-driven method ($\delta = 0{,}75$); **Bottom Right:** Disparity-driven solution disparity map.

Now let us proceed with the discussion how fast these methods are. In order to do that, we will compare the number of iterations and how much time one iteration require for both methods in the following section.

6.2.2 Estimation of the convergence speed

To estimate the speed of convergence of the algorithms for the depth-driven and disparity-driven method we plot the graphs which illustrate the percentage of bad pixels depend on the iteration number. Here we took the finest level with the same initial data and 6 thousand iterations.

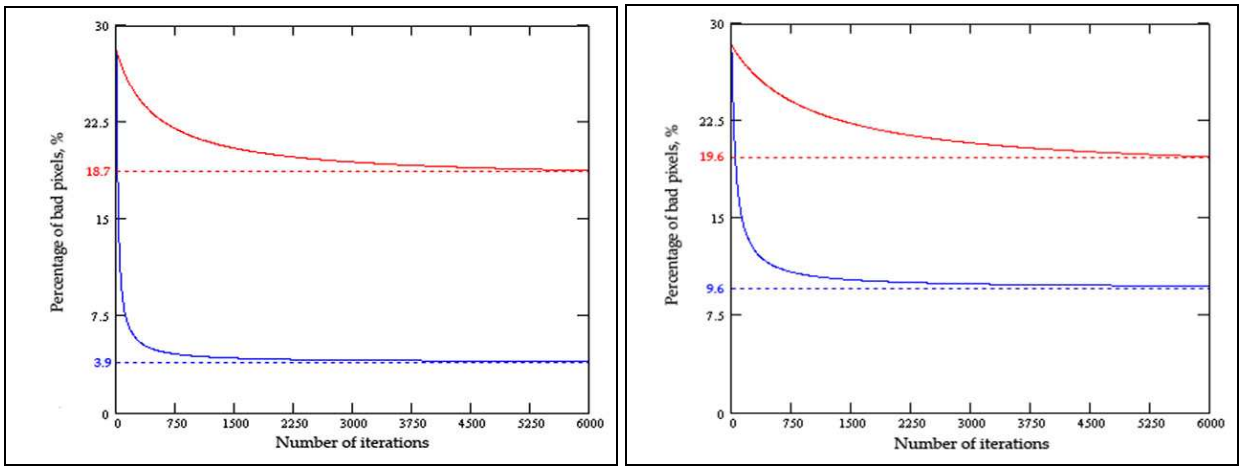

Figure 6.5: Experiment III: Convergence speed: red graph – the depth-driven method; blue graph – disparity-driven method: **Left:** "Dolls" scene for $\delta = 0{,}75$; **Right:** "Moebius" scene for $\delta = 0{,}75$.

At the figure 6.5 we can see that for the disparity-driven method less than 50 iterations are to reach the limit which the depth-driven method reaches in 6000 iterations. In general the blue graph is more tended to the ordinate axis, while the red one is more smooth and steady. It shows that the speed of convergence of the linear disparity-driven problem is faster than the speed of convergence of the non-linear depth-driven problem. Moreover, the straight-forward implementation of the algorithm shows that the one iteration of the non-linear system of equations takes more time that one iteration of the linear system of equations.

6.3 First order Taylor expansion vs. linear interpolation

In this section we make experiments on the "Art" and "Flowerpots" scenes. Each stereo image consists of two bitmap pictures. The stereo image of "Art" scene has resolution 463 x 370 pixels and the stereo image if "Flowerpots" scene has resolution 437 x 370 pixels. In the experiment we used arithmetic levels pyramid with 16 coarse levels and the coarsest level resolution about 29 x 23 pixels. The displacement of objects in the scene varies between 7 and 21 pixels, so the maximal displacement on the coarsest level does not exceed 1,5 pixels. The scale factor here is equal to 12. We ought to use so much coarse levels and so small resolution of the coarsest level, because as we have described in section 4.2 the method of first order Taylor expansion is not able to process big displacements and, moreover, should involve the warping technique.

6.3.1 Error estimation

We start with the "Art" scene. Here we use the parameters, shown in the table 6.5:

Table 6.5: "Art" scene: setup parameters.

	First order Taylor expansion	Linear interpolation
ρ	*1,5*	*1*
φ	*5x10²*	*3x10²*
CL[6]	*16*	*8*
λ_d	*0,3*	*0,3*
λ_s	*2,5x10⁻³*	*2,5x10⁻³*

Since the methods in some extend are very similar – the method of linear interpolation is the general case of the method of first order Taylor expansion, as it was proven in section 4.3, we can use almost the same setup parameters for both methods, except the number of coarse levels. The method of linear interpolation in contrast to the first order Taylor expansion can handle few times bigger displacements and without warping, so here we use the two times smaller number of the coarse levels.

At the figure 6.6 we can se that both methods gives almost the same results, except that, the method of first order Taylor expansion gives more artefacts at the near situated objects and more outliers overall the solution. This is resulting from the inaccuracy of the warping technique and impossibility of the first order expansion method to handle large displacements at the finest level.

At the figure 6.6 we have presented the distribution maps of the bad pixels in the solution images achieved by the investigated methods for the threshold value $\delta = 1$. The percentage of the bad pixel for different δ is presented in the table 6.6:

Table 6.6: "Art" scene: percentage of bad pixels for different δ.

δ	First order Taylor expansion	Linear interpolation
0,5	*19,88 %*	*18,66 %*
0,75	*16,05 %*	*13,98 %*
1	*14,09 %*	*11,58 %*
1,5	*11,55 %*	*9,44 %*
2	*10,01 %*	*7,96 %*

The errors are very similar and do not differ from each other more than 1 % – 2 % percents. But truly speaking all the values in the table 6.6 are smaller for the linear interpolation method than the corresponding values for first order Taylor expansion. We can conclude that on the "Art" scene the linear interpolation method gave better results, and appeared to be faster, since it requires less number of coarse levels.

[6] Abbreviation from "coarse levels" (CL), i.e. number of coarse levels.

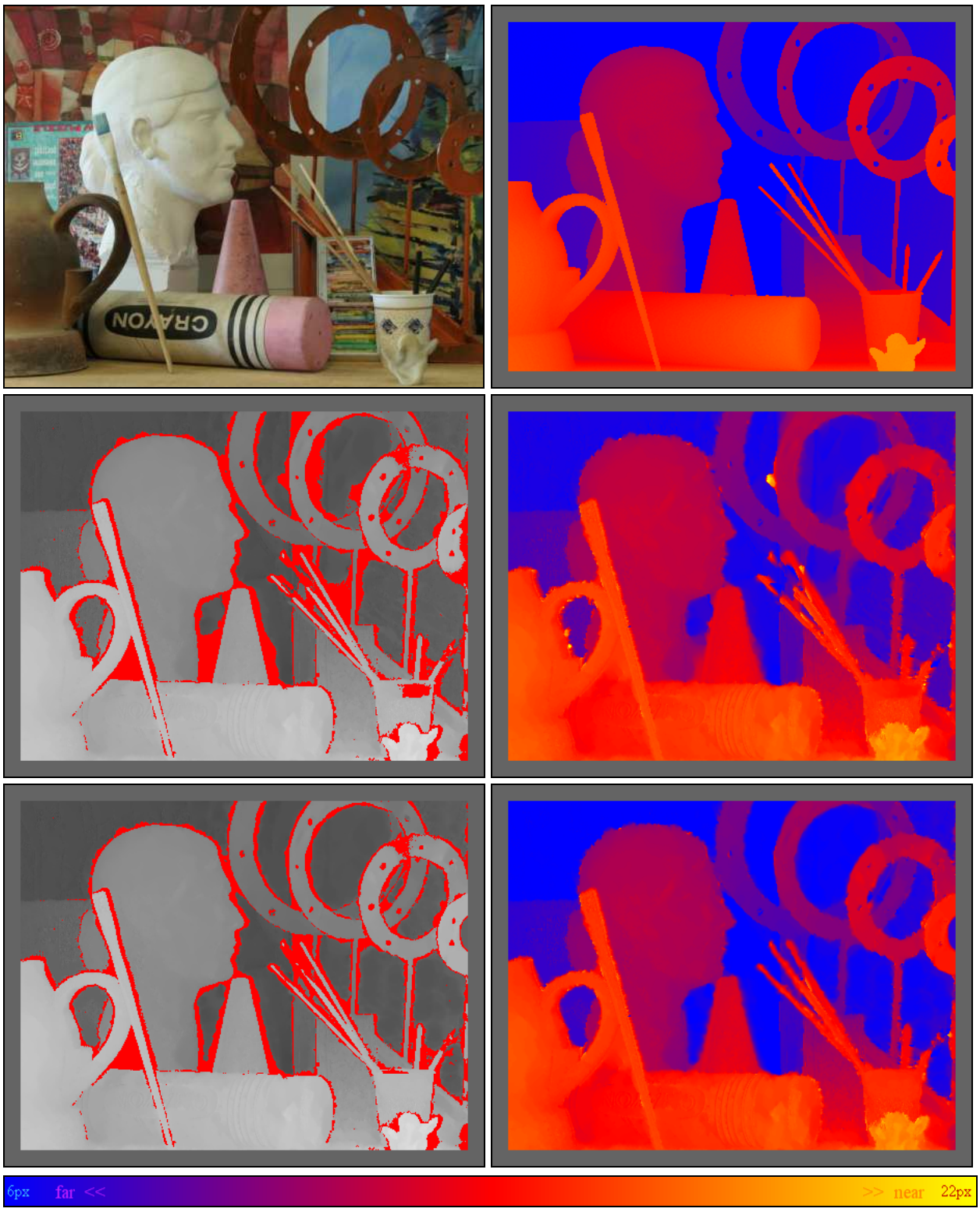

Figure 6.6: Experiment IV: "Art" scene: **Top Left:** The middle frame of the scene; **Top Right:** Ground truth disparity map; **Middle Left:** Distribution of the bad pixels in the solution achieved by the first order Taylor expansion ($\delta = 1$); **Middle Right:** First order Taylor expansion solution disparity map; **Bottom Left:** Distribution of the bad pixels in the solution achieved by the linear interpolation method ($\delta = 1$); **Bottom Right:** Linear interpolation solution disparity map.

Now let us do the similar experiment on another scene –′"Flowerpots". The setup parameters are shown in table 6.7 and the results of the experiment are illustrated at the figure 6.7.

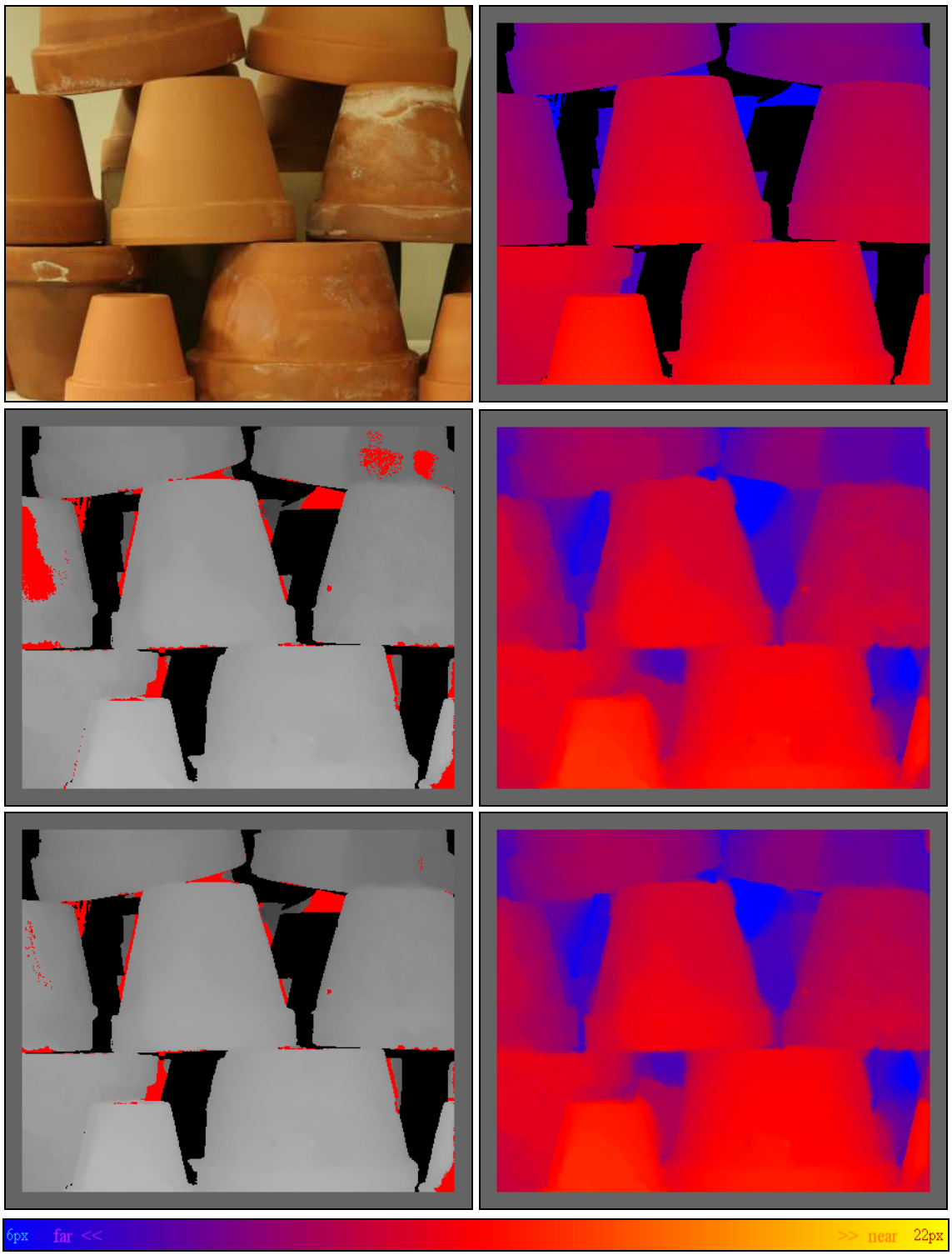

Figure 6.7: Experiment V: "Flowerpots" scene: **Top Left:** The middle frame of the scene; **Top Right:** Ground truth disparity map; **Middle Left:** Distribution of the bad pixels in the solution achieved by the first order Taylor expansion ($\delta = 1$); **Middle Right:** First order Taylor expansion solution disparity map; **Bottom Left:** Distribution of the bad pixels in the solution achieved by the linear interpolation method ($\delta = 1$); **Bottom Right:** Linear interpolation solution disparity map.

As it was ought to be expected the results are very similar, expect some small artifacts, which are appears on the object's borders and the nearest points of the objects. Also we can see that the number of features on both solution pictures is the same.

Table 6.7: "Flowerpots" scene: setup parameters.

	First order Taylor expansion	Linear interpolation
ρ	*1,25*	*0,8*
φ	$2x10^2$	$1x10^2$
CL	*16*	*8*
λ_d	*0,3*	*0,3*
λ_s	$2,5x10^{-3}$	$2,5x10^{-3}$

Comparing the percentage of bad pixels in the solution maps gained by the methods of interest, we conclude that the method of linear interpolation gives again better results on the scenes with massive objects than the method of the first order Taylor expansion (see table 6.8).

Table 6.8: "Flowerpots" scene: percentage of bad pixels for different δ.

δ	First order Taylor expansion	Linear interpolation
0,5	*15,48 %*	*10,55 %*
0,75	*8,82 %*	*5,13 %*
1	*5,13 %*	*2,47 %*
1,5	*2,21 %*	*1,79 %*
2	*1,79 %*	*1,33 %*

6.3.2 Getting rid of the coarse levels

At the conclusion to the paragraph 6.3, we can write that the method of linear interpolation is more accurate then the method of first order Taylor expansion, it does not require the warping technique directly and a big number of the coarse levels, what makes it more fast and reliable. In addition we may say that if we combine the linear interpolation method with the method of automatic control of process parameters, we can get rid of the coarse levels and make all the calculations on the fine level. The experiments, described in this section show reliability of such approach, and that the variation process still capable to avoid getting stuck in a local minima.

The evolution of parameters we can observe in the table 6.9. In the first column we see the setup parameters that are defined by a user, and in the next columns follow the parameters which choose the solver during the iteration process.

Table 6.9: "Art" scene: setup parameters evolution.

It x 10^3 / *parameter*	0 - 2	2 – 3	3 – 3,5	3,5 – 3,8	3,8 – 4
ρ	*3*	*1,5*	*0*	*0*	*0*
φ	*1000*	*100*	*10*	*500*	*300*
λ_d	*Tichonov*	*Tichonov*	*Tichonov*	*0,5*	*0,3*
λ_s	*Tichonov*	*Tichonov*	*Tichonov*	*0,03*	*0,0025*

Now let us observe the figure 6.8, which shows how it works. The initial depth displacement for the solution was set to 12 pixels and the total amount of iterations to 4 thousand. At the table 6.9 we see that during the first 2 thousand iterations the solver uses the Tichonov regularization, which does not require an additional lambda parameter. Moreover, during the first 2 thousand iterations the solver uses a very big smoothness parameter φ and a large image presmoothing parameter ρ. At these steps, the solver creates a very smooth disparity map – the fundament for all the subsequent calculations. Exactly this guaranties us that we will not get stuck in poor local minima.

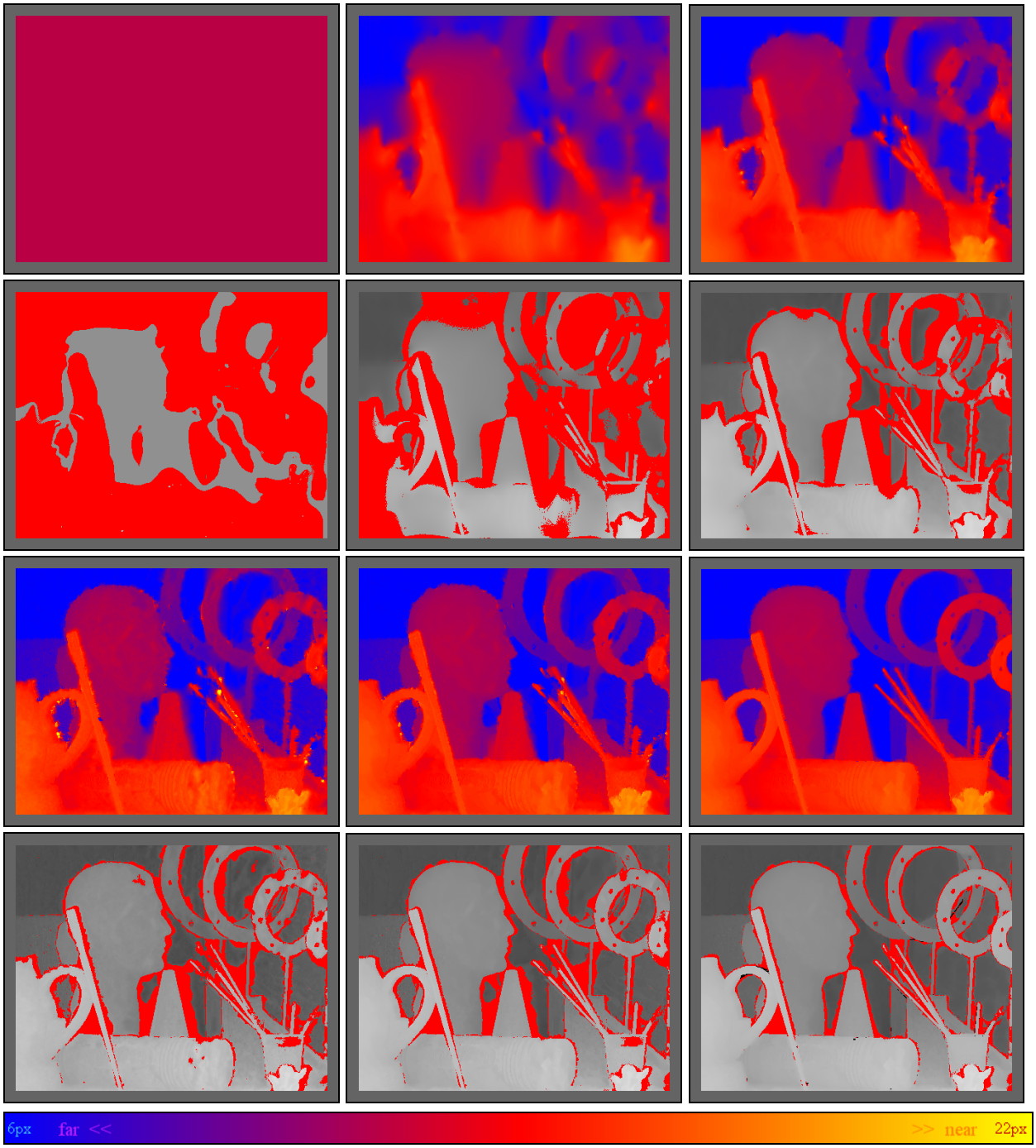

Figure 6.8: Experiment VI: Evolution of "Art" scene: **Top Row:** The disparity map before calculations, after the 2000 iterations and after 3000 iterations; **Second Row:** Corresponding to the top row distribution maps of bad pixels ($\delta = 1$); **Third Row:** The disparity map after 3,5, 3,8, 4 thousand of iterations; **Bottom Row:** Corresponding to the third row distribution maps of bad pixels ($\delta = 1$).

At the next one thousand iterations, the solver reduces both smoothness parameters φ and ρ, and creates more precise and less overblurred solution. The disparity map after this step looks like the usual initial data map for the finest level in case if we used the coarse levels technique. At the next step, the solver even more reduces the smoothness parameters. This time the small details appear, like brushes and small holes in the rings. Also at this time some undesirable noise appears (see figure 6.8, the first picture of the third row).

Starting from the 3500-th iteration the solver starts using Charbonnier penalization in data and smoothness terms, it rises again the smoothness parameter φ. That eliminates the noise from the previous step and due to the nonlinear regularization preserves small important details. At the last 2 hundred iterations of variational process, the solver reduces all the parameters to press out at the end as much details as possible – the smaller λ_d leads to penalizing the data term's influence in energy functional and therefore to the disparity map overblurring. That's why the parameter φ is also reduced. The smaller λ_s makes the borders of objects sharper (see figure 6.8).

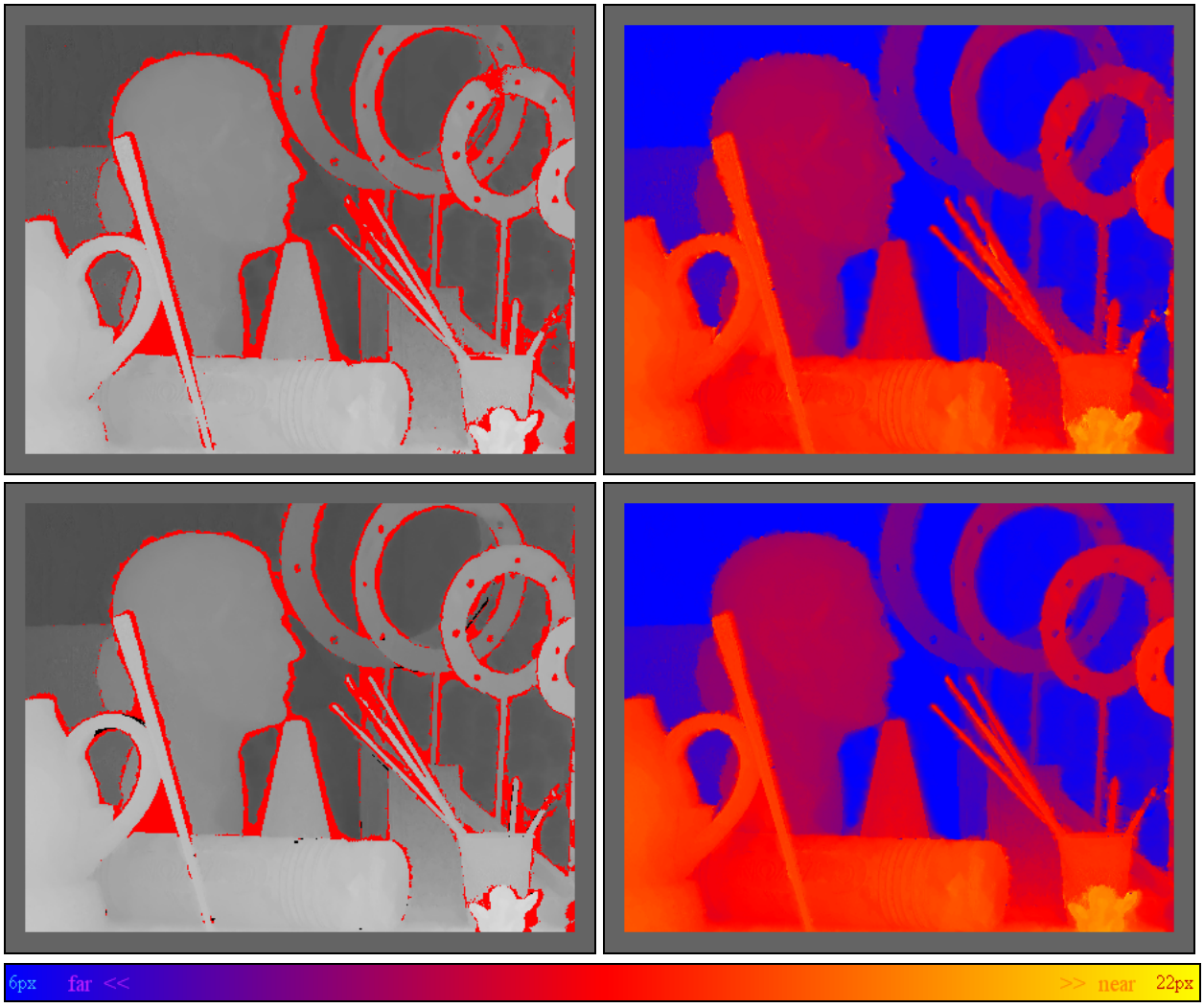

Figure 6.9: Experiment VI: "Art" scene: **Top Left:** Distribution of the bad pixels in the solution achieved by the linear interpolation method with using the coarse levels ($\delta = 1$); **Top Right:** Linear interpolation solution disparity map with using the coarse levels; **Bottom Left:** Distribution of the bad pixels in the solution achieved by the linear interpolation method without using the coarse levels ($\delta = 1$); **Bottom Right:** Linear interpolation solution disparity map without using the coarse levels.

At the figure 6.9 we compare the results achieved by the method of linear interpolation with and without using the coarse levels technique. As we can see the results are very similar except the small details. The method of automatic setup variable calibration during the calculations not only makes the variational process faster, but makes it also more sensitive to the small details and handles them wisely and precise.

In the table 6.10 we compare the results of the linear interpolation method with coarse levels (and without automatic adjustment technique) and without coarse levels (and with automatic adjustment technique):

Table 6.10: "Art" scene: percentage of bad pixels for different δ.

δ	Linear interpolation 8 CL	Linear interpolation 1 CL
0,5	*18,66 %*	*13,67 %*
0,75	*13,98 %*	*10,66 %*
1	*11,58 %*	*9,44 %*
1,5	*9,44 %*	*7,26 %*
2	*7,96 %*	*5,41 %*

We may conclude, that the method of linear interpolation in combination with the method of controlling the matching process and automatic setup parameters correction is faster than existing methods (we do not talk about the real-time methods, which use the multigrid techniques, which, besides, are possible also to apply to our method) may handle the problem of depth-map reconstruction without coarse levels, without warping technique and may give better results in means of percentage of bad pixels estimation. We finish this section with the table 6.11, where the capacities of the methods are compared:

Table 6.11: Method capacities versus the number of coarse levels.

CL	First order Taylor expansion	Linear interpolation
16	*+ warping*	+
8	–	+
1	–	*+ controlling*

6.4 Two cameras vs. multiple cameras

In this section we make experiments with the "Cones", "Teddy" and famous "Tsukuba" scenes. This time, each stereo image consists of five bitmap pictures. The stereo images of the "Cones" and "Teddy" scenes have resolution 450 x 375 pixels and the stereo image of "Tsukuba" scene has resolution 384 x 288 pixels. In the experiments of this section we used no coarse levels due to the method of linear interpolation and interactive parameters adjustment. The displacement of objects in the scenes varies between 4 and 16 pixels. The scale factor here is equal to 16.

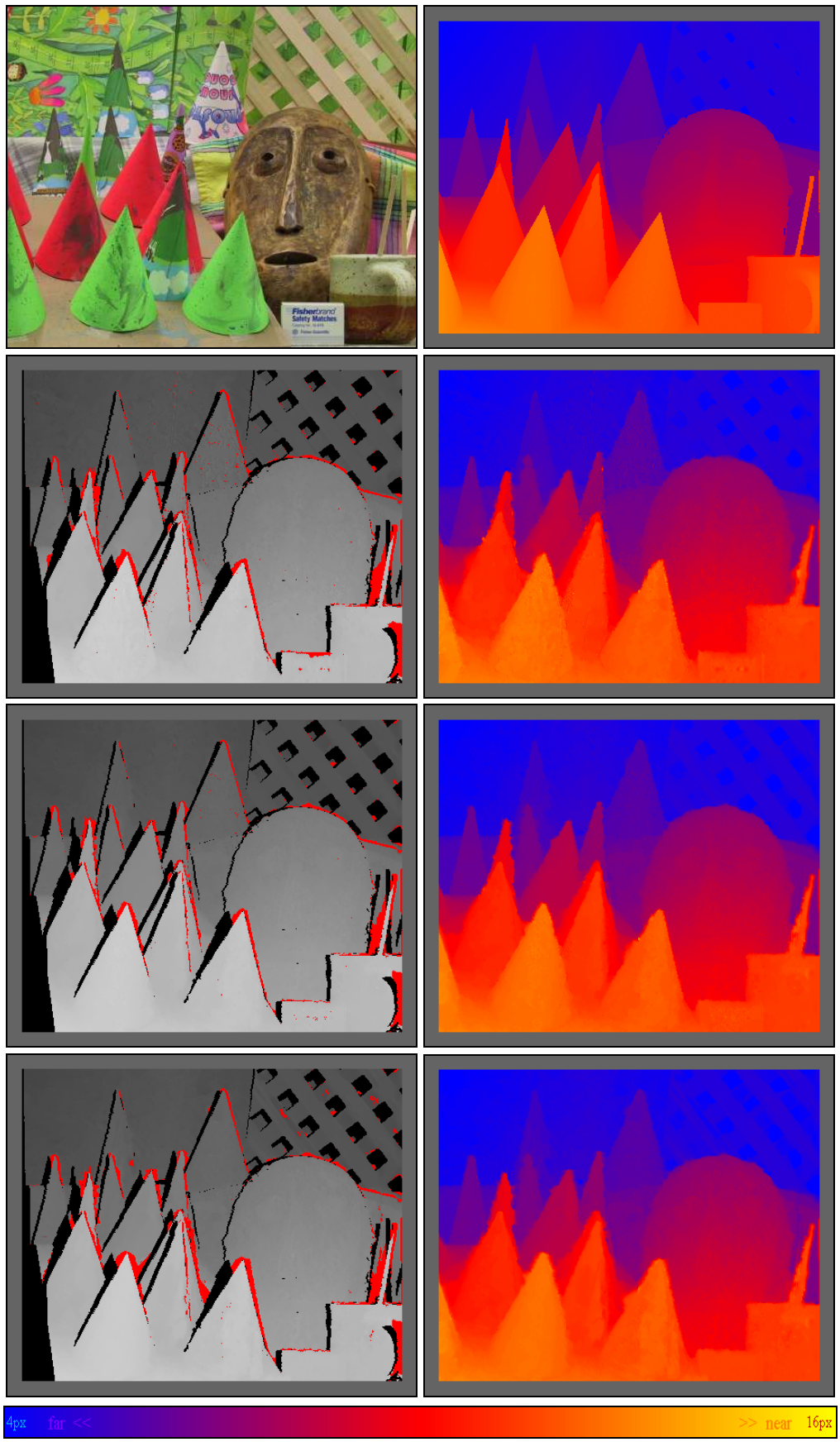

Figure 6.10: Experiment VII: "Cones" scene: **Top Left:** The middle frame of the scene; **Top Right:** Ground truth disparity map; **Left Column:** Distribution of the bad pixels in the solution achieved by the 2, 3 and 5 cameras from top till bottom ($\delta = 0{,}75$); **Right Column:** 2, 3 and 5 cameras solution disparity map from top till bottom.

At the figure 6.10 we can observe the "cones" scene and the results gained by using two, three and five cameras. As we can see they are very similar. Even the distributions of bad pixels for these solutions are almost indistinguishable by a human eye. So let us consider the table 6.12, where the precise percentages of bad pixels are brought:

Table 6.12: "Cones" scene, percentage of bad pixels for different δ.

δ	2 Cameras	3 Cameras	5 Cameras
0,5	4,44 %	3,76 %	5,09 %
0,75	3,18 %	2,79 %	3,46 %
1	2,32 %	2,04 %	2,52 %
1,5	1,46 %	1,36 %	1,65 %
2	1,04 %	1,01 %	1,15 %

Here we can conclude that the result disparity map, achieved by using information from the 3 cameras appears to be the best and the result disparity map, achieved by using information from the 5 cameras appears to be the worst. Anyway, the difference of percentage of bad pixels does not vary significantly and the method based on the 3 cameras does not win a lot from the method based on 2 cameras. At the other hand we conclude, that the additional information from additional cameras does not always help to achieve better results.

Now let us proceed to the next scene. If we look at the table 6.13 we can realise that the results for the "Teddy" scene is more interesting – here we can not say which of the solution disparity map is definitely better or worse:

Table 6.13: "Teddy" scene, percentage of bad pixels for different δ.

δ	2 Cameras	3 Cameras	5 Cameras
0,5	5,66 %	7,66 %	3,75 %
0,75	2,75 %	3,15 %	2,58 %
1	2,01 %	2,19 %	2 %
1,5	0,95 %	0,89 %	1,07 %
2	0,51 %	0,49 %	0,61 %

We see that the variational method that uses information from five cameras is more robust to the small details, since for the small threshold δ it gives the best results. But at the same time it gives the worse results for the large δ. It can be considered as an evidence of that the method, based on five cameras, does not precede the whole objects geometry precisely. Here the best result gives the method that uses information from three cameras. In this section we use only consequent frames of the stereo images, so our error estimation may be incomparable with the error measurement from the [WMB07]. Please refer to the appendix A

At the figure 6.11 the results for the threshold $\delta = 0.5$ are depicted. As we can see, the 5 cameras result is more blurry, that's why the regions inside objects are calculated more properly than in other methods – the information from tree or two cameras is not enough to fill these areas (see the distributions maps of the bad pixels at the next page):

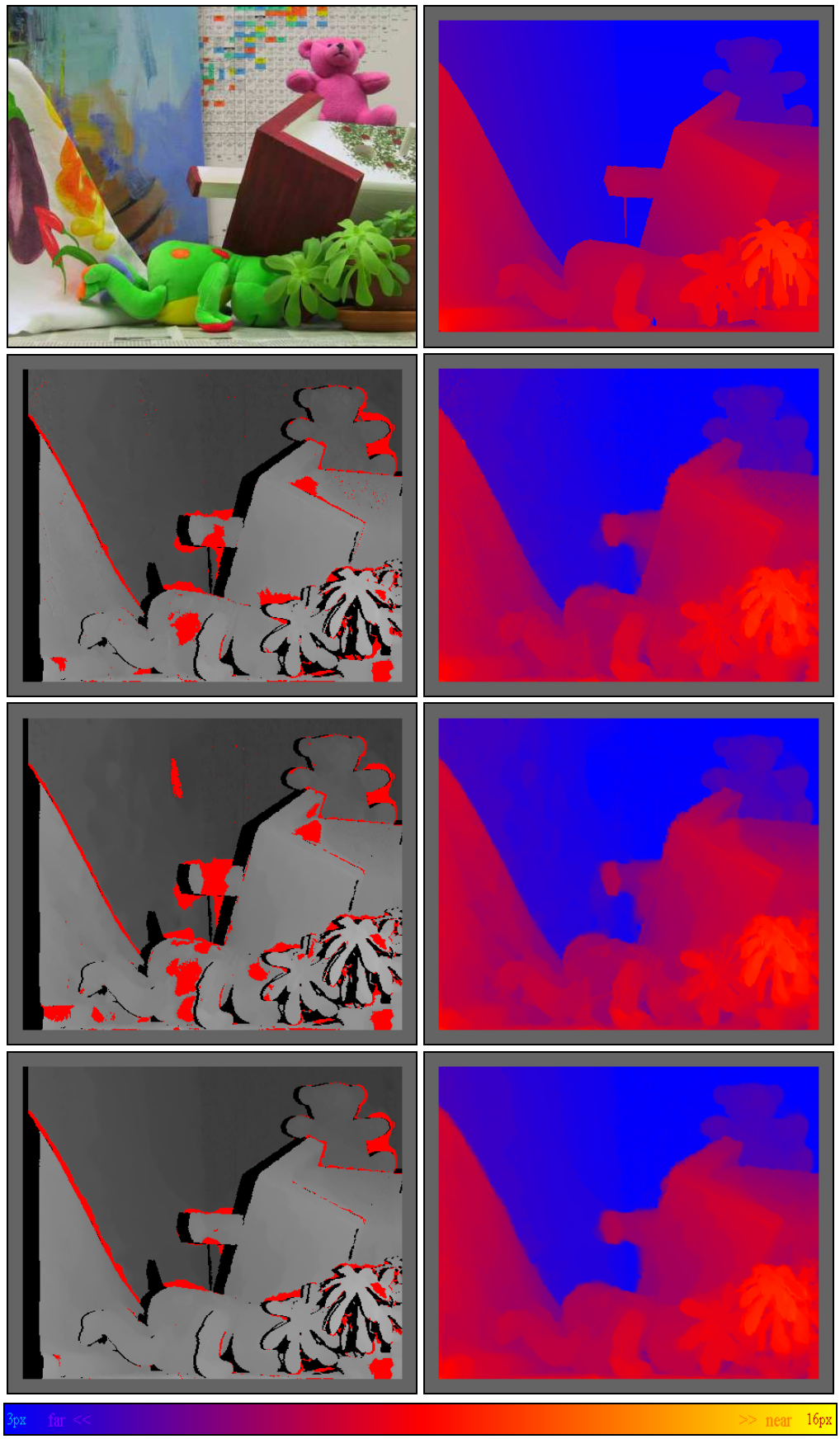

Figure 6.11: Experiment VIII: "Teddy" scene: **Top Left:** The middle frame of the scene; **Top Right:** Ground truth disparity map; **Left Column:** Distribution of the bad pixels in the solution achieved by the 2, 3 and 5 cameras from top till bottom ($\delta = 0{,}5$); **Right Column:** 2, 3 and 5 cameras solution disparity map from top till bottom.

Now let us discuss famous Tsukuba scene. First of all watch the figure 6.12, where the results for three and five cameras are compared. We do not consider now the case of two cameras, since we know already the capability of the method and in this section we focus the attention on the multi – view depth-map reconstruction and further comparison with the results of other methods:

Figure 6.12: Experiment IX: "Tsukuba" scene: **Top Left:** The middle frame of the scene; **Top Right:** Ground truth disparity map; **Left Column:** Distribution of the bad pixels in the solution achieved by the 3 and 5 cameras from top till bottom ($\delta = 1$); **Right Column:** 3 and 5 cameras solution disparity map from top till bottom.

At the figure 6.12 we see that the results for the three and five cameras reconstruction could be predicted by the previous conclusions of the experiments with the "Cones" and "Teddy" scenes. Looking at the distributions of bad pixels, we can notice that the distribution map for the five cameras case is richer with the red color. But considering these maps more closely, we see that the distribution map of bad pixels in the solution, achieved by the five cameras method has no red pixel at the background as well as it has much more less "stand alone" red pixels. We just conclude again, that the method based on five cameras is more accurate with the small details but still has problems with the whole objects' geometry: if we look at the solution disparity maps we see that the objects at the 5 cameras solution disparity map appear to be "wider" and more blurry than the objects at the 3 cameras solution disparity map.

We would like to compare our results with the results, achieved by other existing methods. The computer vision department's web site of the Middlebury University [WMB07] provides us with the possibility to familiarize with the capabilities of more than 30 depth-map reconstruction methods and to compare theirs solution disparity maps for the "Tsukuba" scene with our own solution. Since the disparity map, calculated with the help of the variational method in case of 3 cameras gives us better results than in case of 5 cameras (see table 6.14), we compare it with the most interesting three other results, calculated with help of alternative methods: the infection method [OFPL06], the method of dynamic programming, similar to Bobick and Intille [SS02] and the graph cuts method [KZ01]. These depth maps and theirs distributions of bad pixels are illustrated at the figure 6.13.

Table 6.14: "Tsukuba" scene, percentage of bad pixels for different δ.

δ	VM[7] 3Cam	VM 5Cam	Infection	Dyn. Prog.	Graph Cuts
0,5	*18,4 %*	*13,6 %*	*22,1 %*	*19,6 %*	*6,19 %*
0,75	*8,17 %*	*9,86 %*	*21,9 %*	*19,6 %*	*6,10 %*
1	*6,19 %*	*8,17 %*	*7,95 %*	*4,12 %*	*1,19 %*
1,5	*4,62 %*	*6,53 %*	*7,36 %*	*4,12 %*	*1,19 %*
2	*3,9 %*	*5,6 %*	*6,34 %*	*3,43 %*	*0,88 %*

In the table 6.14 we observe the percentage of bad pixels for our method and alternative methods. We must confess that nowadays the graph cuts method allows gaining the best results on "Tsukuba" scene. Nevertheless, as we can see from the table 6.14, the variational method, being a new and young approach for depth-map reconstruction, is not only competitive with other methods, but even takes the high stand among them.

At the figure 6.13 we compare the resulting disparity maps of variational method with alternative ones. The distributions of bad pixels for the $\delta = 1$ are also illustrated. As we can see, the graph cuts method produces the best disparity map, and other methods introduce some distortions. The result of the variational method looks pretty good among other ones and we hope that further development will only improve it and spread it widely.

[7] Abbreviation from "variational method" (VM).

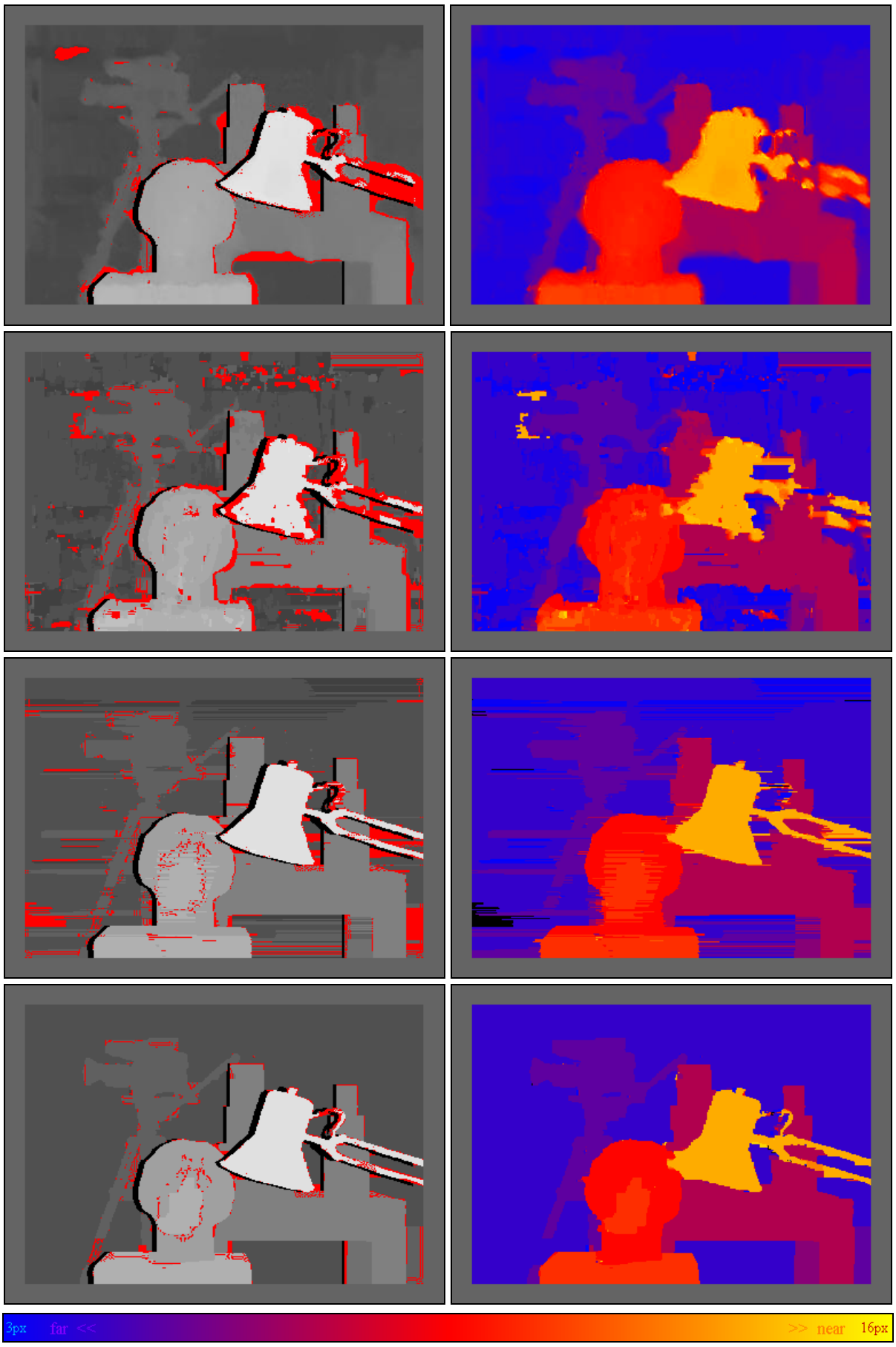

Figure 6.13: Comparison of computer vision methods on "Tsukuba" scene ($\delta = 1$): **Top Row:** Our method; **Second Row:** Infection method; **Third Row:** Method of dynamic programming; **Bottom Row:** Graph cuts method.

6.5 Conclusion

The variational methods are the most successful methods used for optic flow computation. One of the contributions of this work is the investigation of application of the famous variational methods of Horn-Schunck and Brox et al., to the problem of depth-map reconstruction. We combine multiple constancy assumptions in the data term and use different penalizing functions in the smoothness term. These penalizing functions are bound with the diffusion process that is well known from physics. We have discussed two approaches for the depth-map reconstruction: the depth-driven and the disparity-driven; and investigated the behavior of the data term and the smoothness term in the energy functionals constructed in accordance to these approaches. For this purpose we have presented a powerful tool for analyzing the smoothness process during the reconstruction – the RnB pyramid. Theoretical issues and experiments showed the advantages of the disparity-driven method.

The main goal of all the research done in this thesis is to find a suitable variational method for computing a correct depth-map for real-world data. During this research a few fundamental improvements were discovered and offered. First of all, we have offered to replace the popular nowadays method of the first order Taylor expansion, which is used to get rid of implicitness in the data term and linearize it, with the more general method of the linear interpolation. This improvement, almost without any additional computational effort, allowed us to handle large object's displacements in pictures of a stereo image and to get rid of nasty warping technique, when using coarse levels. The problem of the depth-map reconstruction with variational methods – as the major part of the computer vision problems – leads to the iterative numerical problem, or by other words, the process of calculation the depth-map in our case is the iteration process. The second improvement, that we offered, is the automatic trace of the convergence of the iteration process on run and even the control of it with the help of invented technique. Such an improvement not only uncover the possibility to speed up the process and release it from errors, but in combination with the method of linear interpolation, mentioned right above, allows to get rid of coarse levels at all. In the experimental sections of this thesis we have shown that these innovations work for real-world data and give much better results than current techniques and methods. Moreover, we have to mention, that the described improvements could be applied to the original optic flow methods.

Another goal of this thesis is to define how the number of cameras and their position / orientation influences the resulting depth-map. In order to successfully accomplish this task, the general theory of multi – view depth-map reconstruction in the meaning of the two frame model extension was constructed, applied and evaluated. We made a numerous experiments and compared our results with the alternative methods for the depth-map reconstruction.

We hope that with our work we have managed to connect the worlds of computer vision with the world of variational methods. We further hope that our efforts have contributed to the improvement of the highly accurate depth-map computation techniques and the variational methods.

Appendix A

In the section 6.4 we compared the variational method with the alternative methods for the depth-map reconstruction. Here we estimate gained results with the other computer vision methods. We used the experiment setup strategy and the error estimator, provides by the Middlebury University. Hereby we put here the table of the percentages of bad pixels, generated at the Middlebury University webpage.

The table includes the error estimations for 4 scenes: "Tsukuba", "Venus", "Teddy" and "Cones". The first column contains the short name of a method, the second column – the average rank of the method. Black numbers denote the percentage of bad pixels, blue numbers – the rank of the method within the current column. The table uses the following abbreviations:

- *nonocc* – non occluded pixels only – during the estimation occluded pixels were not considered;
- *all* – all the pixels – during the estimation all the pixels were considered;
- *disc* – discontinuities – during the estimation only pixels from the areas were the discontinuity arises were considered.

During the experiment the following results were used:

- *"Tsukuba" scene* – 3 cameras result, linear interpolation and the matching process controlling techniques were used. No coarse levels, no warping. Scale factor is equal to 16.
- *"Venus" scene* – 3 cameras result, linear interpolation and the matching process controlling techniques were used. No coarse levels, no warping. Scale factor is equal to 8.
- *"Teddy" scene* – 5 cameras result, linear interpolation and the matching process controlling techniques were used. No coarse levels, no warping. Scale factor is equal to 4. (In the table 6.13 the scale factor while estimation is equal to 16).
- *"Cones" scene* – 3 cameras result, linear interpolation and the matching process controlling techniques were used. No coarse levels, no warping. Scale factor is equal to 4. (In the table 6.12 the scale factor while estimation is equal to 16).

For more details, please refer to [WMB07].

$\delta = 1$

Algorithm	Avg	Tsukuba			Venus			Teddy			Cones		
	Rank	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc
AdaptingBP	2.4	1.11 5	1.37 2	5.79 6	**0.10** 1	0.21 2	**1.44** 1	4.22 3	7.06 2	11.8 3	**2.48** 1	7.92 2	**7.32** 1
DoubleBP	3.8	**0.88** 1	**1.29** 1	**4.76** 1	0.14 4	0.60 10	2.00 6	3.55 2	8.71 5	**9.70** 1	2.90 3	9.24 9	7.80 2
SubPixDoubleBP	4.6	1.24 8	1.76 10	5.98 7	0.12 2	0.46 4	1.74 4	**3.45** 1	8.38 4	10.0 2	2.93 4	8.73 6	7.91 3
AdaptOvrSegBP	8.6	1.69 18	2.04 17	5.64 5	0.14 3	**0.20** 1	1.47 2	7.04 13	11.1 7	16.4 11	3.60 9	8.96 8	8.84 9
PlaneFitBP	8.9	0.97 4	1.83 11	5.26 4	0.17 6	0.51 5	1.71 3	6.65 9	12.1 11	14.7 6	4.17 16	10.7 17	10.6 15
SymBP+occ	9.3	0.97 3	1.75 9	5.09 3	0.16 5	0.33 3	2.19 7	6.47 8	10.7 6	17.0 14	4.79 20	10.7 18	10.9 16
Segm+visib	9.8	1.30 12	1.57 3	6.92 15	0.79 17	1.06 15	6.76 18	5.00 4	**6.54** 1	12.3 4	3.72 10	8.62 5	10.2 13
C-SemiGlob	10.3	2.61 25	3.29 20	9.89 22	0.25 9	0.57 7	3.24 12	5.14 5	11.8 8	13.0 5	2.77 2	8.35 4	8.20 4
SO+borders	10.4	1.29 11	1.71 6	6.83 12	0.25 10	0.53 6	2.26 8	7.02 12	12.2 12	16.3 9	3.90 12	9.85 13	10.2 14
DistinctSM	11.8	1.21 7	1.75 8	6.39 9	0.35 11	0.69 13	2.63 11	7.45 17	13.0 15	18.1 17	3.91 13	9.91 15	8.32 6
OverSegmBP	12.0	1.69 19	1.97 14	8.47 19	0.50 14	0.68 12	4.69 15	6.74 10	11.9 10	15.8 7	3.19 7	8.81 7	8.89 10
SegmentSupport	12.3	1.25 9	1.62 4	6.68 11	0.25 8	0.64 11	2.59 10	8.43 21	14.2 19	18.2 18	3.77 11	9.87 14	9.77 12
RegionTreeDP	13.0	1.39 15	1.64 5	6.85 13	0.22 7	0.57 7	1.93 5	7.42 16	11.9 9	16.8 13	6.31 24	11.9 23	11.8 19
EnhancedBP	13.8	0.94 2	1.74 7	5.05 2	0.35 12	0.86 14	4.34 14	8.11 19	13.3 17	18.5 20	5.09 22	11.1 19	11.0 17
AdaptWeight	14.7	1.38 14	1.85 12	6.90 14	0.71 15	1.19 16	6.13 16	7.88 18	13.3 18	18.6 22	3.97 15	9.79 11	8.26 5
SegTreeDP	14.9	2.21 23	2.76 18	10.3 24	0.46 13	0.60 9	2.44 9	9.58 24	15.2 23	18.4 19	3.23 8	**7.86** 1	8.83 8
ImproveSubPix	15.4	3.00 26	3.61 23	10.9 26	0.88 18	1.47 17	7.10 20	7.12 14	12.4 14	16.6 12	2.96 5	8.22 3	8.55 7
SemiGlob	16.4	3.26 27	3.96 24	12.8 29	1.00 19	1.57 18	11.3 24	6.02 6	12.2 13	16.3 10	3.06 6	9.75 10	8.90 11
RealtimeBP	19.3	1.49 16	3.40 22	7.87 17	0.77 16	1.90 21	9.00 23	8.72 23	13.2 16	17.2 15	4.61 18	11.6 21	12.4 23
GC+occ	20.3	1.19 6	2.01 16	6.24 8	1.64 24	2.19 23	6.75 17	11.2 27	17.4 27	19.8 25	5.36 23	12.4 24	13.0 24
Layered	20.6	1.57 17	1.87 13	8.28 18	1.34 21	1.85 19	6.85 19	8.64 22	14.3 20	18.5 21	6.59 26	14.7 26	14.4 25
MultiCamGC	21.0	1.27 10	1.99 15	6.48 10	2.79 30	3.13 27	3.60 13	12.0 28	17.6 28	22.0 27	4.89 21	11.8 22	12.1 21
GenModel	23.3	2.57 24	4.74 27	13.0 30	1.72 25	3.08 26	16.9 28	6.86 11	15.0 22	19.2 23	4.64 19	14.9 27	11.4 18
RealTimeGPU	23.7	2.05 22	4.22 26	10.6 25	1.92 27	2.98 25	20.3 30	7.23 15	14.4 21	17.6 16	6.41 25	13.7 25	16.5 27
OUR METHOD	23.8	6.19 35	8.23 35	28.6 36	2.70 29	3.52 30	30.7 35	6.04 7	8.17 3	15.8 8	7.46 27	9.82 12	18.2 28
CostRelax	24.3	4.76 31	6.08 30	20.3 33	1.41 23	2.48 24	18.5 29	8.18 20	15.9 24	23.8 28	3.91 14	10.2 16	11.8 20
ReliabilityDP	25.5	1.36 13	3.39 21	7.25 16	2.35 28	3.48 29	12.2 27	9.82 26	16.9 25	19.5 24	12.9 34	19.9 33	19.7 30
TreeDP	26.2	1.99 21	2.84 19	9.96 23	1.41 22	2.10 22	7.74 21	15.9 32	23.9 32	27.1 33	10.0 30	18.3 30	18.9 29
GC	26.8	1.94 20	4.12 25	9.39 21	1.79 26	3.44 28	8.75 22	16.5 33	25.0 34	24.9 30	7.70 28	18.2 29	15.3 26
DP	30.6	4.12 29	5.04 29	12.0 27	10.1 37	11.0 37	21.0 31	14.0 29	21.6 29	20.6 26	10.5 31	19.1 31	21.1 31
PhaseBased	31.8	4.26 30	6.53 31	15.4 31	6.71 33	8.16 33	26.4 34	14.5 30	23.1 30	25.5 31	10.8 33	20.5 34	21.2 32
SSD+MF	32.3	5.23 34	7.07 32	24.1 34	3.74 31	5.16 31	11.9 26	16.5 34	24.8 33	32.9 35	10.6 32	19.8 32	26.3 34
STICA	33.7	7.70 36	9.63 37	27.8 35	8.19 34	9.58 34	40.3 37	15.8 31	23.2 31	37.7 36	9.80 29	17.8 28	28.7 36
SO	34.0	5.08 33	7.22 34	12.2 28	9.44 36	10.9 36	21.9 32	19.9 36	28.2 37	26.3 32	13.0 35	22.8 36	22.3 33
PhaseDiff	34.7	4.89 32	7.11 33	16.3 32	8.34 35	9.76 35	26.0 33	20.0 37	28.0 36	29.0 34	19.8 37	28.5 37	27.5 35
Infection	35.4	7.95 37	9.54 36	28.9 37	4.41 32	5.53 32	31.7 36	17.7 35	25.1 35	44.4 37	14.3 36	21.3 35	38.0 37

Bibliography

[AK02] G. Aubert, P. Kornprobst. *Mathematical Problems in Image Processing: Partial Differential Equations and the Calculus of Variations.* Volume 147 of Applied Mathematical Sciences. Springer, New York, 2002.

[ALM92] L. Alvarez, P.L. Lions, J.M. Morel. *Image selective smoothing and edge detection by nonlinear diffusion (ii).* Volume 29 of SIAM Journal on Numerical Analysis, pp. 845-866. Society for Industrial and Applied Mathematics, 1992.

[BA96] M. J. Black, P. Anandan. *The robust estimation of multiple motions: parametric and piecewise smooth flow fields*. Computer Vision and Image Understanding, 63(1):75-104, January 1996.

[BAK91] R. Battiti, E. Amaldi, C. Koch. *Computing optical flow across multiple scales: an adaptive coarse-to-fine strategy*. International Journal of Computer Vision, 6(2):133-145, 1991.

[BBPW04] T. Brox, A. Bruhn, N. Papenberg, J. Weickert. *High accuracy optic flow estimation based on a theory for warping*. In T. Pajdla and J. Matas, editors, Computer Vision ECCV 2004, volume 3024 of Lecture Notes in Computer Science, pp 25-36. Springer, Berlin, 2004.

[BCAB95] L. Blanc-Ferraud, P. Charbonnier, G. Aubert, M. Barlaud. *Nonlinear image processing: modeling and fast algorithm for regularization with edge detection*. Volume 1 of 1995 International Conference on Image Processing, p. 474. 1995.

[Bru06] A. Bruhn. *Variational optic flow computation: accurate modelling and efficient numerics*. PhD thesis, Department of Mathematics and Comptuer Science, Saarland University, Germany, July 2006.

[BWSW03] T. Brox, M. Welk, G. Steidl, J. Weickert. *Equivalence results for TV diffusion and TV regularization*. In L. D. Griffin and M. Lillholm, editors, Scale-Space Methods in Computer Vision, volume 2695 of Lecture Notes in Computer Science, pp. 86-100. Springer, Berlin, June 2003.

[CABB94] P. Charbonnier, G. Aubert, M. Blanc-Ferraud, M. Barlaud. *Two-deterministic half-quadratic regularization algorithms for computed imaging*. Volume 2 of International Conference on Image Processing, pp. 168-172. November 1994.

[DKA95] R. Deriche, P. Kornprobst, G. Aubert. *Optical-flow estimation while preserving its discontinuities: a variational approach*. Volume 2 of Second Asian Conference on Computer Vision, pp. 290-295. Singapore, December 1995.

[Els61] L. E. Elsgolc. *Calculus of Variations*. Pergamon, Oxford, 1961.

[GR92] S. Geman, G. Reynolds. *Constrained restoration and the recovery of discontinuities*. Volume 14 of IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 367-383. March 1992.

[HJ94] R. A. Horn, C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, Cambridge, UK, 1994.

[HS81] B. Horn, B. Schunck. *Determining optical flow*. Volume 17 of Artificial Intelligence, pp. 185-203, 1981.

[KB07] F. Kücükay, J. Bergholz. *Driver assistant systems*. Scientific report. Could be found on http://www.osd.org.tr/. Last accessed: December 2007.

[KZ01] V. Kolmogorov, R. Zabih. *Computing visual correspondence with occlusions using graph cuts.* Volume 2 of International Conference on Computer Vision, pp. 508-515, 2001.

[Mei02] E. Meijering. *A chronology of interpolation: from ancient astronomy to modern signal and image processing*. Volume 90 of proceedings of the IEEE, pp. 319-342. March 2002.

[OFPL06] G. Olague, F. Fernández, C. Pérez, and E. Lutton. *The infection algorithm: an artificial epidemic approach for dense stereo correspondence.* Volume 12 of Artificial Life, pp. 593-615. MIT Press, Cambridge, USA, October 2006.

[PM90] P. Perona, J. Malik. *Scale-space and edge detection using anisotropic diffusion*. Volume 12 of IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 629-639. IEEE Computer Society, June 1990.

[PH04] M. Pharr, G. Humphreys. *Physically based rendering: from theory to implementation*. Morgan Kaufmann publishers, 2004.

[RD96] L. Robert, R. Deriche. *Dense map reconstruction: a minimization and regularization approach with preserves discontinuities.* Volume 1064 of Lecture Notes in Computer Science, pp. 439-451. Springer, Berlin, 1996.

[Rob95] L Robert. *Camera calibration without feature extraction*. Volume 63 of Computer Vision, and Image Understanding, pp. 314-325. 1995.

[SS02] D. Scharstein, R. Szeliski. *A taxonomy and evaluation of dense two-frame stereo correspondence algorithms.* Volume 47 of International Journal of Computer Vision, pp. 7-42. Springer, Netherlands, April 2002.

[Ter86] D. Terzopoulos. *Image analysis using multigrid relaxation methods*. Volume 8 of IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 129-139, March 1986.

[TP84] O. Tretiak, L. Pastor. *Velocity estimation from image sequences with second order differential operators*. In Proc. Seventh International Conference on Pattern Recognition, pp 16-19. Montreal, Canada, July 1984.

[Wei98] J. Weickert. *Anisotropic diffusion in image processing*. Teubner, Stuttgart, 1998.

[WF07] Fishki, the intertainment portal. *http://www.fishki.net/.* Last accessed: December 2007.

[WMB07] Middlebury stereo page. *http://vision.middlebury.edu/stereo/.* Last accessed: December 2007.

[WMW07] MathWorld. *http://mathworld.wolfram.com/.* Last accessed: December 2007.

[WPX07] Project X. Image processing program. *http://www.project-10.de/.* Last accessed: December 2007.

[WRM07] Robot Maximilian. *http://www.howtoandroid.com/.* Last accessed: December 2007.

[WW07] Wikipedia, the free encyclopedia. *http://www.wikipedia.org/*. Last accessed: December 2007.

[WWS05] M. Welk, J. Weickert, G. Steidl. *A four-pixel scheme for singular differential equations.* Volume 3459 of Lecture notes in computer science, pp. 610-621. Springler, Berlin, March 2005.

[YHR04] I. A. Ypsilos, A. Hilton, S. Rowe. *Video-rate capture of dynamic face shape and appearance*. In Proc. of IEEE International Conference on Automatic Face and Gesture Recognition, pp. 117-122. May 2004.